ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Διατριβή

ΣΥΝΔΥΑΣΜΕΝΗ ΧΡΗΣΗ ΡΟΜΠΟΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗΣ ΟΡΑΣΗΣ ΣΕ ΕΦΑΡΜΟΓΕΣ ΑΣΦΑΛΕΙΑΣ

Βούρκος Ελευθέριος

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΜΗΧΑΝΟΛΟΓΙΑΣ ΚΑΙ ΚΑΤΑΣΚΕΥΑΣΤΙΚΗΣ

ΜΑΙΟΣ 2023

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΜΗΧΑΝΟΛΟΓΙΑΣ
ΚΑΙ ΚΑΤΑΣΚΕΥΑΣΤΙΚΗΣ

Συνδυασμένη Χρήση Ρομποτικών Συστημάτων και Μηχανικής Όρασης σε Εφαρμογές Ασφαλείας

Βούρκος Ελευθέριος

Επιβλέποντες

Δρ. Χριστοφόρου Ευτύχιος

(Τμήμα Μηχανικών Μηχανολογίας και Κατασκευαστικής, Πανεπιστήμιο Κύπρου)

Δρ. Παναγίδης Αντρέας

(CYENS Centre of Excellence)

Η Διατριβή υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων του Master of Science του Τμήματος Μηχανικών Μηχανολογίας και Κατασκευαστικής

Μάιος 2023

Ευχαριστίες

Πολλές ευχαριστίες στον Δρ. Ευτύχιο Χριστοφόρου για την υποστήριξη και την αμέριστη συμπαράσταση καθ' όλη την διάρκεια του Μεταπτυχιακού μου. Ένα μεγάλο ευχαριστώ για την εμπιστοσύνη που μου έχει δείξει, καθώς επίσης και για τις ευκαιρίες που μου έχει προσφέρει τόσο σε ακαδημαϊκό επίπεδο, όσο και σε επαγγελματικό.

Επιπλέον, θα ήθελα να ευχαριστήσω τον Δρ. Αντρέα Παναγίδη για την άψογη συνεργασία και διαρκή καθοδήγηση στην πορεία για εκπόνηση της παρούσας Διατριβής. Το γεγονός ότι σε τόσο σύντομο χρονικό διάστημα κατέστη εφικτό να αποκτήσω τις απαιτούμενες γνώσεις στην περιοχή της Μηχανικής Όρασης οφείλεται αποκλειστικά σε αυτόν. Τον ευχαριστώ επίσης και για την ευκαιρία συμμετοχής στην διαδικασία δημιουργίας δημοσίευσης. Ένα μεγάλο ευχαριστώ στο CYENS CoE τόσο για την ευκαιρία πρακτικής άσκησης, όσο και για την μετέπειτα εργοδότηση. Η συνεισφορά τους ήταν πραγματικά πολύτιμη, όπως επίσης πολύτιμο είναι και το να εργάζεσαι σε ένα τόσο φιλόξενο περιβάλλον.

Παράληψη θα αποτελούσε να μην ευχαριστήσω τον συνάδελφο Αντρέα Κούρρη προπτυχιακό φοιτητή στο Τμήμα μας, για την άμεση ανταπόκριση σε κάθε μου κάλεσμα για τεχνητή υποστήριξη. Οι διευρυμένες γνώσεις του στον τομέα της Ρομποτικής ήταν πραγματικά πολύτιμες.

Ένα τεράστιο ευχαριστώ στην οικογένειά μου για την συνεχή στήριξη με κάθε τρόπο σε όλα τα επίπεδα. Ιδιαίτερα όμως ευχαριστώ τον γιο μου Αντρέα Βούρκο ο οποίος έγινε η αιτία να θέτω υψηλούς στόχους και να αγωνίζομαι νυχθημερόν για την επίτευξή τους.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια της FreshLand και συγκεκριμένα στους Φίλιππο Παπά, Δημήτρη Δημητρίου και Νικόλα Δημητρίου για την διαρκή υποστήριξη σε αυτά τα 4 χρόνια που είμαστε μαζί.

Περίληψη

Η παρούσα Διατριβή αφορά εφαρμογές θεμάτων ασφαλείας με συνδυασμό ρομποτικών συστημάτων και όρασης. Ο έλεγχος του ρομπότ βασίζεται στο ROS (Robot Operating System) και περιλαμβάνει mapping, localization, path planning και navigation. Η όραση επικεντρώνεται στην ανίχνευση αντικειμένων σε πραγματικό χρόνο χρησιμοποιώντας διάφορες εκδόσεις του YOLO (You Only Look Once), το οποίο συνδυάστηκε επιτυχώς με το ROS. Έπειτα, παρουσιάζεται εφαρμογή στις υποδομές του Πανεπιστημίου Κύπρου (Κεντρικά - Καλλιπόλεως). Για τον σκοπό αυτό χρησιμοποιήθηκαν ένα τροχοφόρο ρομπότ και ένα drone. Από πλευράς ρομποτικής όρασης έγινε χρήση θερμικής και μη κάμερας.

Περιεχόμενα

Εικόνες

Παραρτήματα

Κεφάλαιο 1

1 ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγή

«Τεχνολογία είναι το κόλπο για να οργανώνουμε τον Κόσμο με τρόπο τέτοιο, ώστε να μην χρειάζεται να τον βιώνουμε», είπε ο Ελβετός Συγγραφέας Max Frisch (1911 - 1991) [59] και ποιος θα τολμούσε να τον αμφισβητήσει άλλωστε εν έτει 2023. Δεν το επιτρέπουν εξάλλου τα αμέτρητα παραδείγματα τόσο σε προσωπικό, όσο και σε συλλογικό επίπεδο, αρχής γενομένης από το τηλέφωνο και κατ' επέκταση τις τόσες έξυπνες εφαρμογές που το στελεχώνουν.

Η παρούσα Διατριβή επιχειρεί να συζεύξει δύο κόσμους με σημαντικές διαφορές κατά κύριο λόγο ως προς την υποδομή τους. Από την μία η αυτοματοποίηση μίας διαδικασία πλοήγησης, με τον ολοκληρωμένο έλεγχο του ρομποτικού συστήματος αρχικά μέσω της δημιουργίας του χάρτη και έπειτα με την πλοήγηση στο προκαθορισμένο μονοπάτι στην χαρτογραφημένη περιοχή. Στον αντίποδα, η Τεχνητή Όραση αποτελεί μία από τις πιο δημοφιλής περιοχές που προσελκύουν το ενδιαφέρον του επιστημονικού κοινού σε μεγαλύτερο βαθμό μέρα παρά μέρα.

Τα δύο πεδία, παρόλο που έχουν εκπλήξει ευχάριστα το ευρύ κοινό, με τα αποτελέσματα στις ποικίλες εφαρμογές, βρίσκονται ωστόσο σε πολύ αρχικό στάδιο. Υπάρχουν διάφορες εφαρμογές που συνδυάζουν τον χειρισμό ρομπότ με την τεχνητή νοημοσύνη εξάγοντας πραγματικά εκπληκτικά αποτελέσματα. Όταν όμως τα αποτελέσματα αυτά χαρακτηρίζονται ως πειραματικά και παρουσιάζουν επίσης σημαντικό σφάλμα, τότε αντιλαμβάνεται κανείς τις δυνατότητες που προκύπτουν με το πάντρεμα των δύο περιοχών.

Συνοψίζοντας, η επιστημονική συνεισφορά του συνολικού έργου αφορά ουσιαστικά τον συνδυασμό του ρομποτικού συστήματος λειτουργίας ROS με τον αλγόριθμο αναγνώρισης αντικειμένων YOLOv7. Κατ' επέκταση έχει επιτευχθεί ο συνδυασμός ρομποτικών συστημάτων και όρασης συμπεριλαμβανομένου θερμικής κάμερας. Τέλος, παρατίθεται η επιτυχημένη επίδειξη ολοκληρωμένου συστήματος σε εφαρμογές ασφαλείας.

Η βιβλιογραφική μελέτη κάνει αρχή με μία ανασκόπηση στα ήδη υπάρχοντα ρομπότ που είναι εξοπλισμένα με αισθητήρες. Στο Κεφάλαιο αυτό παρουσιάζεται μία πρώτη κατηγοριοποίηση ούτως ώστε να διαφανεί ο σκοπός λειτουργίας της κάθε συσκευής και ακολουθεί τεχνική ανάλυση της κάθε ομάδας με τους βασικούς στόχους και τον εξοπλισμό να είναι τα κύρια ζητήματα. Έπειτα, διαγράφεται ο τρόπος με τον οποίο αλληλοεπιδρούν τα διάφορα στοιχεία προκειμένου να επιτευχθεί το επιθυμητό αποτέλεσμα. Στη συνέχεια παρατίθενται οι μέχρι τώρα εφαρμογές του συγκεκριμένου τύπου αισθητήρων ενώ στο Κεφάλαιο 3 ξετυλίγεται η πορεία δράσης προς την υλοποίηση της αναγνώρισης ανθρώπων ή άλλων αντικειμένων.

Εικόνα 1-1: Surveillance Robot [39].

Η αποτύπωση της διαδικασίας κάνει στάση στον αλγόριθμο για αναγνώριση αντικειμένων κατά την διάρκεια ροής της ταινίας που εξασφαλίζεται από την κάμερα. Στο σημείο αυτό, τίθεται το ζήτημα αμεσότητας και το χρονικό κενό που μεσολαβεί από την στιγμή ανίχνευσης της ανθρώπινης οντότητας έως την αναγνώριση. Τέλος, επισημαίνονται οι δυσκολίες στην λειτουργία και οι ιδιαιτερότητες χρήσης των διαφόρων συστημάτων όπως επίσης και οι ήδη υπάρχουσες λύσεις και τρόποι αντιμετώπισης. Η βιβλιογραφική μελέτη καταλήγει στις μελλοντικές εφαρμογές των συστημάτων και στον τρόπο αναβάθμισης των υφισταμένων.

Στα κεφάλαια που παρατίθενται στη συνέχεια, παρουσιάζεται η διαδικασία πλοήγησης του SUMMIT XL GEN, ρομπότ που αγοράστηκε από την εταιρεία Robotnik που αποτελεί ένα από τα πιο δυνατά «όπλα» στο Εργαστήριο Ρομποτικής του Τμήματος Μηχανικών Μηχανολογίας και Κατασκευαστικής του Πανεπιστημίου

Κύπρου (UCY Robotics Lab). Ακολούθως, παρουσιάζεται το YOLO (You Only Look Once), ένα μοντέλο Deep Learning που χρησιμοποιείται για ανίχνευση σε εικόνες και βίντεο. Τα αποτελέσματα που παρουσιάζονται αφορούν τον συνδυασμό YOLO και SUMMIT XL σε εφαρμογές επιτήρησης και ασφάλειας.

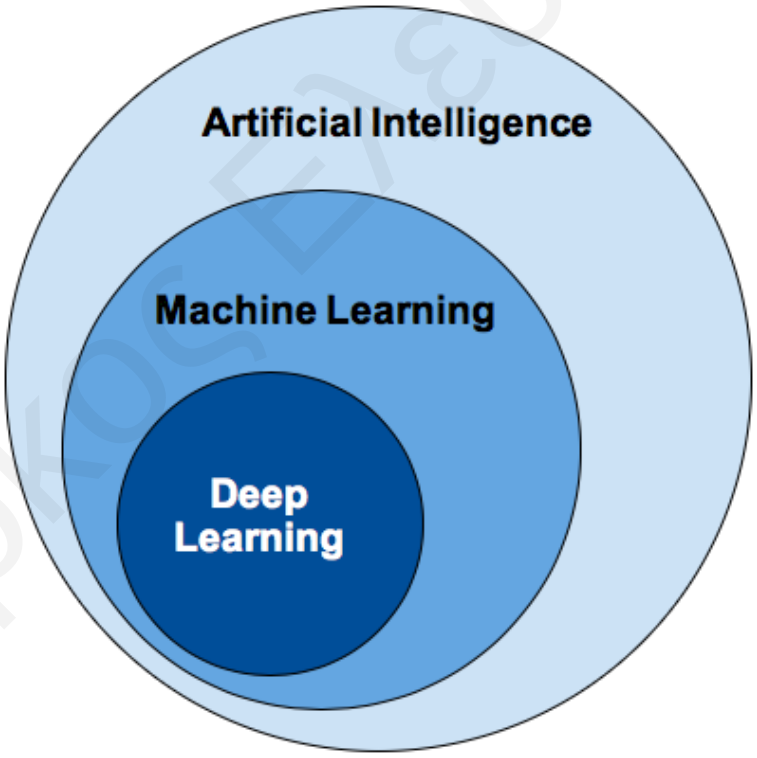
Κατόπιν, διαγράφεται εναλλακτικός τρόπος εφαρμογής του YOLO συνδυάζοντάς το με ιπτάμενα ρομπότ (drone). Τα δύο ρομποτικά συστήματα μπορούν είτε να συνυπάρξουν σε μία εφαρμογή αλληλοσυμπληρώνοντας το ένα το άλλο, ή να λειτουργήσει το καθένα αυτόνομα. Το Drone που χρησιμοποιήθηκε στην εφαρμογή είναι το MAVIC 2.

Κεφάλαιο 2

2 ΥΠΟΒΑΘΡΟ

2.1 Τεχνητή Νοημοσύνη

Το Deep Learning είναι παράρτημα του Machine Learning, το οποίο με την σειρά του αποτελεί παράρτημα της Τεχνητής Νοημοσύνης (AI). Η γραφική αναπαράσταση της πιο πάνω σχέσης φαίνεται στην Εικόνα 2-1.



Εικόνα 2-1: Συσχέτιση Deep Learning, Machine Learning και AI [40].

Ο κύριος στόχος της τεχνητής νοημοσύνης, στην οποία συμπεριλαμβάνεται το Deep Learning, είναι η ανάπτυξη συνόλου αλγορίθμων και τεχνικών, ώστε να μπορούν να χρησιμοποιηθούν για την επίλυση μίας ομάδας προβλημάτων. Αυτά αφορούν

ζητήματα, τα οποία ο άνθρωπος επιλύει διαισθητικά και σχεδόν αυτόματα, παρόλα αυτά όμως η επίλυσή τους αποτελεί τεράστια πρόκληση για ένα υπολογιστή.

Ένα παράδειγμα που ταιριάζει απόλυτα με τα όσα προ λίγου ειπώθηκαν είναι η κατανόηση του περιεχομένου μίας εικόνας. Δίνεται δηλαδή μία εικόνα στην οποία παρουσιάζεται κάποιο στοιχείο ή αναπαρίσταται ένα γεγονός, ακολούθως παρατηρείται και συγκρίνεται το ποσοστό αντίληψης και ο χρόνος που απαιτείται για κατανόηση τόσο από τον άνθρωπο όσο και από τον υπολογιστή. Σαφώς και ο άνθρωπος μπορεί να διεκπεραιώσει την εργασία με ελάχιστη έως καθόλου προσπάθεια σε εξαιρετικά σύντομο χρονικό διάστημα. Αντίθετα, ο υπολογιστής παρουσιάζει σημαντικές δυσκολίες προκειμένου να υλοποιήσει τον στόχο.

Σκοπός λοιπόν της Τεχνητής Νοημοσύνης είναι η σχεδίαση και υλοποίηση αλγορίθμων ώστε να μπορούν να μιμηθούν τις ανθρώπινες γνωστικές ικανότητες για την υιοθέτηση χαρακτηριστικών που αποδίδονται συνήθως στην ανθρώπινη συμπεριφορά. Μερικές από τις βασικές ικανότητες για τις οποίες θα τύχει εκπαίδευσης ο υπολογιστής είναι η μάθηση, κατανόηση και επίλυση προβλημάτων.

2.2 Machine Learning

Καθώς η Τεχνητή Νοημοσύνη εστιάζει στην ενσωμάτωση ενός συνόλου εργασιών που απαρτίζεται από τις ικανότητες στις οποίες έγινε αναφορά πιο πάνω, το τμήμα της Μηχανικής Μάθησης επικεντρώνεται στην αναγνώριση προτύπων και τη μάθηση μέσω δεδομένων.

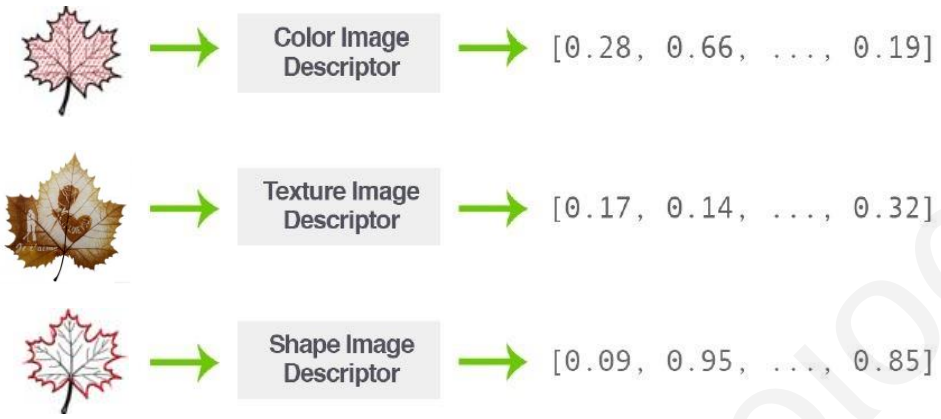
Μία κατηγορία αλγορίθμων Μηχανικής Μάθησης που χρίζει αναφοράς είναι τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs), το όνομα της οποίας προήλθε από την δομή και τη λειτουργία του ανθρώπινου εγκεφάλου απ’ όπου και εμπνεύστηκε ο τρόπος ανάπτυξής τους. Πρόκειται για δίκτυα τα οποία μαθαίνουν από δεδομένα και ειδικεύονται στην αναγνώριση προτύπων. Αποτελούνται κατά κανόνα από διασυνδεδεμένα υπολογιστικά στοιχεία με ικανότητα ανταπόκρισης σε τυχόν ερεθίσματα που θα βρεθούν στην είσοδό τους, ενώ παράλληλα μαθαίνουν να προσαρμόζονται στο εκάστοτε περιβάλλον.

2.3 Deep Learning

Το Deep Learning ανήκει στην οικογένεια των Τεχνητών Νευρωνικών Δικτύων και στις πλείστες των περιπτώσεων οι δύο όροι συναντώνται εναλλακτικά. Με μία πιο προσεκτική ματιά, διαπιστώνεται ότι ο τομέας του Deep Learning υφίσταται πέραν των 60 χρόνων και συναντάται με διαφορετικά ονόματα και ενσαρκώσεις. Ανάλογα με την έρευνα, το διαθέσιμο υλικό και τα σύνολα δεδομένων ο αλγόριθμος διαφέρει αντίστοιχα.

Σε μια προσπάθεια εμβάθυνσης και ταξινόμησης των αλγορίθμων μηχανικής μάθησης, διαπιστώνεται ότι χωρίζονται σε τρεις μεγάλες υποομάδες. Την μάθηση με επίβλεψη (supervised learning), χωρίς επίβλεψη και μάθηση με μερική επίβλεψη. Όσον αφορά την μάθηση με επίβλεψη – στην οποία βασίζεται το YOLO – δίνεται ένα σύνολο εισόδων μαζί με ένα στόχο εξόδου. Ακολούθως, ο αλγόριθμος επιχειρεί να προσαρμοστεί στα μοτίβα τα οποία με την σειρά τους χρησιμοποιούνται για αυτόματη αντιστοίχιση μεταξύ δεδομένων εισόδου ως προς τον ορθό στόχο στην έξοδο. Αντίθετα, κατά την μάθηση χωρίς οποιαδήποτε επίβλεψη ο αλγόριθμος προσπαθεί να ανακαλύψει διακριτικά χαρακτηριστικά χωρίς όμως να υφίσταται η όποια ένδειξη σχετικά με τις εισόδους. Προσπαθεί με άλλα λόγια να ομαδοποιήσει παρόμοιες καταστάσεις εισόδου και εξόδου χωρίς να γνωρίζει το πως ακριβώς συνδέονται μεταξύ τους. Στα πλαίσια λοιπόν του Machine Learning αναφορικά με την κατηγοριοποίηση εικόνων, βασικό στόχο αποτελεί η λήψη των συνόλων εικόνων και παράλληλα ο εντοπισμός μοτίβων τα οποία αργότερα θα χρησιμοποιηθούν ώστε να καταστεί εφικτή η ταξινόμηση των εικόνων συγκριτικά με το περιεχόμενό τους.

Αξίζει να γίνει αναφορά ότι πρωτύτερα λαμβάνονταν υπόψη χειροποίητα χαρακτηριστικά προκειμένου να ποσοτικοποιηθεί το περιεχόμενο μίας εικόνας. Για κάθε εικόνα δηλαδή, εξαγόταν ένα σύνολο διανυσμάτων, μία λίστα αριθμών με άλλα λόγια. Το κάθε διάνυσμα αφορούσε κάποιο από τα χαρακτηριστικά της εικόνας περιγράφοντας το σχήμα, το χρώμα και την υφή της. Κάτι παρόμοιο σκιαγραφείται και στην Εικόνα 2-2. Ο εν λόγω αλγόριθμος αποσκοπεί στην κατηγοριοποίηση διαφόρων ειδών φύλλων. Ο υπολογιστής εν τούτοις, εκπαιδεύεται στην ταξινόμηση των εικόνων/φύλλων, διαχωρίζοντάς τα λόγω χαρακτηριστικών που διαφέρουν από κατηγορία σε κατηγορία. Ορισμένα από τα εν λόγω χαρακτηριστικά είναι οι διαστάσεις (μήκος και πλάτος) του φύλλου, το χρώμα και το περίγραμμά του. Όσο πιο σύνθετο ή έντονο είναι το εκάστοτε χαρακτηριστικό, τόσο περισσότεροι αριθμοί θα περιέχονται στο διάνυσμα.



Εικόνα 2-2: Σύνολο διανυσμάτων που περιγράφουν χαρακτηριστικά [41].

Από την άλλη, το Deep Learning ακολουθεί μια διαφορετική προσέγγιση προκειμένου να φτάσει στην ταξινόμηση. Για την εξαγωγή χαρακτηριστικών από μία εικόνα δεν προκαθορίζονται χειροκίνητα ένα σύνολο κανόνων και αλγορίθμων. Αντ' αυτού, λαμβάνει χώρα μία διαδικασία εκπαίδευσης κατά τη διάρκεια της οποίας τα χαρακτηριστικά αυτά μαθαίνονται αυτόματα. Αφού λοιπόν το Machine Learning στοχεύει στη μάθηση από εμπειρία σε ήδη υπάρχουσα λύση του ιδίου προβλήματος που προσπαθούν να επιλύσουν, με τη σειρά του το Deep Learning εκτελεί μία προσπάθεια κατανόησης του προβλήματος μέσω μίας ιεραρχίας εννοιών (hierarchical learning).

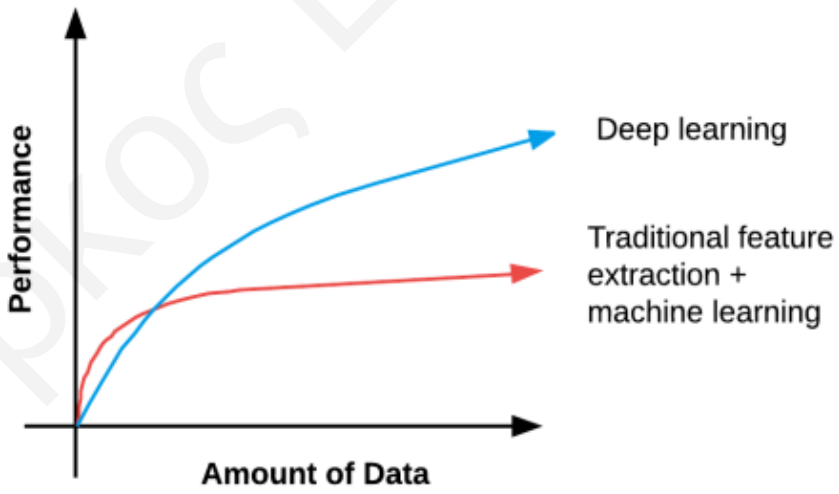
Ο όρος ιεραρχία αφήνει περιθώρια εκτενούς ανάλυσης, ωστόσο για την ώρα αρκεί και μόνο να ειπωθεί ότι κάθε έννοια βασίζεται σε άλλες. Στα πιο χαμηλά επίπεδα του δικτύου, κάθε έννοια κωδικοποιεί βασική αναπαράσταση του ζητήματος προς επίλυση. Αντίθετα, καθώς το επίπεδο των στρωμάτων ανεβαίνει, οι έννοιες αναπτύσσονται σε όλο και πιο αφηρημένες χρησιμοποιώντας τα χαμηλότερα στρώματα προκειμένου αυτό να καταστεί εφικτό.

2.4 Convolutional Neural Network (CNN)

Τα Συνεπτυγμένα Νευρωνικά Δίκτυα χρησιμοποιούνται ευρέως σε εργασίες αναγνώρισης εικόνων και βίντεο. Σχεδιάζονται ώστε να συμβαδίζουν με την αυτόματη εκμάθηση και εξαγωγή σχετικών χαρακτηριστικών από εικόνες εφαρμόζοντας συνέλιξη, συγκέντρωση και συναρτήσεις ενεργοποίησης.

Η συνέλιξη περιλαμβάνει την ολίσθηση ενός συνόλου φίλτρων πάνω σε μία εικόνα προκειμένου να εξαχθούν διαφορετικά χαρακτηριστικά. Το κάθε φίλτρο παράγει ένα νέο «χάρτη χαρακτηριστικών» ο οποίος αναδεικνύει συγκεκριμένη πτυχή της εικόνας όπως ακμές, υφές και γωνίες. Τα επίπεδα συγκέντρωσης εκτελούν δειγματοληψία στην έξοδο των επιπέδων συνέλιξης, μειώνοντας την διαστασιολόγηση των δεδομένων και ευκολύνοντας με αυτό τον τρόπο την επεξεργασία τους. Τέλος, οι συναρτήσεις ενεργοποίησης χρησιμοποιούνται για την εισαγωγή μη γραμμικότητας στο δίκτυο επιτρέποντάς του να μαθαίνει πιο σύνθετα μοτίβα δεδομένων.

Καθορίζοντας λοιπόν την ποσότητα των συνελικτικών στρωμάτων, υφίσταται εμμέσως το ποσό «Deep» είναι το Deep Learning. Σύμφωνα και με τον Dr. Adrian Rosebrock [18], οποιοδήποτε δίκτυο με περισσότερα από δύο hidden layers μπορεί να θεωρηθεί «Deep». Καθώς το βάθος του δικτύου αυξάνεται, παρατηρείται αύξηση και στην ακρίβεια ταξινόμησης, κάτι που διαφέρει από τους παραδοσιακούς αλγόριθμους Μηχανικής Μάθησης. Η εξάρτηση των δύο ποσοτήτων διαγράφεται στην Εικόνα 2-3. Με την αύξηση του όγκου δεδομένων του Deep Learning, η ακρίβεια ξεπερνά σημαντικά τις παραδοσιακές προσεγγίσεις.



Εικόνα 2-3: Σύγκριση ακρίβειας μεταξύ Deep Learning, παραδοσιακών μεθόδων και Μηχανικής Μάθησης [19].

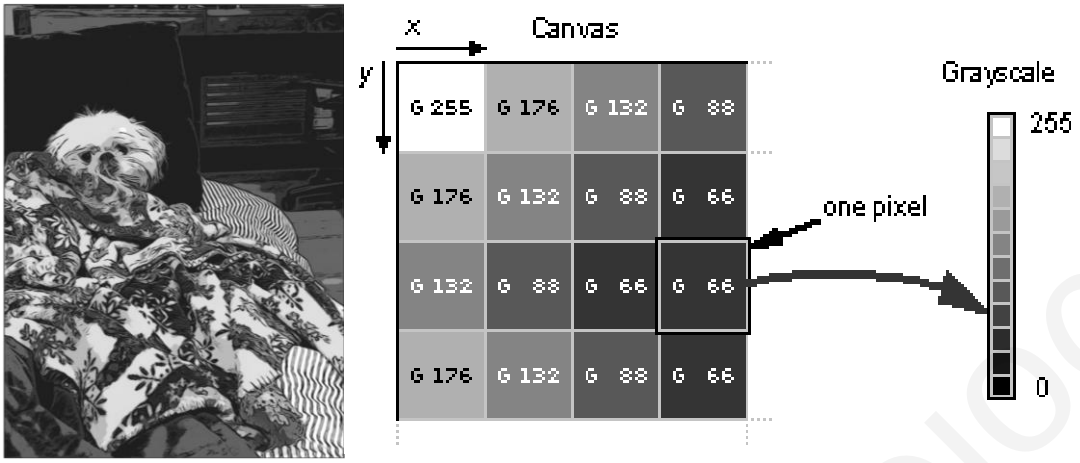
2.5 Βασικές Αρχές Εικόνας

Για να καταφέρει ένας άνθρωπος να ταξινομήσει εικόνες με τρόπο που να μπορεί να προσεγγιστεί από υπολογιστή, πρέπει αρχικά να αντιληφθεί τι είναι μία εικόνα και τι δεδομένα υπάρχουν πίσω από το προφανές. Η γνώση, των βασικών τουλάχιστο, εικονοστοιχείων θα επιτρέψει αν μη τι άλλο μία συμβατή ταξινόμηση μεταξύ ανθρώπου και υπολογιστή.

2.5.1 Τα Δομικά Στοιχεία της Εικόνας

Το ελάχιστο στοιχείο στο οποίο μπορεί να διαιρεθεί μία εικόνα είναι το pixel. Αποκαλούνται ως τα ακατέργαστα δομικά στοιχεία της εικόνας αφού κάθε εικόνα δημιουργείται από ένα σύνολο από pixels. Αυτό μπορεί να το αντιληφθεί κανείς ευκολότερα θεωρώντας μία εικόνα ως ένα πλέγμα. Κάθε κελί του πλέγματος αντιστοιχεί σε ένα pixel, το οποίο αναπαριστά την ένταση του φωτός ή αλλιώς το χρώμα που εμφανίζεται σε ένα δεδομένο της εικόνας.

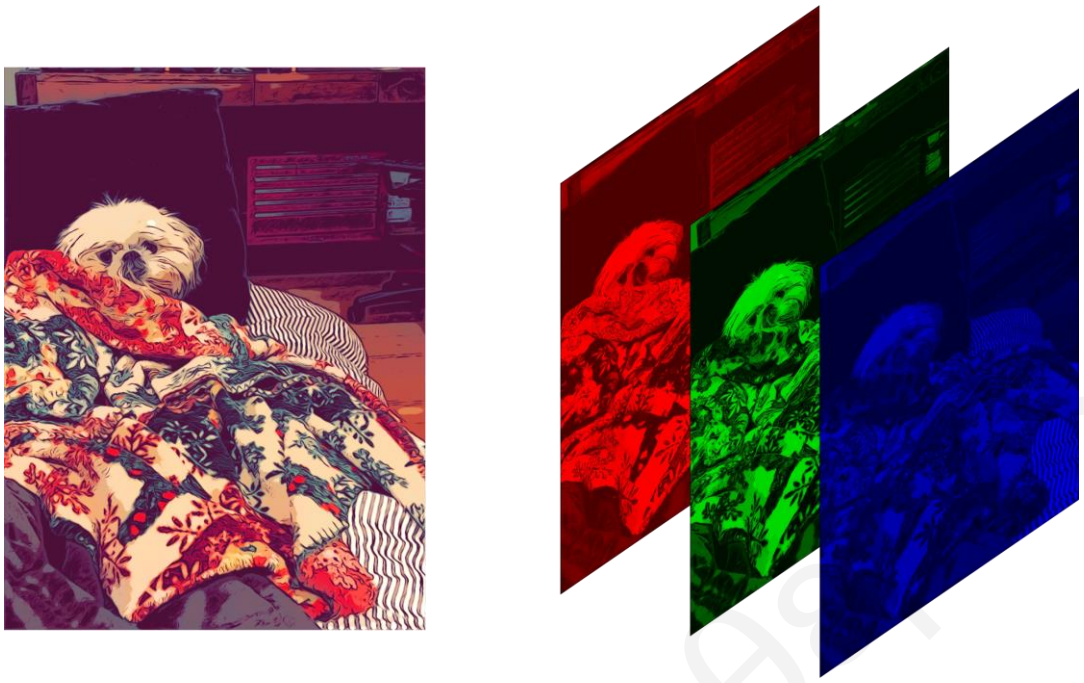
Ακολουθώντας την θεωρία του πλέγματος, το μήκος και πλάτος της εικόνας καθορίζονται από την ποσότητα των pixel, το ίδιο και το εμβαδόν. Στις πλείστες των περιπτώσεων τα κελιά αναπαρίστανται είτε σε κλίμακα του γκρι, ή με χρώμα. Σύμφωνα με την πρώτη περίπτωση, το κάθε κελί μπορεί να αντιμετωπιστεί ως μία κλιμακωτή τιμή μεταξύ 0 και 255. Στην εν λόγω κλίμακα το 255 αντιστοιχεί στο λευκό. Μεταξύ των δύο τιμών περιέχονται άλλες 254 αποχρώσεις του γκρι και αντίστοιχα όσο πιο κοντά στο 0 βρίσκονται, τόσο πιο σκούρες θα είναι. Με μία πιο προσεκτική ματιά στην Εικόνα 2-4, η εικόνα στα αριστερά χωρίζεται σε στοιχεία, καθένα από τα οποία αντιπροσωπεύει μία τιμή. Ενδεικτικό είναι το σχεδιάγραμμα στα δεξιά της Εικόνας 2-4, το οποίο περιγράφει μία μικρή περιοχή της εικόνας. Πολλές φορές η αποτύπωση της εικόνας σε πίνακα τιμών γίνεται μέσω ποσοστού. Δηλαδή στο κάθε στοιχείο αναγράφεται μία τιμή η οποία αντιστοιχεί στο επί τοις εκατό ποσοστό μεταξύ του εύρους [0,255].



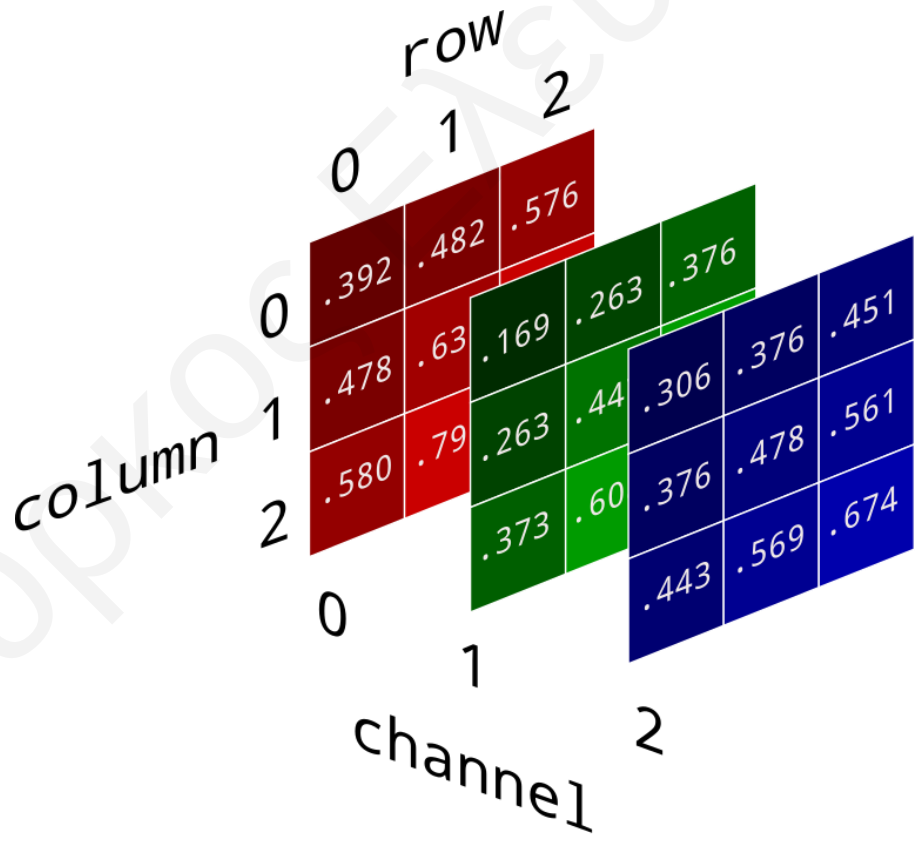
Εικόνα 2-4: Αποτύπωση εικόνας της κλίμακας γκρι [20].

Στα πλαίσια του Deep Learning, τα έγχρωμα pixels αναπαρίστανται στον χρωματικό χώρο RGB (Red Green Blue). Σε αντίθεση με την κλίμακα γκρι, το κάθε pixel, αντιπροσωπεύεται πλήρως από μία λίστα τριών τιμών – μία τιμή για κάθε χρώμα. Πιο απλά, προκειμένου να προσδιοριστεί ένα χρώμα στο χρωματικό μοντέλο RGB, αρκεί μόνο να οριστεί η ποσότητα του κόκκινου, πράσινου και μπλε για το εκάστοτε μεμονωμένο στοιχείο. Οι τιμές περιορίζονται όπως και στην κλίμακα γκρι, στο εύρος 0 με 255, ώστε να αναπαρασταθεί η ένταση με το 255 να αντιστοιχεί στην πλήρη αποτύπωση του χρώματος. Με τον συνδυασμό των τριών στρωμάτων, προκύπτει έγχρωμη η εικόνα.

Παράδειγμα της διαδικασίας συνδυασμού των στρωμάτων παρουσιάζεται στην Εικόνα 2-5, ενώ στην Εικόνα 2-6, συγκεκριμένη περιοχή επιλέγεται και αναπαρίσταται αριθμητικά για το κάθε στρώμα. Ο προσδιορισμός συγκεκριμένου στοιχείου σε οποιοδήποτε στρώμα επιτυγχάνεται με τον καθορισμό της σειράς και της στήλης στην οποία βρίσκεται, καθώς επίσης και στο στρώμα που αντιστοιχεί. Προγραμματιστικά η Εικόνα 2-6, ορίζεται ως πολυδιάστατος πίνακας 3D NumPy με ύψος, πλάτος και βάθος.



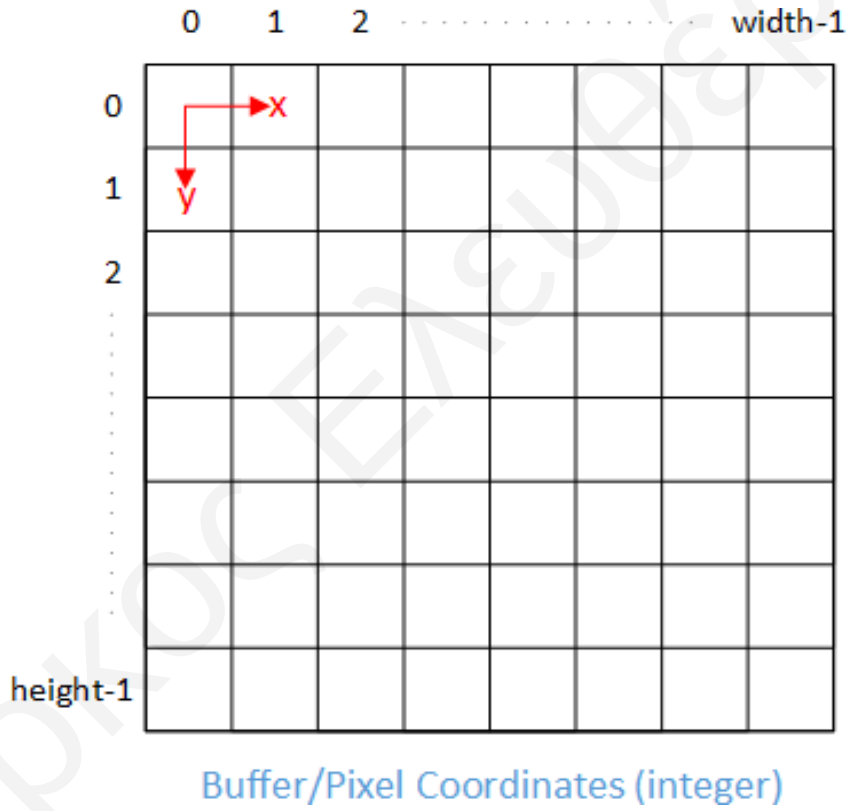
Εικόνα 2-5: Στρώματα RGB εικόνας [20].



Εικόνα 2-6: Αριθμητική απεικόνιση στρωμάτων RGB εικόνας [20].

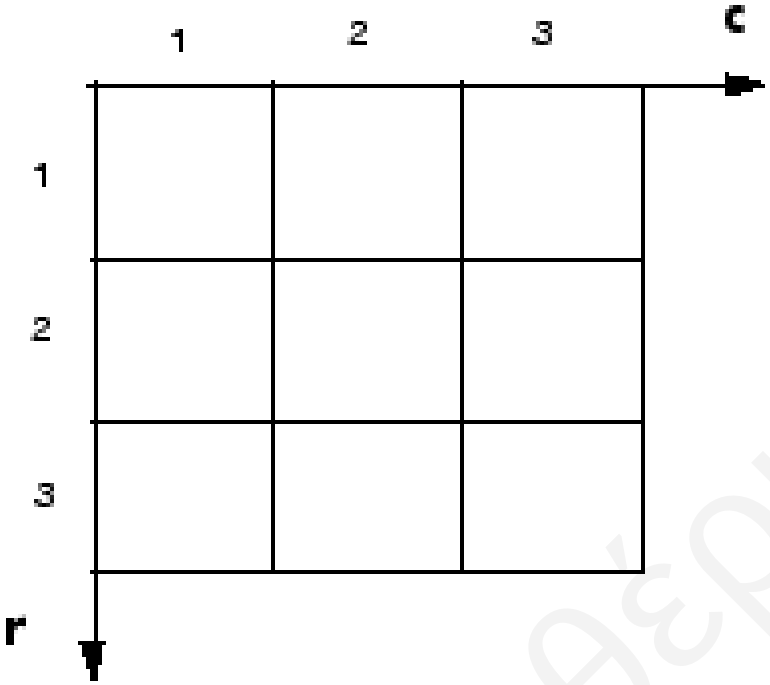
2.5.2Σύστημα Συντεταγμένων Εικόνων

Όπως έχει ήδη ειπωθεί, μία εικόνα αναπαρίσταται ως ένα πλέγμα από pixels. Υιοθετώντας το διάγραμμα στην Εικόνα 2-7, η αρχή των αξόνων αντιστοιχεί στην άνω αριστερή γωνία της εικόνας. Οι τιμές x, y αυξάνονται προς τα δεξιά και προς τα κάτω. Αυτό θα διαφανεί καλύτερα μέσω παραδειγμάτων στα μετέπειτα κεφάλαια. Προς το παρόν, είναι σημαντικό να επισημανθεί η διαφορά μεταξύ του συστήματος συντεταγμένων εικόνων σε Python και άλλες γλώσσες προγραμματισμού, έτσι ώστε να μην προκληθούν λάθη λόγω σύγχυσης.



Εικόνα 2-7: Σύστημα Συντεταγμένων εικόνων σε Python [21].

Σημαντική διαφορά λοιπόν, είναι ότι η Python είναι zero indexed γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η αρίθμηση αρχίζει πάντοτε από το μηδέν. Στο περιβάλλον της MatLab από την άλλη, η αρίθμηση ξεκινά από την μονάδα. Απ' εκεί και πέρα, δεν παρουσιάζονται μεγάλες διαφορές μεταξύ τους. Όσον αφορά την διαστασιολόγηση, ο αριθμός των γραμμών του πλέγματος αντιστοιχεί στο ύψος της εικόνας, ενώ ο αριθμός των στηλών στο πλάτος της.



Εικόνα 2-8: Σύστημα Συντεταγμένων εικόνων σε MatLab [22].

2.5.3Aspect Ratio

Υπάρχουν περιπτώσεις που η αλλαγή μεγέθους μίας εικόνας καθίσταται αναγκαία ώστε να διεκπεραιωθεί κάποια λειτουργία. Για το resize μίας εικόνας λοιπόν, υπάρχει τυποποιημένη διαδικασία η οποία εξασφαλίζει την ακεραιότητα του περιεχομένου της εικόνας. Υπάρχουν πολλές παγίδες στις οποίες μπορεί να πέσει κανείς στην προσπάθεια διαμόρφωσης του σχήματος της εικόνας. Παράλληλα όμως υπάρχει και ένα σύνολο κανόνων που αρκεί να ακολουθηθούν πιστά για να μην προκύψουν εσφαλμένα αποτελέσματα.

Αρχικά, σχετικά με την αλλαγή μεγέθους μίας εικόνας, πρέπει οπωσδήποτε να τηρείται η αναλογία στις διαστάσεις της. Αυτό σημαίνει ότι μία εικόνα μπορεί να διαμορφωθεί ανάλογα, αρκεί να παραμείνει ίδιος, ή τουλάχιστον να μην διαφέρει σημαντικά, ο λόγος του πλάτους προς το ύψος της. Σε περίπτωση που αυτό δεν ληφθεί σοβαρά υπόψη η εικόνα που θα προκύψει πιθανό να φαίνεται παραμορφωμένη ή συμπιεσμένη. Παραταύτα, εάν δεν είναι εφικτό να διαμορφωθεί το μέγεθος της εικόνας διατηρώντας σταθερό το aspect ratio, υπάρχει και η επιλογή αποκοπής. Σε τέτοιες περιπτώσεις συνιστάται ιδιαίτερη προσοχή, ώστε να μην χαθούν σημαντικές πληροφορίες από το περιεχόμενο της εικόνας.

Παράλληλα, στα περισσότερα μοντέλα χρειάζεται να προκαθοριστούν, μέσω εντολής, οι διαστάσεις. Αυτό σημαίνει ότι όλες οι εικόνες στο σύνολο δεδομένων

πρέπει να έχουν τις συγκεκριμένες διαστάσεις. Υπάρχουν έτοιμα σύνολα δεδομένων τα οποία περιέχουν εικόνες ιδίων διαστάσεων. Καλό θα ήταν όμως οι διαστάσεις των εικόνων που θα χρησιμοποιηθούν σε μία διαδικασία εκπαίδευσης (Training), να συνάδουν με αυτές της εικόνας που θα γίνει το Detection. Η συγκεκριμένη λεπτομέρεια μειώνει τα ποσοστά απόκλισης αυξάνοντας έτσι την ακρίβεια των αποτελεσμάτων.

Τέλος, θα γίνει αναφορά σε ένα ζήτημα το οποίο παρόλο που μπορεί να οδηγήσει σε εντελώς εσφαλμένα αποτελέσματα, ωστόσο θα επιτρέψει στο μοντέλο να ολοκληρώσει τη διαδικασία χωρίς να αφήσει να νοηθεί πως κάτι δεν πάει καλά. Αυτό θα το διαπιστώσει αργότερα ο χειριστής του μοντέλου όταν κληθεί να αξιολογήσει τα αποτελέσματα. Ο λόγος για την αλλαγή των διαστάσεων στις εικόνες του συνόλου δεδομένων μετά την προσθήκη ετικετών. Αναλυτική περιγραφή της διαδικασίας θα γίνει στα επόμενα κεφάλαια, αυτό όμως που χρειάζεται να ειπωθεί σε αυτό το σημείο είναι ότι αφού καθοριστούν οι ετικέτες για ένα σύνολο εικόνων, με την αλλαγή των διαστάσεων παραμένουν ως έχουν. Με τον τρόπο αυτό αλλοιώνεται η διαδικασία της εκπαίδευσης του μοντέλου. Περεταίρω ανάλυση θα ακολουθήσει στο τμήμα της εφαρμογής ετικετών στις εικόνες στο Κεφάλαιο 3.1.

2.6 Κατηγοριοποίηση Εικόνων

Ταξινόμηση των εικόνων, αποκαλείται η ανάθεση μίας ετικέτας σε μία εικόνα από προκαθορισμένο σύνολο κατηγοριών. Πρακτικά αυτό αφήνει να νοηθεί ότι αναλύεται μία εικόνα και επιστρέφεται μία ετικέτα η οποία κατηγοριοποιεί την εικόνα. Η διαδικασία ακολουθεί πιστά την πορεία: είσοδος – επεξεργασία – έξοδος. Η ετικέτα που αφορά την εικόνα, σχετίζεται πάντοτε με ένα προκαθορισμένο σύνολο πιθανών κατηγοριών.

Στον αντίποδα, ένα σύνολο δεδομένων αποτελεί μία συλλογή σημείων, στόχος της οποίας είναι η εφαρμογή αλγορίθμου Machine / Deep Learning, ούτως ώστε να εντοπιστούν τα υποκείμενα μοτίβα στο σύνολο δεδομένων. Επιτρέπεται έτσι η ορθή ταξινόμηση των σημείων στο σύνολο δεδομένων. Στα σύνολα δεδομένων που χρησιμοποιούνται στην ανίχνευση αντικειμένων, κάθε εικόνα ή frame αντιστοιχεί σε ένα σημείο του συνόλου δεδομένων.

Το ζήτημα που προκύπτει ως προς την αντίληψη του περιεχομένου ονομάζεται σημασιολογικό χάσμα (Semantic Gap). Αυτό περιγράφεται ως η διαφορά μεταξύ του τρόπου με τον οποίο ο άνθρωπος αντιλαμβάνεται το περιεχόμενο μίας εικόνας, σε σύγκριση με τον τρόπο που αναπαρίσταται μία εικόνα ώστε ένας υπολογιστής να

βρίσκεται σε θέση να αντιληφθεί τη διαδικασία. Παρόλο που ο άνθρωπος χρειάζεται μόλις μερικά δευτερόλεπτα προκειμένου να αντιληφθεί πλήρως το περιεχόμενο μίας εικόνας, αντίθετα για τον υπολογιστή χρειάζεται μία γραμμή στην οποία να κωδικοποιούνται όλες οι πληροφορίες με τρόπο που να μπορεί να τις κατανοήσει.

Αυτό επιτυγχάνεται με την εφαρμογή της εξαγωγής χαρακτηριστικών για την ποσοτικοποίηση του περιεχομένου μίας εικόνας. Σε αυτά τα χαρακτηριστικά περιλαμβάνονται: η χωρική διάταξη, η υφή και το χρώμα. Μέσω της διαδικασίας που ειπώθηκε προηγουμένως, η εξαγωγή χαρακτηριστικών είναι η διαδικασία λήψης μίας εικόνας στην είσοδο του συστήματος, επεξεργασία μέσω εφαρμογής αλγορίθμου και λήψη, στην έξοδο του συστήματος, ενός διανύσματος χαρακτηριστικών το οποίο ποσοτικοποιεί την εικόνα.

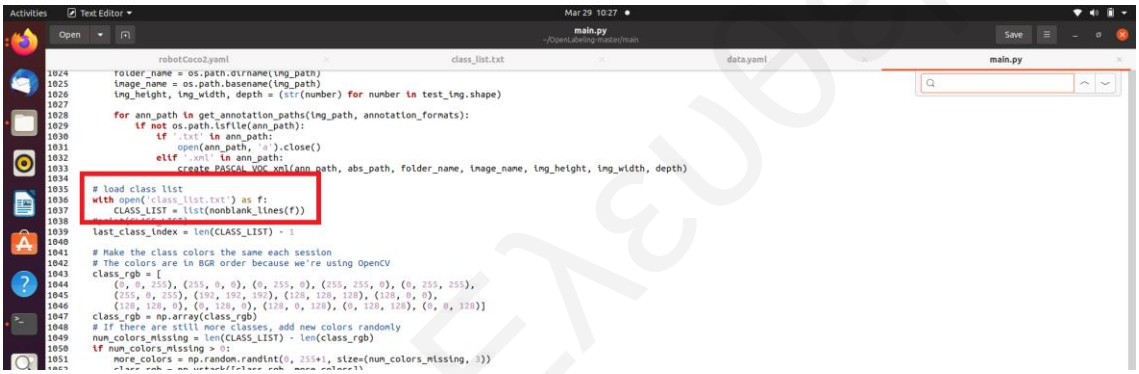
Απ’ εκεί και πέρα ο αλγόριθμος που δημιουργείται διακρίνει την διαφορά μεταξύ των εικόνων. Αντί λοιπόν να προσπαθεί να κατασκευάσει ένα σύστημα βασισμένο σε κανόνες με σκοπό να περιγράψει το περιεχόμενο της κάθε εικόνας, προσπαθεί να ακολουθήσει μία προσέγγιση η οποία βασίζεται στα δεδομένα, μέσω παραδειγμάτων για το πως μοιάζει η κάθε κατηγορία. Στη συνέχεια ο αλγόριθμος διδάσκεται ώστε να αναγνωρίζει την διαφορά μεταξύ των κλάσεων χρησιμοποιώντας τα εν λόγω παραδείγματα. Τα παραδείγματα αυτά ονομάζονται σύνολο δεδομένων εκπαίδευσης των σεσημασμένων εικόνων, όπου κάθε σημείο δεδομένων στο σύνολο αποτελείται από μία εικόνα και την ετικέτα που την συνοδεύει στην οποία περιέχεται η κλάση.

2.7 Διαμόρφωση Συνόλου Δεδομένων

Η δημιουργία ενός δικτύου Deep Learning βασίζεται στην ανάκτηση στοιχείων, τα οποία στη συνέχεια θα τοποθετηθούν στην κατάλληλη θέση ώστε να προκύψει το επιθυμητό αποτέλεσμα. Πρώτο λοιπόν στοιχείο είναι η συγκέντρωση του συνόλου δεδομένων το οποίο θα αξιοποιηθεί για την εκπαίδευση, επιβεβαίωση και έλεγχο του μοντέλου. Όπως έχει ειπωθεί, αποτελείται από τις εικόνες και τις ετικέτες τους. Οι ετικέτες μεταφέρουν δύο είδη πληροφορίας, την κατηγορία στην οποία ανήκει το αντικείμενο, καθώς επίσης και η τοποθεσία του στην εικόνα. Η καταχώρηση κατηγοριών χρίζει προσοχής αφού μέχρι το τελικό αποτέλεσμα συνδυάζεται μεγάλος αριθμός αρχείων τα οποία αναφέρονται σε αυτές.

2.7.1 Καταχώρηση Κλάσεων

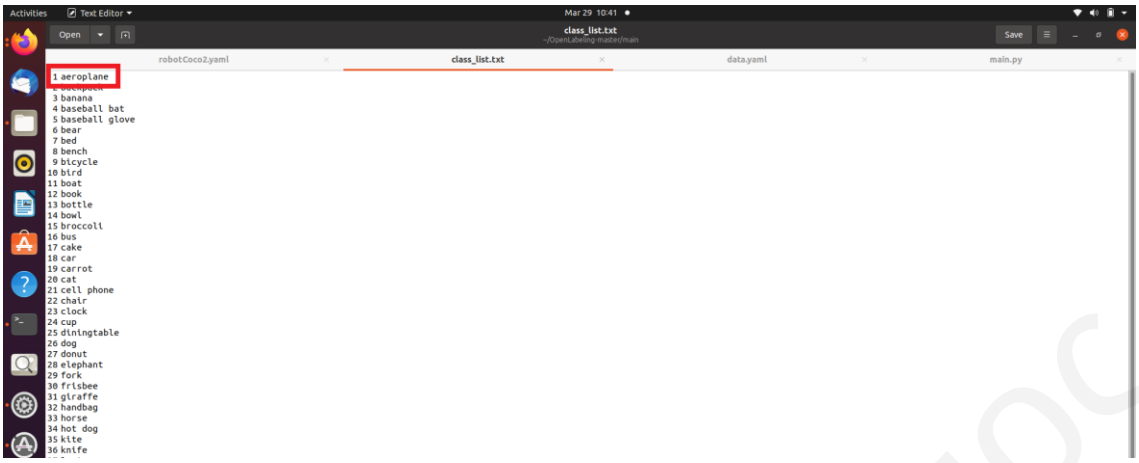
Οι ετικέτες που θα συνοδεύουν τις εικόνες, χρειάζεται να συμμορφώνονται με ένα πεπερασμένο σύνολο κατηγοριών. Για να υφίσταται μία κατηγορία και να μπορεί να επιλεχθεί κατά την διαδικασία της προσθήκης ετικετών, θα πρέπει να καταχωρηθεί στην λίστα με τις κατηγορίες. Το συγκεκριμένο .txt αρχείο καλείται μέσω του κώδικα main.py, όπως παρουσιάζεται στην Εικόνα 2-9, για να συμπεριληφθούν οι κατηγορίες που αναγράφονται. Οι κατηγορίες, πρέπει επίσης να συμβαδίζουν με το .yaml αρχείο το οποίο θα τρέχει ταυτόχρονα κατά την διάρκεια της εκπαίδευσης. Ανάλογα με τις κατηγορίες που απαιτεί η εκάστοτε εφαρμογή δημιουργείται ξεχωριστό αρχείο δεδομένων.



Εικόνα 2-9: Κώδικας main.py.

Το σημείο στο οποίο πρέπει να δοθεί έμφαση είναι ότι από την στιγμή που ολοκληρώνεται η διαδικασία με τις ετικέτες η κάθε κλάση αντιπροσωπεύεται με ένα αριθμό. Πλέον η κατηγορία aeroplane για παράδειγμα υφίσταται ως ο αριθμός 1. Κατά τη διάρκεια της εκπαίδευσης όποια κατηγορία εμφανίζεται πρώτη στο αρχείο δεδομένων, αυτή και μόνο αυτή θα περιγράφει την εικόνα κάθε φορά που εντοπίζεται αεροπλάνο. Εάν δηλαδή μπερδευτεί για κάποιο λόγο η διάταξη των κλάσεων με

Εικόνα 2-10: Αρχείο δεδομένων – data.yaml.



Εικόνα 2-11: Λίστα κατηγοριών – class_list.txt.

αποτέλεσμα στο αρχείο δεδομένων να αναγράφεται πρώτη η κατηγορία άνθρωπος, κάθε φορά που το μοντέλο θα εντοπίζει αεροπλάνο θα το ονομάζει άνθρωπο.

Εφόσον επιβεβαιωθεί ότι το αρχείο δεδομένων συμμορφώνεται με τη λίστα, τότε μπορεί να χρησιμοποιηθεί. Το data.yaml (βλ. Εικόνα 2-10) μπορεί να παρουσιάζει και περισσότερες κατηγορίες απ' ότι η λίστα, αρκεί όσες υπάρχουν στο αρχείο class_list.txt (βλ. Εικόνα 2-11), να αναγράφονται με την ίδια σειρά. Εννοείται ότι δεν πρόκειται να ανιχνευθούν αντικείμενα που ανήκουν στις επιπλέον κατηγορίες αφού όπως είναι λογικό δεν υπάρχει καμία ετικέτα που να τις περιλαμβάνει, έτσι το μοντέλο δεν θα τύχει εκπαίδευσης για το συγκεκριμένο περιεχόμενο.

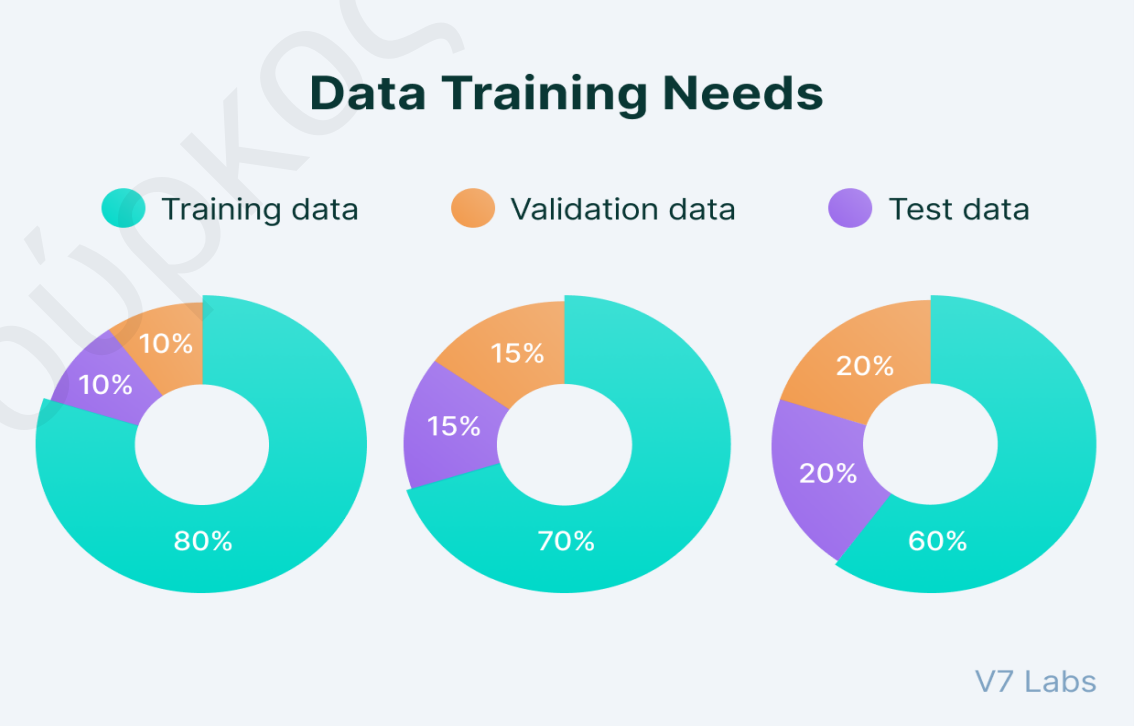
2.7.2Καταμερισμός εικόνων

Τα σύνολα δεδομένων ενδέχεται να διαφέρουν μεταξύ τους. Αυτό όμως που έχει σημασία είναι να τηρούνται οι στοιχειώδης τουλάχιστον προδιαγραφές ως προς την υποδομή τους. Στο κομμάτι αυτό παρουσιάζονται ορισμένοι τρόποι διαμόρφωσης των συνόλων ώστε η εκπαίδευση του υπολογιστή που θα ακολουθήσει να είναι όσο πιο αποτελεσματική γίνεται. Αυτό με την σειρά του θα έχει καθοριστικό ρόλο στην εξαγωγή αποτελεσμάτων.

Ένα κλειδί στην υλοποίηση ενός ολοκληρωμένου συνόλου δεδομένων είναι ο ισόποσος καταμερισμός των εικόνων βάσει του περιεχομένου τους. Αυτό προτρέπει σε ομοιόμορφο αριθμό εικόνων για την κάθε κλάση. Σε αντίθετη περίπτωση ο ταξινομητής (classifier) θα γίνει προκατειλημμένος σε υπερπροσαρμογή στις κατηγορίες που αντιπροσωπεύονται σε μεγαλύτερο βαθμό. Η ανισορροπία κλάσεων μπορεί να αντιμετωπιστεί, ωστόσο το καλύτερο για το μοντέλο είναι να μην προκύψει εξ αρχής.

Στην Εικόνα 2-10 όπου παρουσιάζεται το αρχείο δεδομένων, διακρίνεται στις πρώτες γραμμές η καταχώρηση μονοπατιών για τρεις διαφορετικές περιπτώσεις. Είναι τα σύνολα εικόνων και ετικετών που θα χρησιμοποιηθούν για εκπαίδευση, επιβεβαίωση και έλεγχο του μοντέλου. Παρόλο που τα τρία μονοπάτια μπορεί να είναι τα ίδια, να παραπέμπουν δηλαδή στις ίδιες εικόνες, αυτό δεν θα είχε και τόση σημασία. Ειδικά τα αποτελέσματα από επιβεβαίωση και ο έλεγχος πολύ πιθανόν να οδηγούσαν σε λανθασμένα συμπεράσματα. Ο έλεγχος για παράδειγμα της ακρίβειας του μοντέλου θα πρέπει να γίνει χρησιμοποιώντας εικόνες με τις οποίες ο υπολογιστής δεν έχει βρεθεί ξανά αντιμέτωπος εξασφαλίζοντας έτσι ότι τα υποσύνολα δεδομένων δεν επικαλύπτονται. Μόνο τότε τα αποτελέσματα που θα προκύψουν θα έχουν κάποια υπόσταση προκειμένου να γίνει αξιολόγηση της δουλειάς που έγινε σε εκπαίδευση και επιβεβαίωση.

Ως εκ τούτου, το αρχικό σύνολο δεδομένων χρειάζεται να χωριστεί σε τρία διαφορετικά μέρη. Το πρώτο και μεγαλύτερο τμήμα του συνόλου δεδομένων προορίζεται για την εκπαίδευση του μοντέλου καθοδηγώντας το να κατανοήσει τα μοτίβα που παρουσιάζει η κάθε κατηγορία. Όσον αφορά τα δύο εναπομείναντα συνηθίζεται να έχουν το ίδιο μέγεθος, ωστόσο η επιλογή εναπόκειται στον χειριστή ο οποίος θα διαμορφώσει τα υποσύνολα ανάλογα με την εφαρμογή. Όσον αφορά το δεύτερο υποσύνολο αξιοποιείται ώστε ο υπολογιστής να προβεί σε προβλέψεις για την



Εικόνα 2-12: Διαχωρισμός αρχικού Συνόλου Δεδομένων [23].

ταξινόμηση, και τη μετέπειτα διόρθωσή τους. Τέλος, η αξιολόγηση της μέχρι τώρα πορείας θα γίνει με την αξιοποίηση του τρίτου υποσυνόλου. Μερικά από τα πιο συνήθη μεγέθη διαχωρισμού του αρχικού συνόλου παρατίθενται στην Εικόνα 2-12.

2.8 Αξιολόγηση εικόνων

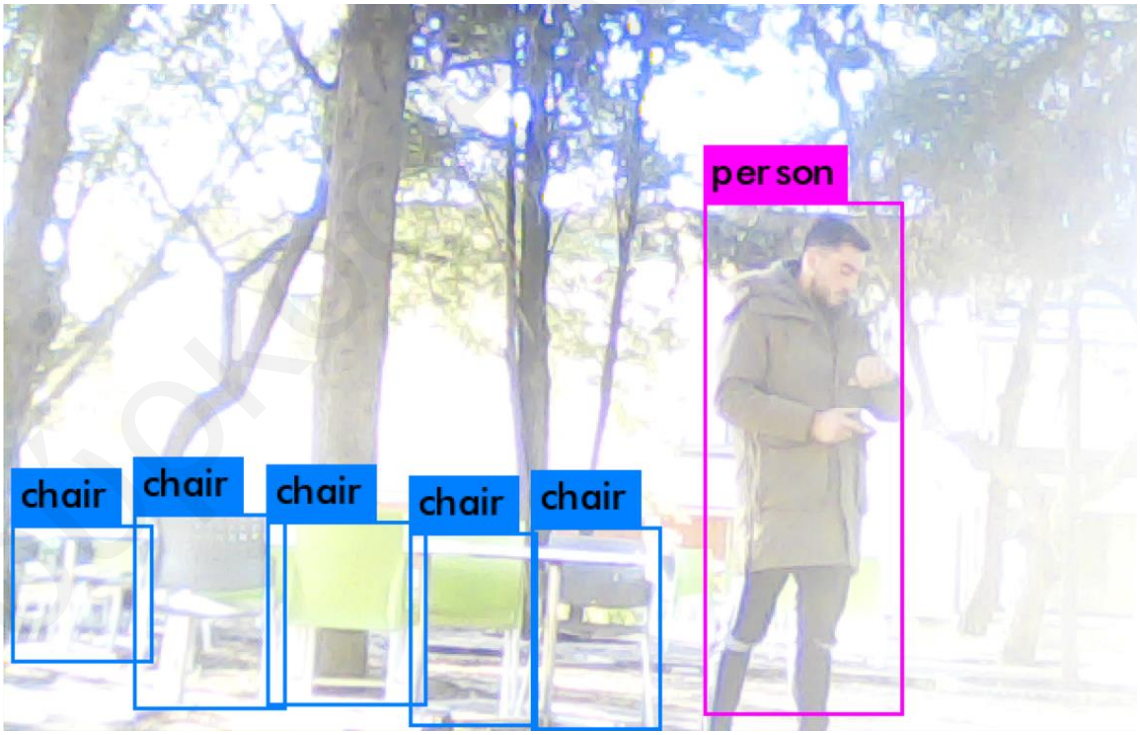
Εν κατακλείδι, ο τρόπος με τον οποίο εκπαιδεύεται το δίκτυο, είναι βασισμένος στα λάθη που θα υποπέσει κατά τη διάρκεια της επικύρωσης (Validation). Με τον εντοπισμό του λάθους, το μοντέλο μαθαίνει από αυτό και βελτιώνεται. Κατά την αξιολόγηση του εκπαιδευμένου πλέον δικτύου λαμβάνει χώρα σύγκριση μεταξύ των προβλέψεων και των ground-truth ετικετών που προκύπτει από το σύνολο των δοκιμών. Οι εν λόγω ετικέτες αντιπροσωπεύουν την πραγματική κατηγορία της εικόνας. Μέσω αυτών υπολογίζεται ο αριθμός των προβλέψεων που έγιναν σωστά από τον υπολογιστή, ενώ παράλληλα υπολογίζονται συγκεντρωτικές αναφορές όπως η ακρίβεια, ανάκληση και f-measure που επιτρέπουν την ποσοτικοποίηση της απόδοσης του δικτύου στο σύνολό του.

Υπάρχουν ωστόσο και περιπτώσεις όπου το μοντέλο παρουσιάζει μεγάλη ακρίβεια, προκύπτουν αποτελέσματα που επιτρέπουν την προώθησή του για σκοπούς εφαρμογής και ενώ όλα δημιουργούν ένα καλό οιωνό, το δίκτυο παρουσιάζεται τραγικά κατώτερο των περιστάσεων. Το φαινόμενο αυτό αποκαλείται generalization. Είναι η ικανότητα του δικτύου να γενικεύει και να προβλέπει ορθά την κλάση στην οποία ανήκει μία εικόνα που δεν υπάρχει στο αρχικό σύνολο δεδομένων. Σε παρόμοιες περιπτώσεις εξετάζεται το σύνολο των παραμέτρων ποσοτικοποίησης. Το υποσύνολο δεδομένων της εκπαίδευσης θα έπρεπε να αντικατοπτρίζει στο έπακρο παραδείγματα των παραμέτρων αυτών. Εάν η συνθήκη αυτή δεν υφίσταται, τότε ένας τρόπος αντιμετώπισης είναι η ποσοτική αύξηση του υποσυνόλου εκπαίδευσης.

Κεφάλαιο 3

3 YOU ONLY LOOK ONCE

Το YOLO είναι ένας αλγόριθμος ανίχνευσης αντικειμένων σε εικόνες και βίντεο. Αποτελεί αλγόριθμο Deep Learning που χρησιμοποιείται για να εντοπίζει τα αντικείμενα σε πραγματικό χρόνο. Βασίζεται σε Δίκτυα Νευρωνικών Συνελικτικών επιπέδων και χρησιμοποιεί μία μοναδική αρχιτεκτονική δικτύου που επιτρέπει τον εντοπισμό αντικειμένων με μεγάλη ακρίβεια και ταχύτητα. Το YOLO λειτουργεί εντοπίζοντας περιοχές ενδιαφέροντος στις εικόνες και αναθέτει σε κάθε περιοχή μία πιθανότητα και ένα κουτί που περιγράφει το αντικείμενο. Έχει επίσης τη δυνατότητα να ανιχνεύει πολλαπλά αντικείμενα ταυτόχρονα και να τα κατηγοριοποιεί σε διάφορες κλάσεις.



Εικόνα 3-1: Ταυτόχρονη ανίχνευση διαφόρων κατηγοριών.

Η ανίχνευση επιλέγηκε να γίνει με τον συγκεκριμένο αλγόριθμο λόγω σημαντικών δυνατοτήτων τις οποίες παρέχει. Αρχικά, λόγω ταχύτητας. Για όλες τις εφαρμογές ανίχνευσης σε ρομπότ ασφαλείας απαιτείται γρήγορη ανταπόκριση

προκειμένου να υπάρχει αμεσότητα στην αναγνώριση και ταυτοποίηση. Το YOLO μπορεί να ανιχνεύσει αντικείμενα σε εικόνες σε πραγματικό χρόνο, σε αντίθεση με άλλες μεθόδους που απαιτούν σημαντικά μεγαλύτερα χρονικά διαστήματα για την ανίχνευση. Παράλληλα παρουσιάζει υψηλή ακρίβεια στην ανίχνευση αντικειμένων, καθώς η μοναδική αρχιτεκτονική που χρησιμοποιεί παρέχει τη δυνατότητα εκπαίδευσης με μεγάλα σύνολα δεδομένων. Περεταίρω, μπορεί να ανιχνεύει πολλαπλά αντικείμενα σε μία εικόνα και να τα κατηγοριοποιεί σε διάφορες κλάσεις (βλ. Εικόνα 3-2). Τέλος, εκτός από το ότι παρέχει υψηλή ακρίβεια και αξιοπιστία, μπορεί να ανιχνεύσει αντικείμενα με διαφορετικά μεγέθη και αναλογίες στην ίδια εικόνα.

Open

robotCoco2.yaml
~/yolov7/robococo2

```yaml
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
# COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017) by Ultralytics
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov7
# └── datasets
#     └── coco128  ← downloads here (7 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, path/to/imgs2, ..]
train: /home/user/yolov7/robococo2/cocoRobot2/images/train2017   # train images (relative to 'path') 128 images
val: /home/user/yolov7/robococo2/cocoRobot2/images/train2017   # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
nc: 81 # number of classes
names: ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
        'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
        'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
        'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
        'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
        'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
        'hair drier', 'toothbrush','robot']  # class names


# Download script/URL (optional)
download: https://ultralytics.com/assets/coco128.zip
```

Εικόνα 3-2: Αρχείο διαμόρφωσης στο οποίο κατηγοριοποιούνται οι κλάσεις.

Για παράδειγμα, στην Εικόνα 3-3 παρουσιάζεται η ίδια φιγούρα σε διαφορετικές παραμορφώσεις. Ο υπολογιστής έχει πιθανόν εκπαιδευτεί στην αναγνώριση των πρώτων δύο στάσεων του σώματος, ωστόσο σε περίπτωση όπου εμφανιστεί η φιγούρα με διαμόρφωση όπως την τελευταία της ίδιας εικόνας, τότε απαιτείται για λόγους εφαρμογής να εντοπιστεί. Με το YOLO υπάρχει αυτή η δυνατότητα αφού όπως έχει ειπωθεί λειτουργεί με τον εντοπισμό περιοχών ενδιαφέροντος.

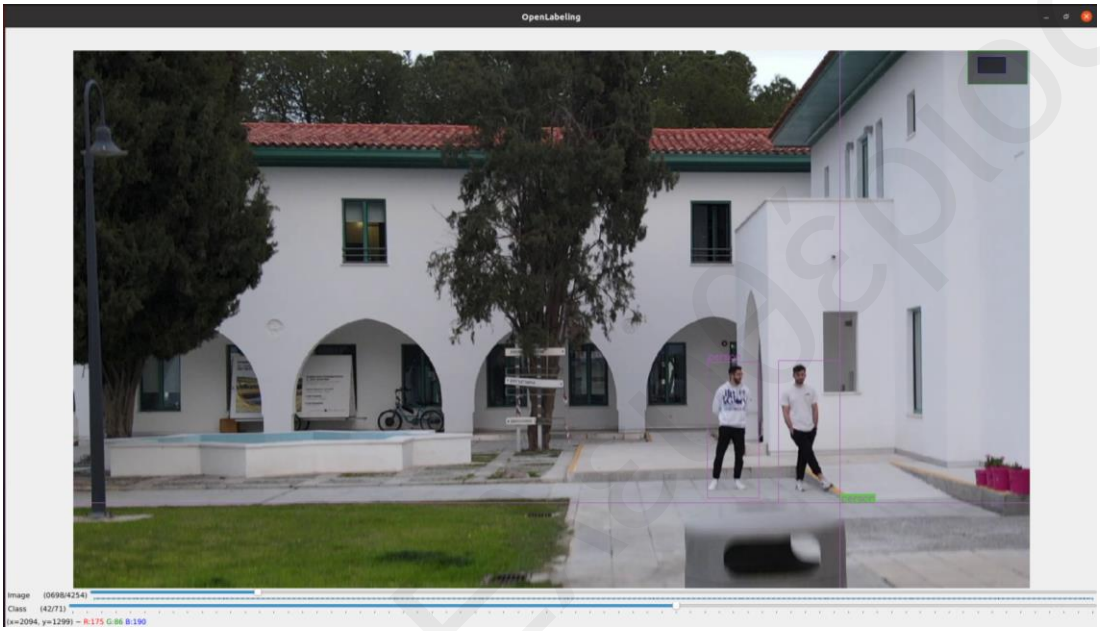
Εικόνα 3-3: Ίδιο μοντέλο αντικειμένου σε τρεις διαφορετικές διαμορφώσεις [18].

Πριν όμως να γίνει η αναγνώριση, όπως αναφέρθηκε και στο Κεφάλαιο 2, προηγείται μία μακροσκελής διαδικασία ώστε ο υπολογιστής να βρίσκεται σε θέση να αναγνωρίζει ανά πάσα στιγμή τα αντικείμενα για τα οποία έχει εκπαιδευτεί. Η διαδικασία κάνει αρχή από την εκπαίδευση του υπολογιστή. Μετά την εκπαίδευση ακολουθεί η επιβεβαίωση και τέλος γίνονται μερικές δοκιμές ούτως ώστε να διαπιστωθεί το κατά πόσο έγιναν τα προηγούμενα δύο βήματα με επιτυχία.

3.1 Κατηγοριοποίηση με Ετικέτες (Labelling)

Εφόσον ο αλγόριθμος βασίζεται στη μάθηση με επίβλεψη ώστε να κατηγοριοποιεί την κάθε αναγνώριση πρέπει αρχικά να εισαχθούν προς αναγνώριση ένα σύνολο από εικόνες το περιεχόμενο των οποίων θα αποτελείται από τα αντικείμενα ενδιαφέροντος. Προκειμένου να μπορέσει το YOLO να εκπαιδεύσει τον υπολογιστή εισάγεται μεγάλος αριθμό τυχαίων φωτογραφιών. Καλό θα ήταν να υπάρχουν ποικίλες φωτογραφίες όσον αφορά τη γωνία λήψης, τη φωτεινότητα, το φόντο και τη στάση του σώματος. Αυτό θα ήταν αρκετά βοηθητικό για τον υπολογιστή, να εκπαιδευτεί στην αναγνώριση, δηλαδή υπό δυσμενής συνθήκες, ούτως ώστε να είναι σε θέση να αναγνωρίσει το αντικείμενο ενδιαφέροντος σε οποιαδήποτε εικόνα. Μέσω ενός συνόλου από κώδικες που βρίσκονται συγκεντρωμένοι στο πακέτο Open Labelling στο GitHub [50], δίνεται η δυνατότητα επεξεργασίας της κάθε φωτογραφίας ξεχωριστά. Ο προγραμματιστής χρειάζεται να εντοπίσει και να προσδιορίσει που ακριβώς στην κάθε φωτογραφία βρίσκεται το κάθε αντικείμενο. Υπάρχει η δυνατότητα ταυτόχρονης προσθήκης ετικετών για τα διάφορα αντικείμενα με αλλαγή της κλάσης, η οποία επιτυγχάνεται με το πάτημα ενός μόνο κουμπιού, χωρίς να χρειαστεί να διακοπεί το πρόγραμμα και να τεθεί ξανά σε λειτουργία.

Στις Εικόνες 3-4 και 3-5 αποτυπώνεται ο τρόπος με τον οποίο μαρκάρονται οι περιοχές ενδιαφέροντος στην κάθε εικόνα. Κατά την διαδικασία δημιουργίας πλαισίου, αναγράφεται και η κατηγορία που έχει επιλεχθεί. Για κάθε κατηγορία αναπτύσσεται ένας διαφορετικός συνδυασμός χρωμάτων προς αποφυγή σύγχυσης τους από τον χρήστη. Με την ολοκλήρωση της διαδικασίας, ο χειριστής εξέρχεται του κώδικα (πατώντας στο πληκτρολόγιο Ctrl+C).



Εικόνα 3-4: Μαρκάρισμα στοιχείων ενδιαφέροντος στην φωτογραφία.

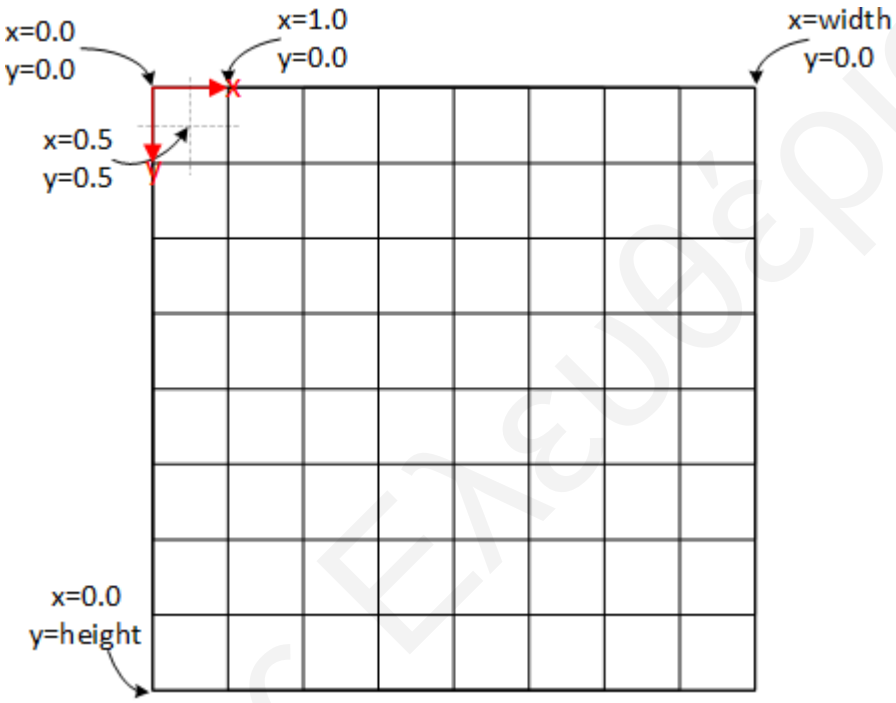
Εικόνα 3-5: Αρχείο .txt που αντιστοιχεί στην Εικόνα 3-4.

< object-class-ID> <X center> <Y center> <Box width> <Box height>

Εικόνα 3-6: Σχεδιάγραμμα επεξήγησης περιεχομένου Εικόνας 3-5.

Με την ολοκλήρωση της διαδικασίας δημιουργείται νέος φάκελος με όλες τις πληροφορίες για την κάθε εικόνα (Εικόνα 3-5). Στην πρώτη στήλη του αρχείου αναγράφεται η κλάση στην οποία ανήκει το περιεχόμενο κάθε πλαισίου. Οι επόμενες τέσσερεις στήλες αφορούν αποκλειστικά τη θέση του πλαισίου σύμφωνα με το σύστημα συντεταγμένων της εικόνας, με τρόπο ίδιο με αυτόν που αποτυπώνεται στην

Εικόνα 3-6. Αναφορικά με το εν λόγω σύστημα, όπως φαίνεται και στην Εικόνα 3-7, οι πρώτες δύο στήλες στο αρχείο με τις ετικέτες αφορούν τις συντεταγμένες του κέντρου του πλαισίου. Οι επόμενες δύο είναι για το μήκος και πλάτος της εικόνας. Έτσι προσδιορίζεται η ακριβής θέση του εκάστοτε πλαισίου στην κάθε μία από τις εικόνες. Ακολούθως, ο υπολογιστής εκπαιδεύεται στο να αντιληφθεί το μοτίβο έντασης των pixels στο κάθε ένα από τα αντικείμενα ενδιαφέροντος.



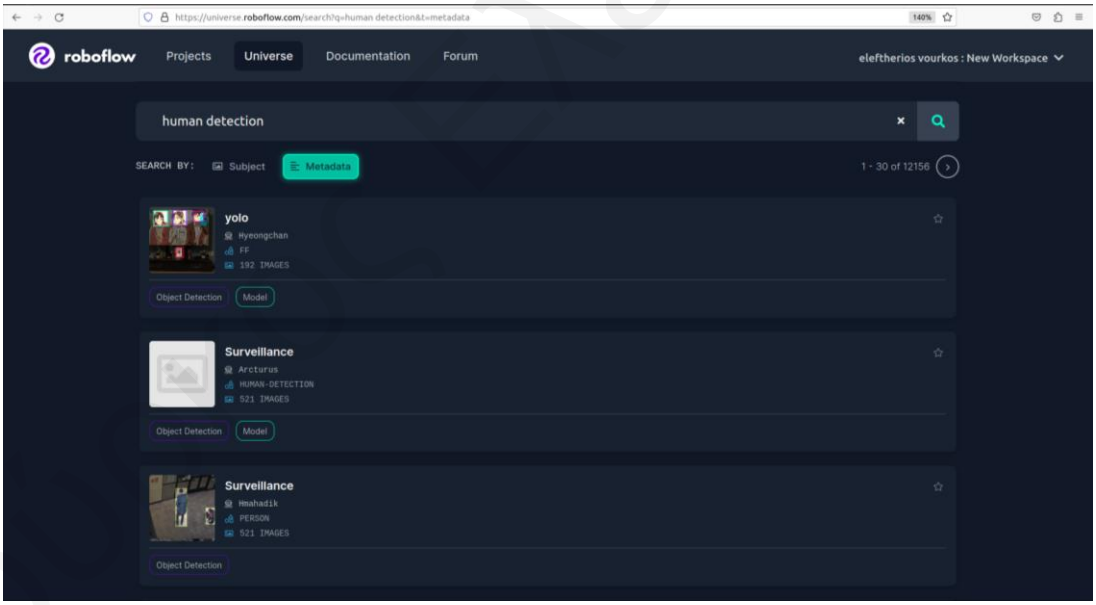
Εικόνα 3-7: Σύστημα συντεταγμένων εικόνας [21].

Με την ολοκλήρωση του συνόλου δεδομένων αποθηκεύεται και μπορεί να χρησιμοποιηθεί σε περισσότερες από μία εφαρμογές. Παράλληλα υπάρχει η δυνατότητα ανανέωσης του συνόλου καθώς επίσης και η συνένωση μεταξύ δύο ή περισσοτέρων συνόλων δεδομένων. Επίσης, ένα ολοκληρωμένο σύνολο δεδομένων δύναται να τύχει επεξεργασίας προκειμένου να μειωθούν οι κλάσεις ή να αντληθεί μόνο μία κλάση από αυτό. Ο λόγος είναι για σκοπούς διαχείρισης υπολογιστικού κόστους. Το να εκπαιδεύσεις ένα υπολογιστή να αναγνωρίζει ανθρώπινες φιγούρες, είναι σαφώς πιο γρήγορο από το να τον εκπαιδεύσεις να αναγνωρίζει ανθρώπους και ακόμη 79 αντικείμενα όπως είναι για παράδειγμα ο κώδικας coco128.

Αφού έγινε αναφορά στο dataset coco128, αξίζει να σημειωθεί ότι αρκετός κόσμος ανά τακτά χρονικά διαστήματα αναρτά σε πλατφόρμες στο διαδίκτυο έτοιμα τα σύνολα δεδομένων. Τις εικόνες δηλαδή μαζί με τις ετικέτες, έτοιμα για λήψη και

εκπαίδευση. Αυτό εξοικονομεί σημαντικό χρόνο από κάποιον που θέλει να κάνει μία απλή αναγνώριση ενός αντικειμένου. Μπορεί χωρίς οποιαδήποτε χρέωση να αναζητήσει σε μία πλατφόρμα dataset σχετικά με το αντικείμενο ενδιαφέροντος του. Από την λίστα που θα προκύψει μπορεί ο ίδιος να επιλέξει με γνώμονα την ακρίβεια που χρειάζεται και το υπολογιστικό κόστος που είναι διατεθειμένος να δαπανήσει.

Ακόμη όμως και για πιο σύνθετα και συγκεκριμένα σύνολα δεδομένων, το Roboflow για παράδειγμα όπως φαίνεται και στην Εικόνα 3-8, δίνει την δυνατότητα πρόσβασης, είτε ελεγχόμενης είτε ανοικτής. Για σκοπούς Διατριβής χρειάστηκε να γίνει αναγνώριση μέσω λήψης από την κάμερα του drone. Σε καμία από τις φωτογραφίες όμως που περιλαμβάνει το coco128, για παράδειγμα, δεν υπάρχει λήψη από κάποιο ύψος. Εκπαιδεύοντας τον υπολογιστή με το σύνολο δεδομένων coco128 προκύπτουν οι εξής δύο πτυχές. Για βίντεο των οποίων η λήψη προέρχεται από το έδαφος, ο υπολογιστής παρουσιάζει μεγάλη ακρίβεια, ωστόσο για βίντεο το οποίο πάρθηκε από drone σε μεγάλο ύψος, το μοντέλο υστερεί.

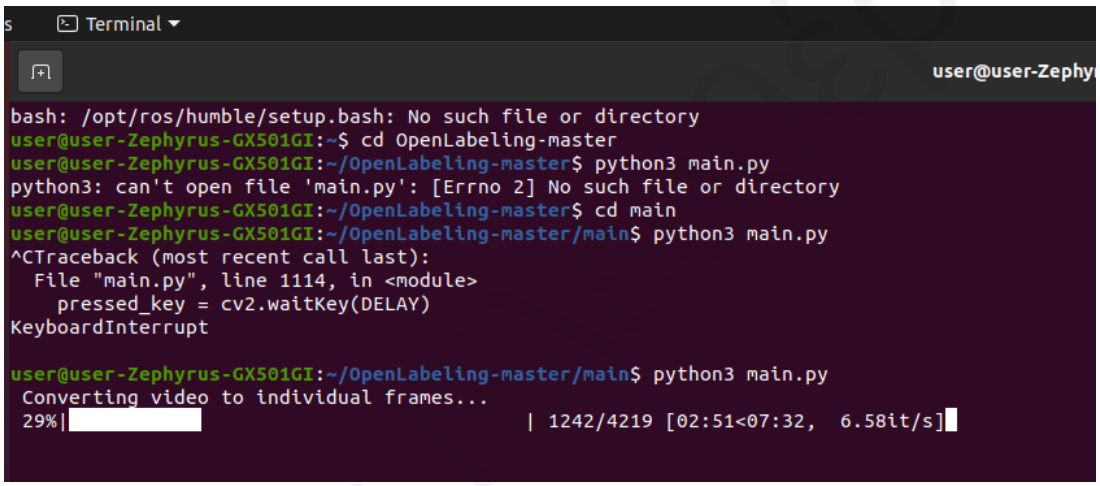


Εικόνα 3-8: Έτοιμα Dataset από Roboflow [24].

Χρειάζεται λοιπόν εκπαίδευση με ένα διαφορετικό σύνολο δεδομένων. Υπό άλλες περιπτώσεις θα ναυλωνόταν το drone σε ειδική αποστολή λήψης εικόνων από ποικίλα ύψη περιτριγυρίζοντας τα αντικείμενα ενδιαφέροντος από διάφορες γωνίες. Έπειτα, ο χειριστής θα έπρεπε να συλλέξει τις εικόνες και να τις κατηγοριοποιήσει με ετικέτες. Χαμένος χρόνος και φαιά ουσία μπορεί να χαρακτηριστεί η πιο πάνω

διαδικασία από την στιγμή που υπάρχει έτοιμο dataset με λήψεις από drone. Μπορεί να μην υπάρχει η πληθώρα επιλογών που παρουσιάζεται σε άλλα σύνολα δεδομένων, από την στιγμή όμως που μπορεί να τροποποιηθεί και να προστεθούν εκ νέου δεδομένα αποτελεί την πιο ωφέλιμη τόσο για το περιβάλλον, όσο και για τον άνθρωπο λύση.

Τέλος, όσο αφορά την κατηγοριοποίηση με ετικέτες, κάτι το οποίο θα ήταν παράληψη να μην ειπωθεί είναι το ότι το πακέτο μπορεί να επεξεργαστεί απευθείας και βίντεο. Νοουμένου ότι το βίντεο βρίσκεται στον ίδιο φάκελο με τις εικόνες προς επεξεργασία, ο κώδικας χωρίζει το βίντεο σε frames και τα παραθέτει με τη σειρά προς ετικετοποίηση (labelling – βλ. Εικόνα 3-9).



Εικόνα 3-9: Επεξεργασία βίντεο μέσω διάσπασης σε frames.

3.2 Εκπαίδευση (Training)

Κατά την εκπαίδευση ο υπολογιστής τρέχει τα σύνολα δεδομένων που έχουν καταχωρηθεί νωρίτερα. Είτε αυτά πάρθηκαν έτοιμα, είτε δημιουργήθηκαν από τον χρήστη, αρκεί ένας φάκελος με εικόνες και ένας δεύτερος με τις ετικέτες των εικόνων. Η κάθε εικόνα έχει το αντίστοιχο .txt file στον φάκελο με τις ετικέτες και κάθε ζευγάρι συνδέεται μεταξύ του μέσω των ονομασιών των δύο αρχείων. Ανεξάρτητα με το περιεχόμενο των αρχείων, η εκπαίδευση θα γίνει νοουμένου ότι τα δύο folders περιέχουν μόνο αρχεία τα οποία αποτελούν ζευγάρια. Για παράδειγμα, εάν μία φωτογραφία δεν ικανοποιεί τις απαιτήσεις του έργου τότε μπορεί απλά να αφαιρεθεί το περιεχόμενο στο αρχείο με τις ετικέτες που της αντιστοιχεί, έτσι θα περάσει απαρατήρητη κατά την εκπαίδευση. Όταν όμως υπάρχει μεγαλύτερος αριθμός εικόνων που περισσεύουν και είναι ανάγκη να απομακρυνθούν, τότε διαγράφοντάς τις ο

χειριστής οφείλει να αναζητήσει προσεκτικά τα αντίστοιχα .txt files και να τα διαγράψει επίσης.

Πέρα από την προετοιμασία σχετικά με τα σύνολα δεδομένων, ένα καλό training εξαρτάται από ένα συγκρότημα παραμέτρων που καθορίζονται από τον χειριστή. Οι περισσότερες από αυτές έχουν μία προκαθορισμένη τιμή, ωστόσο περίπτωση από περίπτωση διαφέρει. Ανάλογα με την εφαρμογή, η απαιτούμενη εκπαίδευση διαφέρει ώστε ο υπολογιστής να είναι σε θέση πλέον να αναγνωρίζει το οτιδήποτε υπό τις περιστάσεις της εν λόγω εφαρμογής. Για παράδειγμα η αναγνώριση ανθρωπίνων φιγούρων κατά τη διάρκεια λήψεων από drone είναι αρκετά πιο δύσκολη σε σύγκριση με την αναγνώριση από στατική κάμερα ασφαλείας. Ως εκ τούτου για την πρώτη εφαρμογή ο υπολογιστής χρειάζεται να εκπαιδευτεί καλύτερα, αντίθετα με τη δεύτερη για την οποία ο εντοπισμός μπορεί να επιτευχθεί ακόμη και με ένα σύνολο pre-trained weights.

Τα weights είναι το ποιοτικό αποτέλεσμα της εκπαίδευσης, δηλαδή ο υπολογιστής ότι έχει αποκομίσει μέσω της εκπαίδευσης, τα αποθηκεύει κωδικοποιημένα στα εν λόγω αρχεία. Έτσι ο χειριστής μπορεί να καλεί αυτό το αρχείο προκειμένου να γίνει η ανίχνευση. Εάν ο υπολογιστής δεν έχει εκπαιδευτεί αρκετά για αναγνώριση συγκεκριμένου αντικειμένου, ή έχει γίνει κάποιο λάθος σχετικά με τα weights, τότε η αναγνώριση δεν θα επιτευχθεί ή ακόμα και να γίνει θα έχει χαμηλή ακρίβεια.

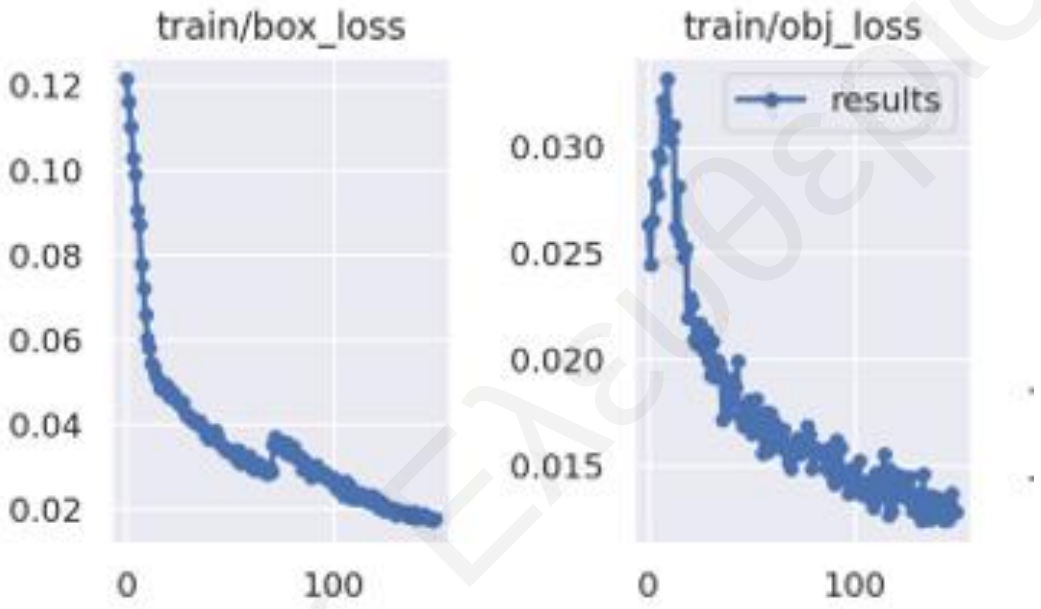
Με άλλα λόγια, τα weights αναφέρονται στις παραμέτρους μάθησης του Neural Network κατά την διάρκεια της εκπαίδευσης. Οργανώνονται σε διαφορετικά επίπεδα του δικτύου και χρησιμοποιούνται για σκοπούς ανίχνευσης αντικειμένων. Σε μια προσπάθεια εμβάθυνσης, το YOLO χρησιμοποιεί ένα δίκτυο CNN το οποίο αποτελείται από πολλά επίπεδα μεταξύ των οποίων συνελικτικά, δειγματοληπτικά και πλήρως συνδεδεμένα επίπεδα. Καθένα από αυτά αντιστοιχεί σε ένα σύνολο μαθησιακών weights τα οποία προσαρμόζονται κατά τη διάρκεια της εκπαίδευσης προκειμένου η απόδοση του μοντέλου να βελτιστοποιηθεί. Αποθηκεύονται συνήθων σε binary files σε μορφή .pt και περιέχουν τις μαθησιακές παραμέτρους του δικτύου. Εκτός από τα weights, το YOLO χρειάζεται ένα αρχείο διαμόρφωσης που βρίσκεται στον φάκελο config και καθορίζει την αρχιτεκτονική του μοντέλου όπως επίσης και άλλες παραμέτρους. Μερικές από αυτές είναι τα anchors και το μέγεθος εισόδου της εικόνας. Τα weights μαζί με το αρχείο διαμόρφωσης συνεργάζονται ώστε να καταστεί εφικτή η ανίχνευση αντικειμένων σε νέες εικόνες και βίντεο.

Δύο άλλες παράμετροι που επηρεάζουν σημαντικά το αποτέλεσμα και είναι ανάγκη να καθορίζονται στοχευμένα είναι ο αριθμός των επαναλήψεων (epochs) και το batch size. Το YOLO αποσκοπεί στη μάθηση μέσω επαναλήψεων, έτσι η εκπαίδευση του υπολογιστή γίνεται σε πολλαπλές επαναλήψεις, καθεμία εκ των οποίων περιλαμβάνει ένα πλήρες πέρασμα από όλα τα δεδομένα εκπαίδευσης. Ο αριθμός των επαναλήψεων επηρεάζει την ποιότητα εκπαίδευσης του μοντέλου και την ταχύτητα σύγκλισης. Όσο μεγαλύτερος είναι ο αριθμός των επαναλήψεων, τόσο καλύτερα αποτελέσματα προκύπτουν, ωστόσο αυξάνεται το υπολογιστικό κόστος. Από την άλλη το batch size ή αλλιώς μέγεθος παρτίδας, αντιστοιχεί με τον αριθμό των εικόνων που επεξεργάζεται το YOLO σε κάθε επανάληψη. Μεγαλύτερο batch size μειώνει τον συνολικό χρόνο εκπαίδευσης, αλλά πιθανόν να χρειαστεί αύξηση της απαιτούμενης μνήμης. Παράλληλα, αναπτύσσεται μεγαλύτερο ρίσκο αύξησης της αστάθειας κατά τη διάρκεια της εκπαίδευσης.

Όσον αφορά τη διακύμανση της αστάθειας, μπορεί να αντιμετωπιστεί είτε με στοχευμένη ρύθμιση των παραμέτρων, είτε με τη μέθοδο δοκιμής και σφάλματος. Από την άλλη προκύπτουν περιπτώσεις όπου η αύξηση του batch size είναι απαραίτητη, ωστόσο η μνήμη του υπολογιστή δεν το επιτρέπει. Σε καταστάσεις όπως την προαναφερθείσα η λύση έρχεται με τη δημιουργία ενός swapfile. Ένα swapfile, ή αλλιώς αρχείο εικονικής μνήμης, αποτελεί αρχείο που χρησιμοποιείται από ένα λειτουργικό σύστημα (Linux, macOS ή Windows) ως προσωρινή επέκταση της φυσικής μνήμης. Όταν η μνήμη RAM (Random Access Memory), ενός υπολογιστή είναι πλήρης, το λειτουργικό σύστημα έχει τη δυνατότητα μετακίνησης δεδομένων που χρησιμοποιούνται λιγότερο συχνά από την RAM, στο swapfile στον σκληρό δίσκο. Με την κίνηση αυτή δημιουργείται χώρος στην RAM για δεδομένα που χρησιμοποιούνται συχνότερα. Η εν λόγω διαδικασία αποτρέπει την εξάντληση της μνήμης του υπολογιστή η οποία με τη σειρά της πιθανόν να προκαλέσει επιβράδυνση του συστήματος ή ακόμα και κατάρρευση. Το swapfile διαχειρίζεται συνήθως το λειτουργικό σύστημα και είναι διαφανές στον χρήστη. Εν κατακλείδι, μπορεί να μονιμοποιηθεί και να υφίσταται έως τη στιγμή που κρίνεται αναγκαίο, ενώ μπορεί παράλληλα να ενεργοποιείται και να απενεργοποιείται μέσω εντολών στο Terminal, χωρίς να χρειάζεται διαγραφή του όταν δεν χρησιμοποιείται.

Στην Εικόνα 3-10, το γράφημα στα αριστερά παρουσιάζει το μέσο τετραγωνικό σφάλμα συναρτήσει των επαναλήψεων που εκπαιδεύεται ο υπολογιστής. Είναι προφανές ότι το σφάλμα ελαττώνεται καθώς ο αριθμός των επαναλήψεων αυξάνεται.

Το αποδεκτό τετραγωνικό σφάλμα καθορίζεται από την εκάστοτε εφαρμογή και ποικίλει από εφαρμογή σε εφαρμογή. Αυτό που αντλείται ξεκάθαρα όμως από το γράφημα είναι ότι πέραν των 120 επαναλήψεων το σφάλμα τείνει να σταθεροποιηθεί, παρουσιάζει δηλαδή πολύ μικρή πτώση στις επόμενες 30 επαναλήψεις. Για ένα περιορισμένο σύνολο δεδομένων αυτό πιθανόν να μην επηρεάζει σημαντικά, ωστόσο σε μεγαλύτερα σύνολα όσο αυξάνονται οι επαναλήψεις, το υπολογιστικό κόστος παρουσιάζει μεγάλη άνοδο.



Εικόνα 3-10: Αποτελέσματα από training.

Το δεύτερο γράφημα απεικονίζει το ποσοστό αξιοπιστίας των μετρήσεων. Ο κατακόρυφος άξονας είναι η απόκλιση και ο οριζόντιος οι επαναλήψεις. Με παρόμοιο τρόπο, καθώς οι επαναλήψεις της εκπαίδευσης του μοντέλου αυξάνονται, η απόκλιση συρρικνώνεται ακόμη περισσότερο με αποτέλεσμα να ανεβαίνει η αξιοπιστία των αποτελεσμάτων. Στο σημείο αυτό παρατηρείται επίσης ότι η καμπύλη αξιοπιστίας δεν είναι τόσο μαζεμένη όσο η καμπύλη σφάλματος. Για παράδειγμα στις 115 επαναλήψεις περίπου, η απόκλιση ξεπερνά το 0.015 κάτι που αντιτίθεται με την απόκλιση στις 105 επαναλήψεις η οποία ανέρχεται μόλις στο 0.014. Σε γενικές γραμμές όμως παρουσιάζει συμπεριφορά ανάλογη με την καμπύλη σφάλματος, όπου με την αύξηση των επαναλήψεων προκύπτουν βελτιωμένα αποτελέσματα.

Με το πέρας κάθε εκπαίδευσης δημιουργείται νέος φάκελος στον οποίο αποθηκεύονται τα αποτελέσματα που προέκυψαν. Υπάρχουν αποτελέσματα σχετικά με το σφάλμα, ακρίβεια, αξιοπιστία όπως επίσης και weights (συντελεστές βαρύτητας) που θα χρησιμοποιηθούν στη συνέχεια για αναγνώριση, νοουμένου ότι τα αποτελέσματα πληρούν τις απαιτήσεις της εφαρμογής.

3.3 Επιβεβαίωση (Validation)

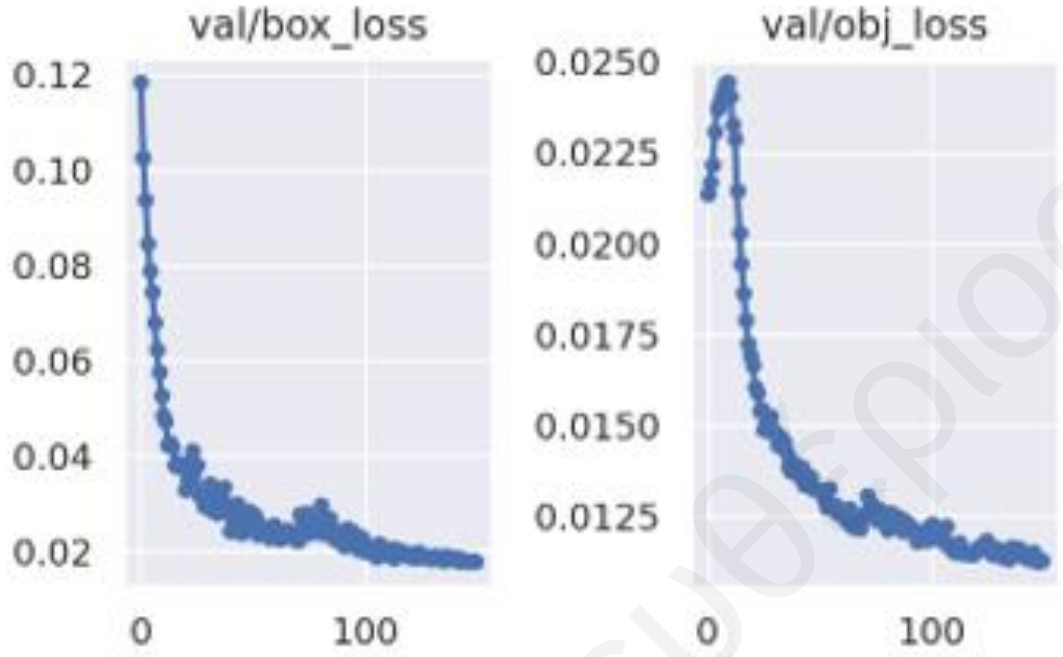
Αναφορικά με το κομμάτι της επιβεβαίωσης, ο αλγόριθμος εκπαιδεύεται στην αναγνώριση του αντικείμενου ενδιαφέροντος σε εικόνες που πρώτη φορά αντιμετωπίζει, ή λειτουργεί σαν να τις αντιμετωπίζει για πρώτη φορά. Πρόκειται για πανομοιότυπο υλικό με αυτό της εκπαίδευσης, ωστόσο το περιεχόμενο των φακέλων δύναται να είναι διαφορετικό από αυτό της εκπαίδευσης.

Είναι δηλαδή σαν να γράφει διαγώνισμα και ενώ γνωρίζει τις λύσεις (την θέση των αντικειμένων ενδιαφέροντος - υπάρχουν στον φάκελο με τις ετικέτες) τις αγνοεί. Τρέχει την διαδικασία βασισμένος στην εκπαίδευση που έτυχε και έπειτα συγκρίνει τα αποτελέσματα που προέκυψαν με το περιεχόμενο στον φάκελο με τις ετικέτες, υλοποιώντας με τον τρόπο αυτό την επιβεβαίωση των αποτελεσμάτων. Το Validation αποτελεί μεταβατικό στάδιο ώστε ο υπολογιστής να ολοκληρώσει την προετοιμασία και να είναι πλέον σε θέση να εκτελέσει αναγνώριση.

Η διαδικασία επιβεβαίωσης του Training που έχει γίνει, διεξάγεται με τον ίδιο ακριβώς τρόπο όπως η εκπαίδευση. Στα περισσότερα πακέτα που υπάρχουν έτοιμα στο διαδίκτυο η επιβεβαίωση γίνεται σαν συνέχεια του Training, χωρίς επιπλέον εντολή. Τα αποτελέσματα του Validation σε αυτές τις περιπτώσεις αποθηκεύονται στον ίδιο φάκελο με τα αποτελέσματα της εκάστοτε εκπαίδευσης.

Με τον ίδιο τρόπο τα γραφήματα στην Εικόνα 3-11 επιβεβαιώνουν ότι όσο αυξάνεται ο αριθμός των επαναλήψεων, έχουμε τόσο πιο αξιόπιστα αποτελέσματα. Θα ήταν καλό να σημειωθεί ότι από κάποια τιμή και έπειτα, περίπου στις 120 επαναλήψεις, τα αποτελέσματα βελτιώνονται ελάχιστα, ενώ το υπολογιστικό κόστος αυξάνεται, άρα κάπου εκεί τοποθετείται ένα όριο. Ωστόσο για τις περιπτώσεις που το Validation γίνεται αυτόματα μετά το Training δεν υπάρχει η δυνατότητα τροποποίησης των παραμέτρων. Αυτό πάει να πει ότι εάν η εκπαίδευση έγινε με batch size 8 και epoch 150, τότε και η επιβεβαίωση θα διεξαχθεί με τις ίδιες παραμέτρους. Στο Validation όμως, νοουμένου ότι έχει προηγηθεί ένα καλό Training, το υπολογιστικό κόστος είναι

κατά πολύ μικρότερο. Εξάλλου, και το σύνολο δεδομένων προς ανάλυση είναι πιο περιορισμένο στο Validation, με τον τρόπο που περιγράφεται στο Κεφάλαιο 2.7.



Εικόνα 3-11: Αποτελέσματα από Validation.

3.4 Έλεγχος (Test)

Αφού προηγηθεί εκπαίδευση και επιβεβαίωση του μοντέλου, απομένει να γίνει έλεγχος για το πόσο ακριβές και αξιόπιστο είναι το μοντέλο. Στο σημείο αυτό ο υπολογιστής καλείται να ανιχνεύσει αντικείμενα σε εικόνες ή βίντεο για τα οποία έχει διεξαχθεί η προετοιμασία μέσω του Training και του Validation. Σύμφωνα πάντα με τα όσα έχουν ειπωθεί στο Κεφάλαιο 2, ένα σύνολο εικόνων της τάξης του 10% - 20% του συνολικού συνόλου δεδομένων, προορίζεται για τον έλεγχο του μοντέλου. Ο υπολογιστής εκτίθεται στην διαδικασία αναγνώρισης σε εικόνες ή/και βίντεο με τα οποία έρχεται για πρώτη φορά σε επαφή.

Για τη διαδικασία του ελέγχου χρησιμοποιούνται τα weights που έχουν δημιουργηθεί στο Training που προηγήθηκε. Η διαδικασία επιλογής των weights γίνεται μέσω αξιολόγησης των αποτελεσμάτων της κάθε προσπάθειας. Η αξιολόγηση με τη σειρά της βασίζεται στο Confusion Matrix, καθώς επίσης και στα γραφήματα που απεικονίζουν τα ποσοστά σφάλματος και ακρίβειας για τα οποία θα ειπωθούν περισσότερα στην συνέχεια.

Με το πέρας της διαδικασίας ελέγχου, το μοντέλο αποθηκεύει τα αποτελέσματα και ο χειριστής καλείται να τα αξιολογήσει. Νοουμένου ότι προηγήθηκε ορθή επιλογή των weights και δεν υπάρχει κάποιο σφάλμα στην καταχώρηση των κλάσεων, το μοντέλο θα έχει εντοπίσει και κατηγοριοποιήσει τα αντικείμενα στα αρχεία βάσει της εκπαίδευσης που έχει τύχει. Αυτό που απομένει είναι να αξιολογηθεί το ποσοστό αξιοπιστίας σε κάθε ανίχνευση. Εάν όντως τα αρχεία που χρησιμοποιήθηκαν διαφέρουν από αυτά στα οποία βασίστηκε η εκπαίδευση και η επιβεβαίωση, τότε αξιοπιστία της τάξης του 85% και έπειτα είναι ευρέως αποδεκτή και αφήνει να νοηθεί ότι τα weights που εξήχθησαν από το Training μπορούν να χρησιμοποιηθούν και στις εφαρμογές ανίχνευσης.

Το στάδιο του ελέγχου χρίζει εξαιρετικής σημασίας εφόσον είναι η τελευταία διαδικασία προτού ένα πακέτο προωθηθεί για αξιοποίηση σε μία εφαρμογή.

3.5 Ανάλυση Αποτελεσμάτων

Προκειμένου να γίνει ορθή αξιολόγηση των αποτελεσμάτων χρειάζεται σχολαστική μελέτη των διαγραμμάτων που προκύπτουν. Τα διαγράμματα αποτυπώνουν γραφικά το αποτέλεσμα της διαδικασίας και μπορούν να χαρακτηριστούν ως χείμαρρος πληροφοριών σχετικά με το εκάστοτε run. Με το πέρας της διαδικασίας δημιουργείται ξεχωριστός φάκελος που περιέχει τα αποτελέσματα κάθε περίπτωσης, καθώς επίσης και τα weights.

Προκειμένου όμως να γίνει ορθολογική μελέτη και σύγκριση των αποτελεσμάτων ο λειτουργός χρειάζεται να γνωρίζει τι ακριβώς αποτυπώνει κάθε διάγραμμα, ενώ παράλληλα να είναι σε θέση να εξαγάγει κάθε πληροφορία που προσφέρει ένα διάγραμμα. Για τον λόγο αυτό στο συγκεκριμένο υποκεφάλαιο θα γίνει αναφορά στις παραμέτρους που βασίζονται τα διαγράμματα, ενώ στη συνέχεια αποτυπώνεται ο τρόπος αξιοποίησης των διαγραμμάτων αυτών προκειμένου να προκύψουν βάσιμα αποτελέσματα.

3.5.1Bounding Box regression loss (box_loss)

Το bounding box regression loss είναι το μέσο τετραγωνικό σφάλμα ή αλλιώς μία μετρική απώλεια, βασισμένη σε μία συνάρτηση απωλειών. Υπολογίζει πόσο στενά μαρκάρουν τα προβλεπόμενα bounding boxes σε σύγκριση με τις ετικέτες στις εικόνες του συνόλου δεδομένων. Συναντάται σε γραφήματα εκπαίδευσης και επιβεβαίωσης

συναρτήσει των επαναλήψεων. Όσο η τιμή του ελαττώνεται, τόσο πιο ακριβή είναι τα προβλεπόμενα πλαίσια οριοθέτησης.

Συγκεκριμένα, υπολογίζεται ως το σφάλμα μεταξύ του πραγματικού bounding box και του bounding box που προβλέπει ο αλγόριθμος. Αποτελεί μεταξύ άλλων ποσότητα κόστους που αυξάνεται όσο μεγαλύτερο είναι το σφάλμα μεταξύ πραγματικών και προβλεπόμενων παραμέτρων του bounding box. Τέλος, συνεισφέρει στον υπολογισμό του συνολικού κόστους που χρησιμοποιείται κατά την εκπαίδευση του αλγορίθμου.

3.5.2Classification loss (cls_loss)

Το classification loss είναι επίσης μία μετρική απώλεια βασισμένη σε μία συνάρτηση απωλειών, ενώ συναντάται συχνά και ως «Cross Entropy». Μετρά την ορθότητα της ταξινόμησης κάθε προβλεπόμενου οριοθετημένου πλαισίου. Κάθε μεμονωμένο πλαίσιο μπορεί να περιέχει μία κλάση αντικειμένου ή ετικέτα φόντου (μηδενική εικόνα). Για χαμηλότερες τιμές σημαίνει ότι το μοντέλο προβλέπει με μεγαλύτερη ακρίβεια την κλάση των αντικειμένων.

Υπολογίζεται ως το σφάλμα μεταξύ της πραγματικής κατηγορίας του αντικειμένου και της κατηγορίας που προβλέπει ο αλγόριθμος. Όσο μεγαλύτερο είναι το σφάλμα μεταξύ των δύο πιο πάνω κατηγοριών, τόσο αυξάνεται το cls_loss.

3.5.3Object loss (obj_loss)

Το object loss είναι ένα μέτρο που χρησιμοποιείται στον αλγόριθμο για να υπολογίσει το κόστος της απώλειας. Η απώλεια αντικειμενικότητας είναι η αυτοπεποίθηση που παρουσιάζει το μοντέλο σχετικά με την παρουσία του αντικειμένου. Με άλλα λόγια είναι το πόσο σίγουρο εμφανίζεται το δίκτυο σχετικά με την πρόβλεψή του. Αν ένα αντικείμενο εμφανίζεται στο κελί το obj_loss θα είναι μικρό και αντίθετα.

Πιο συγκεκριμένα, περιλαμβάνει δύο στοιχεία, τον υπολογισμό του σφάλματος πρόβλεψης της ύπαρξης αντικειμένου (objectness prediction error), και τον υπολογισμό του σφάλματος πρόβλεψης της θέσης του αντικειμένου (bounding box prediction error). Το obj_loss υπολογίζεται ως άθροισμα των δύο αυτών σφαλμάτων και χρησιμοποιείται ως μέτρο αξιολόγησης της απόδοσης του μοντέλου.

3.5.4Differentiable Focal Loss (dfl_loss)

Το differentiable focal loss είναι μία νέα προσθήκη στην αρχιτεκτονική YOLO, στο YOLOv8 και χρησιμοποιείται ως συνάρτηση απώλειας για την εκπαίδευση του δικτύου. Εν ολίγοις επιτρέπει στο δίκτυο να εστιάζει σε μεγαλύτερο βαθμό στα δύσκολα παραδείγματα και να αγνοεί τα πιο εύκολα. Με την μέθοδο αυτή μειώνεται η επίδραση της ανισορροπίας των κλάσεων και βελτιώνεται η απόδοση του αλγόριθμου στην ανίχνευση αντικειμένων.

Πιο αναλυτικά, υπολογίζεται για κάθε πρόβλεψη αντικειμένου στο δίκτυο και προστίθεται στη συνολική απώλεια του δικτύου. Η συνολική απώλεια αποτελείται από στοιχεία όπως η απώλεια του σημείου ενδιαφέροντος (objectness loss) και η απώλεια των συντεταγμένων του πλαισίου (box coordinate loss). Η dfl εφαρμόζεται στο στοιχείο της απώλειας που σχετίζεται με πρόβλεψη της κλάσης του αντικειμένου (class loss), και μπορεί να προσαρμοστεί ανάλογα με τη δυσκολία του παραδείγματος.

3.5.5Precision

Στον αλγόριθμο YOLO, οι θετικές και αρνητικές προβλέψεις αφορούν την ταξινόμηση του κάθε κελιού του πλέγματος σε σχέση με την παρουσία αντικειμένου στο επίπεδό του. Πιο συγκεκριμένα, κάθε κελί του πλέγματος εξετάζεται για την παρουσία ή μη αντικειμένου. Αν το κελί περιέχει το κέντρο ενός αντικειμένου, τότε θεωρείται ότι η πρόβλεψη του αλγόριθμου είναι θετική για το συγκεκριμένο κελί. Αντίθετα, εάν το κελί δεν περιέχει το κέντρο κάποιου αντικειμένου, τότε η πρόβλεψη για το συγκεκριμένο κελί θεωρείται αρνητική.

Οι θετικές προβλέψεις είναι σημαντικές για τον υπολογισμό των αντίστοιχων απωλειών στην εκπαίδευση του YOLO. Οι αντίστοιχες απώλειες συνδέονται με την πρόβλεψη του bounding box του αντικειμένου, καθώς επίσης και με την κατηγοριοποίηση του αντικειμένου στις διαθέσιμες κλάσεις. Στην περίπτωση των αρνητικών προβλέψεων, η απώλεια αναφέρεται στην ακρίβεια του αλγορίθμου στην πρόβλεψη της παρουσίας αντικειμένων.

Οι θετικές και αρνητικές προβλέψεις παίζουν καθοριστικό ρόλο στον βασικό στόχο του YOLO, ο οποίος δεν είναι άλλος από το να προβλέπει με ακρίβεια την παρουσία, τη θέση και την κατηγορία των αντικειμένων σε μία εικόνα αναλύοντας κάθε κελί του πλέγματος και προβλέποντας ένα ορθογώνιο πλαίσιο και ένα ποσοστό αβεβαιότητας για κάθε αντικείμενο. Οι προβλέψεις με την σειρά τους, ενημερώνουν το

μοντέλο για την παρουσία ή απουσία ενός αντικειμένου σε ένα δεδομένο κελί, το οποίο χρησιμοποιείται για να βελτιώσει τις προβλέψεις του μοντέλου και να αυξήσει την ακρίβειά του.

$$P = [\frac{True_{positives}}{True_{positives} + False_{positives}}]$$

3.5.6 Recall

Το recall είναι μία μετρική αξιολόγηση της απόδοσης του αλγορίθμου. Αναφέρεται στο ποσοστό των αντικειμένων που ανιχνεύτηκαν από το YOLO στον συνολικό αριθμό αντικειμένων στο σύνολο δεδομένων. Ένα υψηλό recall υποδηλώνει ότι ο αλγόριθμος μπορεί να ανιχνεύσει σωστά ένα μεγάλο ποσοστό των αντικειμένων στο σύνολο δεδομένων. Αντίθετα, ένα χαμηλό recall υποδηλώνει ότι πολλά αντικείμενα δεν ανιχνεύονται σωστά από τον αλγόριθμο.

$$R = [\frac{True_{positives}}{True_{positives} + False_{negatives}}]$$

3.5.7 Intersection over Union (IOU)

Το IOU είναι μία μετρική συνάρτηση που χρησιμοποιείται για την αξιολόγηση της απόδοσης του αλγορίθμου. Υπολογίζεται ως το ποσοστό της περιοχής του εντοπισμένου αντικειμένου που επικαλύπτεται με την πραγματική περιοχή του αντικειμένου στην εικόνα.

Κατά την ανίχνευση, ο αλγόριθμος χρησιμοποιεί την εν λόγω συνάρτηση προκειμένου να αξιολογήσει το πόσο καλά ένα αντικείμενο ταιριάζει στο bounding box που έχει καθοριστεί. Για υψηλές τιμές IOU, το YOLO θεωρεί ότι η πρόβλεψη του πλαισίου είναι ακριβής και επαρκής. Σε αντίθετη περίπτωση το YOLO θεωρεί ότι η πρόβλεψη του πλαισίου είναι ανεπαρκής και χρειάζεται περαιτέρω βελτίωση.

Παράλληλα, η συνάρτηση χρησιμοποιείται για αξιολόγηση της απόδοσης του δικτύου στα δεδομένα ελέγχου, προκειμένου να αξιολογηθεί η ακρίβεια των bounding boxes που δημιουργούνται από τον αλγόριθμο. Στα αποτελέσματα του YOLO, η τιμή IOU αναφέρεται συνήθως μαζί με τον βαθμό εμπιστοσύνης για κάθε αντικείμενο που έχει ανιχνευθεί. Για υψηλότερες τιμές, νοείται καλύτερη αντιστοιχία μεταξύ

προβλεπόμενων και πραγματικών bounding boxes, άρα και μεγαλύτερη ακρίβεια στον εντοπισμό των αντικειμένων.

3.5.8Threshold

Στα αποτελέσματα του YOLO, το threshold αναφέρεται στον ελάχιστο βαθμό εμπιστοσύνης που απαιτείται για να αναγνωριστεί ένα αντικείμενο από τον αλγόριθμο. Πιο συγκεκριμένα, κάθε φορά που το μοντέλο αναγνωρίζει ένα αντικείμενο στην εικόνα, προκύπτει μία πιθανότητα για το κατά πόσο το αντικείμενο είναι όντως αυτό που αναγνωρίζεται. Η τιμή threshold, ορίζει την κατώτατη αποδεκτή πιθανότητα και εάν η πιθανότητα αυτή είναι πιο κάτω, τότε η αναγνώριση του αντικειμένου απορρίπτεται. Με άλλα λόγια αναγνωρίζει πόσο αξιόπιστα αναγνωρίζονται τα αντικείμενα από το YOLO.

3.5.9Mean Average Precision (mAP_0.5)

Η μέση ακρίβεια, είναι κατά προσέγγιση μία αξιολόγηση της απόδοσης του αλγορίθμου. Αξιολογεί δηλαδή το πόσο αποτελεσματικά ο αλγόριθμος μπορεί να ανιχνεύει αντικείμενα στο υποσύνολο δεδομένων που προορίζονται για έλεγχο.

Το mAP_0.5, υπολογίζει την μέση ακρίβεια ανίχνευσης αντικειμένων για όλες τις κατηγορίες αντικειμένων στο σύνολο δεδομένων, όταν το confidence threshold και το IOU είναι 0.5. Αποτελεί παράλληλα, ένα καλό τρόπο αξιολόγησης της απόδοσης του YOLO, καθώς λαμβάνει υπόψη την ακρίβεια τόσο στον εντοπισμό, όσο και στην ταξινόμηση των αντικειμένων.

Υπολογίζεται συνήθως ως ο μέσος όρος της Precision – Recall καμπύλης για όλες τις κατηγορίες αντικειμένων. Η καμπύλη απεικονίζει την σχέση μεταξύ των δύο για διαφορετικά thresholds του αλγορίθμου. Εν τέλη μία υψηλότερη τιμή mAP_0.5 υποδηλώνει καλύτερη απόδοση του αλγορίθμου στην ανίχνευση αντικειμένων.

3.5.10 Mean Average Precision (mAP_0.5-0.9)

Είναι μία μετρική συνάρτηση που χρησιμοποιείται για αξιολόγηση της απόδοσης του αλγόριθμου. Αναφέρεται στην μέση ακρίβεια για διαφορετικά επίπεδα recall για την ανίχνευση αντικειμένων με διαφορετικά επίπεδα δυσκολίας. Χρησιμοποιεί μία σειρά από τιμές IOU thresholds από 0.5 έως 0.9. Για καθένα από

αυτά υπολογίζει την ακρίβεια στο σύνολο των αντικειμένων που το μοντέλο έχει προβλέψει και έπειτα, υπολογίζει το μέσο όρο της ακρίβειας στα εν λόγω thresholds.

Αξίζει να σημειωθεί ότι η χρήση του mAP_0.5-0.9 είναι αρκετά πιο απαιτητική από τον υπολογισμό του απλού mAP_0.5, καθώς χρειάζεται να εκτιμηθεί η απόδοση του αλγορίθμου σε διάφορα επίπεδα δυσκολίας της ανίχνευσης αντικειμένων.

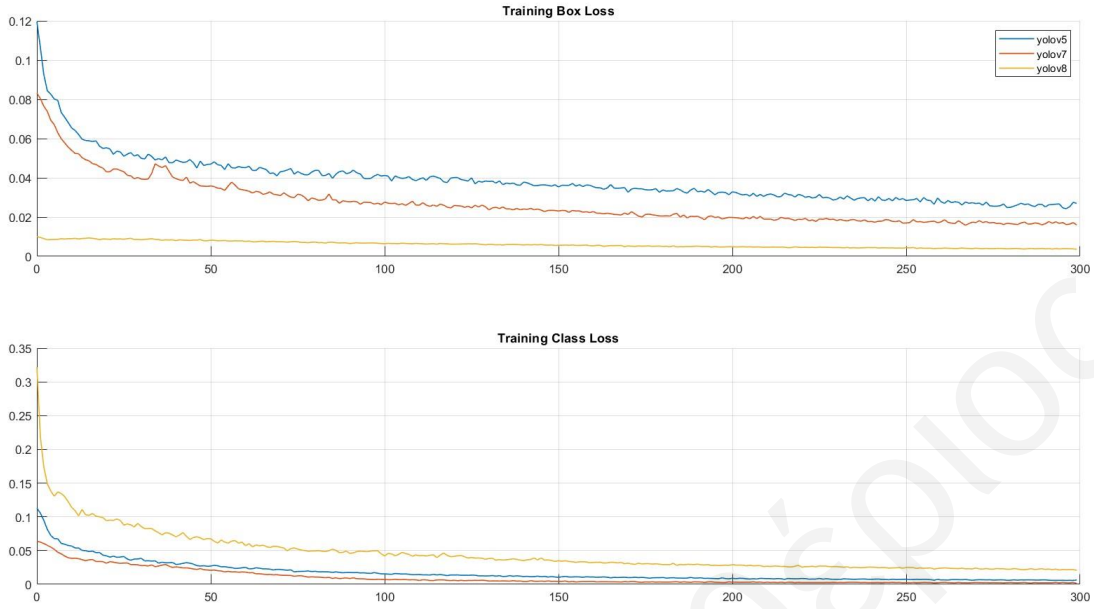
3.6 Σύγκριση YOLOv5, YOLOv7 και YOLOv8

Σε μια προσπάθεια εξαγωγής συμπερασμάτων περί αξιοπιστίας και αποτελεσματικότητας των τριών εκδόσεων του YOLO διεξήχθη μεταξύ τους σύγκριση. Η σύγκριση έγινε με τις ίδιες παραμέτρους και hyperparameters ώστε να διαφανεί κατά πόσο κάποιο μοντέλο υπερισχύει των υπολοίπων επί ίσοις όροις. Για σκοπούς υλοποίησης των τριών μοντέλων χρησιμοποιήθηκε το σύνολο δεδομένων RoboCoco, μία παραλλαγή της έκδοσης του συνόλου Coco128 στην οποία προστέθηκε η κατηγορία «ρομπότ».

Αρχικά, από το bounding box regression loss, στα αποτελέσματα εκπαίδευσης και επικύρωσης στις Εικόνες 3-12 και 3-13 αντίστοιχα, φαίνεται ότι τα YOLOv5 και YOLOv7 έχουν τον ίδιο τύπο καμπύλης με το YOLOv7 να είναι κατά 4% καλύτερο καθ’ όλη την διάρκεια. Παρόλο που το YOLOv8 έχει διαφορετικό τύπο καμπύλης, ο οποίος είναι περίπου 1% από τις πρώτες κιόλας επαναλήψεις, καταλήγει να έχει 4% καλύτερες επιδόσεις απ’ ότι το YOLOv7.

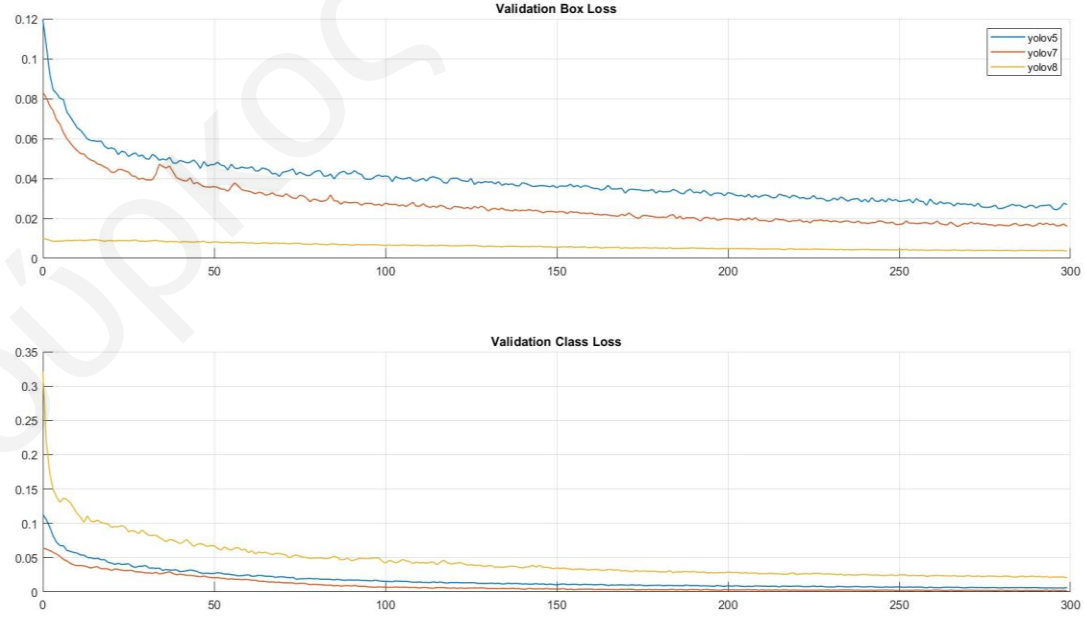
Σύμφωνα με τα αποτελέσματα του cls_loss, τα YOLOv5 και YOLOv7 ξεκινούν με διαφορά 5% και συγκλίνουν στη συνέχεια με το δεύτερο να καταλήγει περίπου στο 0.15%. Παραδόξως, το YOLOv8 είναι κατά πολύ χειρότερο από τα άλλα δύο αφού ξεκινά με 0.2 πάνω από το YOLOv5 και καταλήγει να βρίσκεται 0.05 πάνω και από τα δύο.

Ρίχνοντας μία ματιά στο γράφημα του Precision συναρτήσει των επαναλήψεων, μπορεί να διακρίνει κανείς ότι όλα τα μοντέλα τείνουν να είναι πάνω από 90%. Πέραν των 50 επαναλήψεων, τα μοντέλα συμπεριφέρονται γενικά με τον ίδιο τρόπο, με το YOLOv7 να υπερέχει των υπολοίπων για ένα μικρό ποσοστό. Αναφορικά με τις προβλέψεις των μοντέλων για τα ορθά bounding boxes είναι κοντά μεταξύ τους και αποκτούν τιμές πέραν του 80% τείνοντας στο 90% από την $100^{η}$ έως την $300^{η}$ επανάληψη.

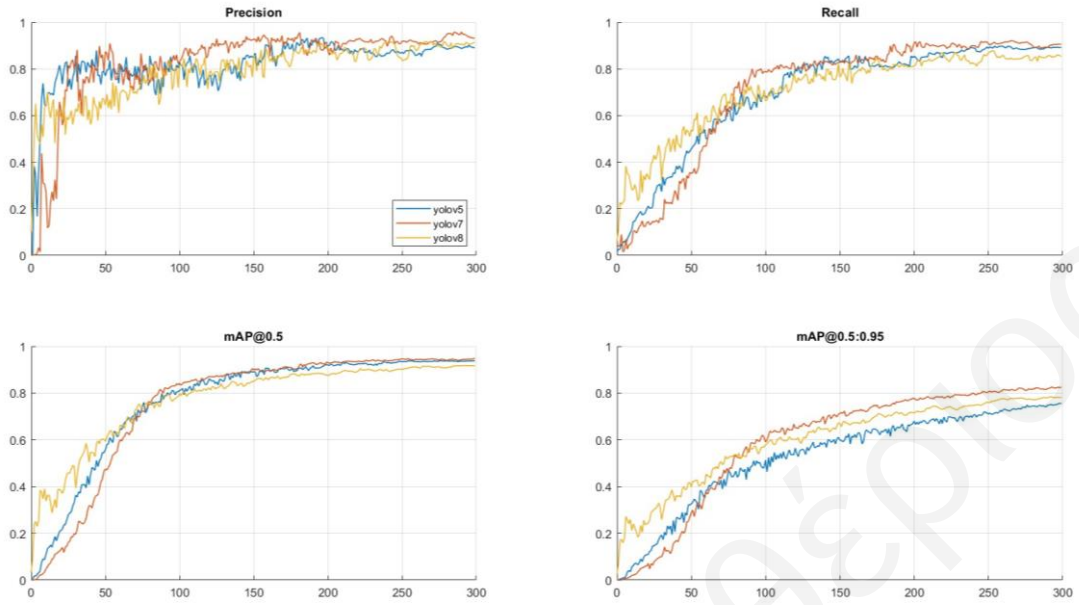


Εικόνα 3-12: Training box – class loss.

Αξίζει να σημειωθεί ότι η απόδοση του YOLOv8 μέχρι την 60η επανάληψη περίπου, υπερέχει των υπολοίπων. Η διαφορά είναι πιο έντονη στα διαγράμματα mAP, ωστόσο οι καμπύλες συγκλίνουν μετά την 60η επανάληψη.

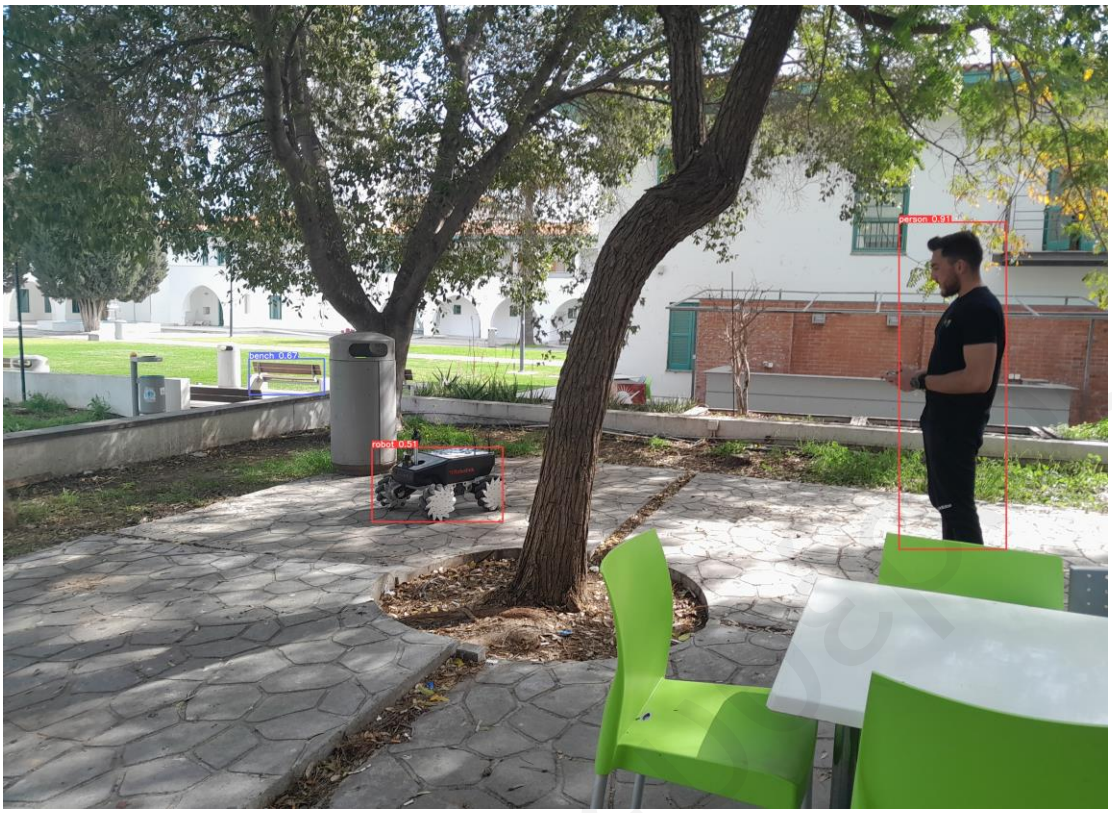


Εικόνα 3-13: Validation box – class loss.

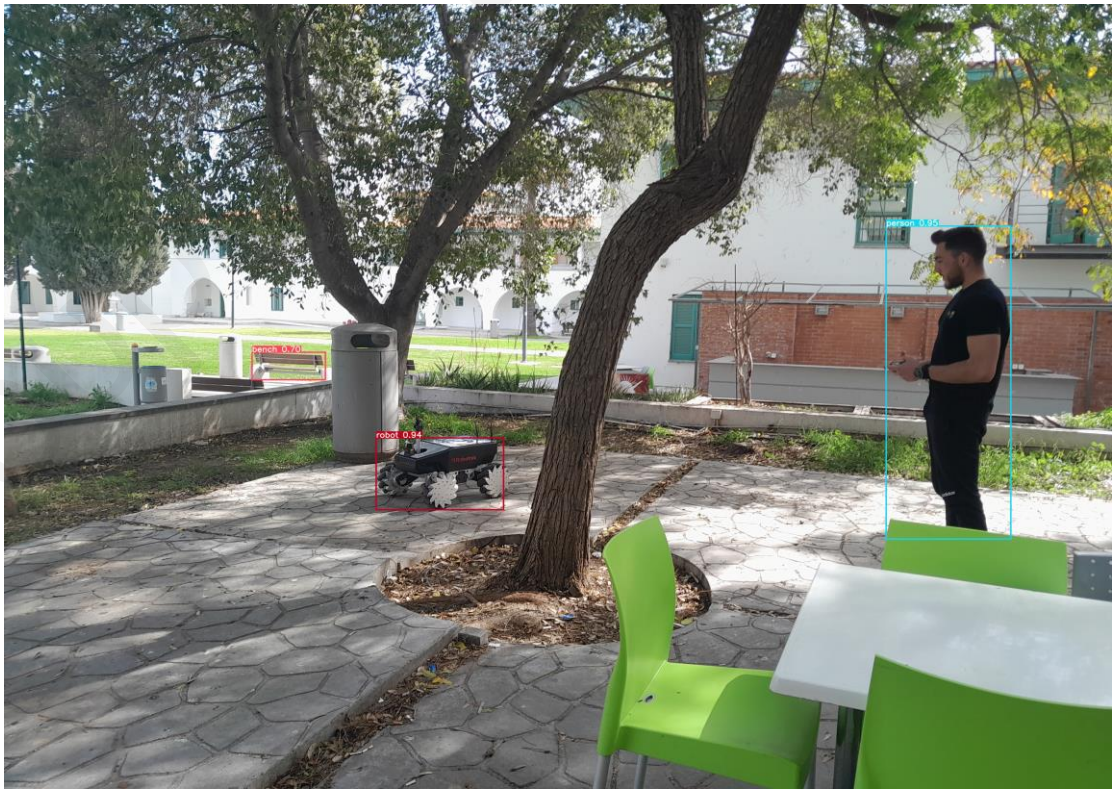


Εικόνα 3-14: Precision, Recall, mAP.

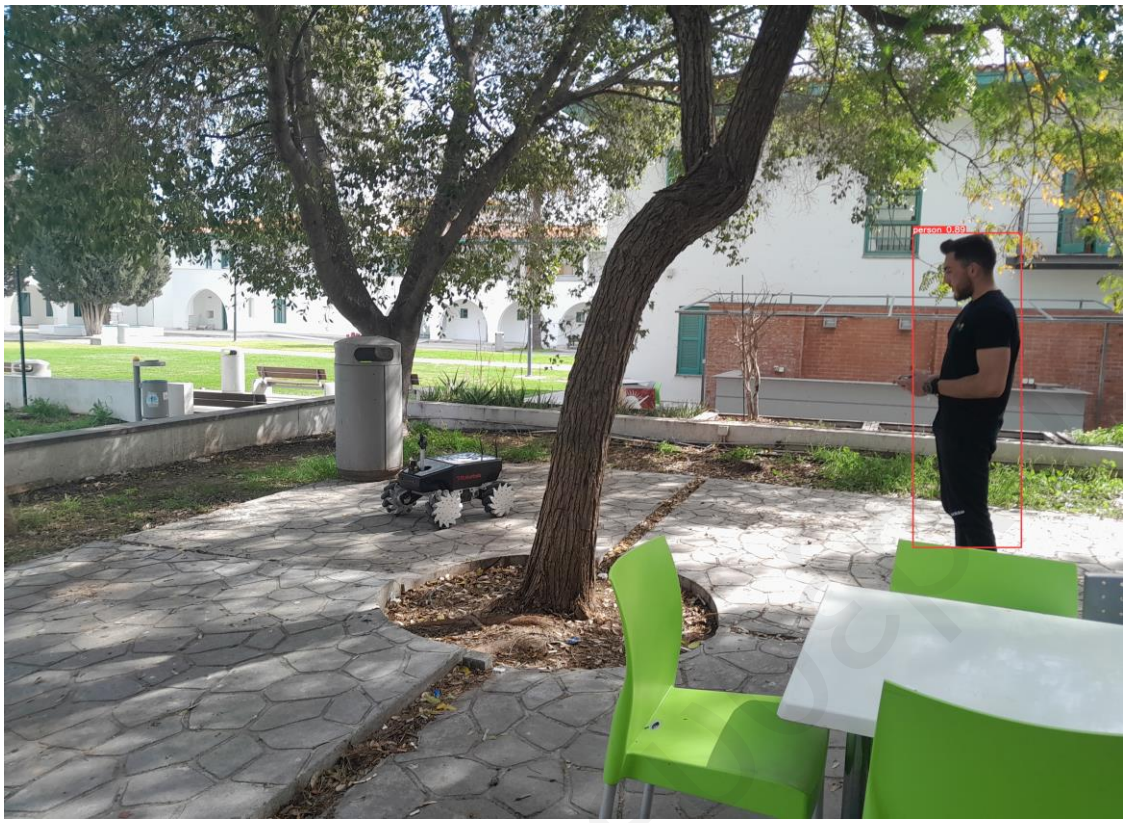
Διατηρώντας σταθερές τις παραμέτρους, έλαβαν χώρα δύο ανιχνεύσεις σε πραγματικό χρόνο για κάθε μοντέλο. Όσον αφορά την πρώτη ανίχνευση χρησιμοποιήθηκαν τα weights από τα runs που παρουσιάστηκαν πιο πάνω για 300 επαναλήψεις. Το YOLOv7 είναι καλύτερο από το YOLOv5 αφού ανιχνεύει ακόμη και το παγκάκι στο πίσω μέρος με μεγαλύτερη ακρίβεια, ωστόσο το YOLOv8 παρουσιάζει αρνητικό bounding box. Δεν αναγνώρισε το ρομπότ όπως φαίνεται στις Εικόνες 3-15 μέχρι 3-17. Σχετικά με το δεύτερο σύνολο αναγνωρίσεων (βλ. Εικόνες 3-18 με 3-20), αυτές έγιναν με weights που ανακτήθηκαν από 60 επαναλήψεις. Σε αυτή την περίπτωση το YOLOv8 υπερισχύει των υπολοίπων αφού είναι το μόνο μοντέλο που κατάφερε να εντοπίσει το ρομπότ.



Εικόνα 3-15: Αποτελέσματα YOLOv5 για 300 επαναλήψεις.



Εικόνα 3-16: Αποτελέσματα YOLOv7 για 300 επαναλήψεις.



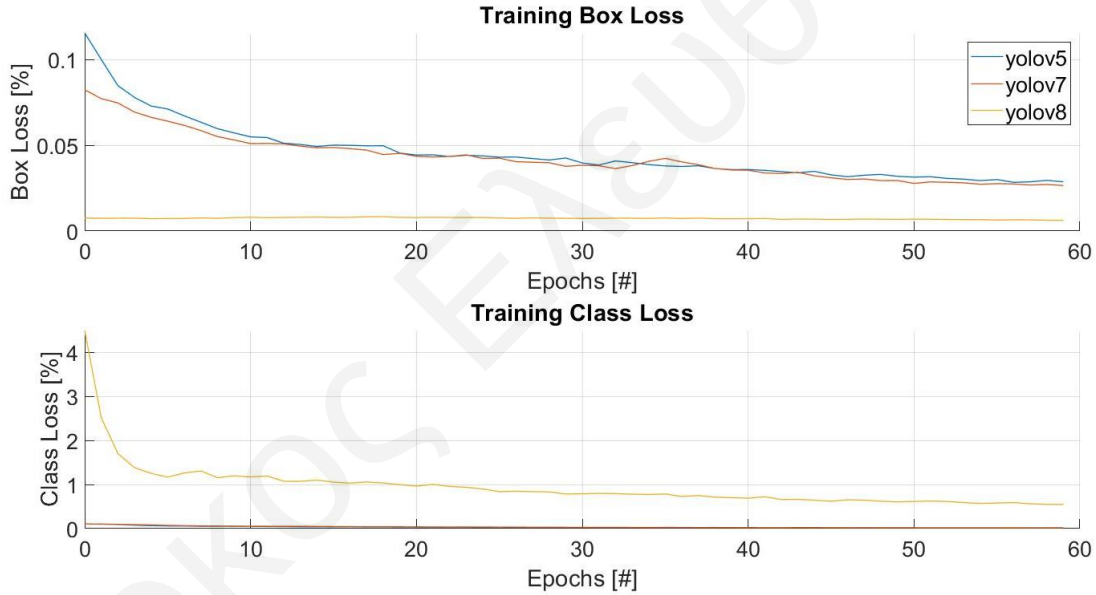
Εικόνα 3-17: Αποτελέσματα YOLOv8 για 300 επαναλήψεις.

Εικόνα 3-18: Αποτελέσματα YOLOv5 για 60 επαναλήψεις.

Εικόνα 3-19: Αποτελέσματα YOLOv7 για 60 επαναλήψεις.

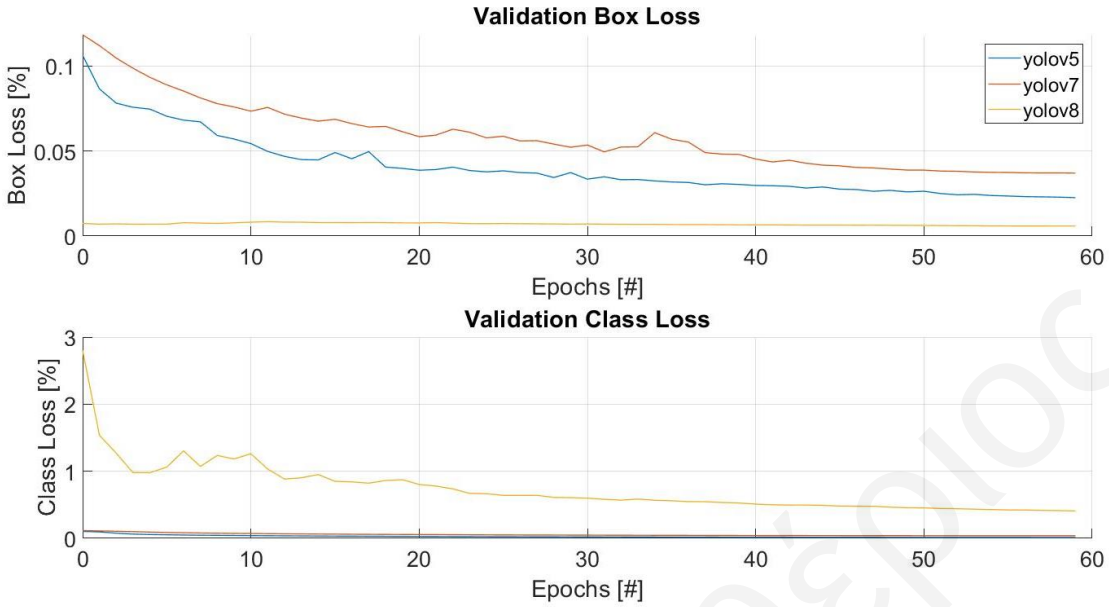
Εικόνα 3-20: Αποτελέσματα YOLOv8 για 60 επαναλήψεις.

Σε γενικές γραμμές όμως τα αποτελέσματα ήταν φτωχά, έτσι ακολούθησε αναδιαμόρφωση των hyperparameters με σκοπό τη βελτιστοποίηση των μοντέλων. Οι νέες τιμές των παραμέτρων παρουσιάζονται στο Παράρτημα V – Πίνακας Σύγκρισης Παραμέτρων YOLOv5 - YOLOv7 - YOLOv8, διαδοχικά για την κάθε έκδοση YOLO. Από τα αποτελέσματα που προέκυψαν σχετικά με το Training και Validation, το bounding box regression loss (βλ. Εικόνες 3-21 και 3-22) για τα YOLOv5 και YOLOv7 παρουσιάζει τον ίδιο τύπο καμπύλης με το δεύτερο να ξεκινά με 4% καλύτερη απόδοση. Παρόλο που το YOLOv8 έχει διαφορετικό τύπο καμπύλης, που πλησιάζει περίπου το 1% από τις πρώτες κιόλας επαναλήψεις, καταλήγει να έχει 2% καλύτερες επιδόσεις από το YOLOv7.



Εικόνα 3-21: Training Comparison (YOLOv5 – YOLOv7 – YOLOv8).

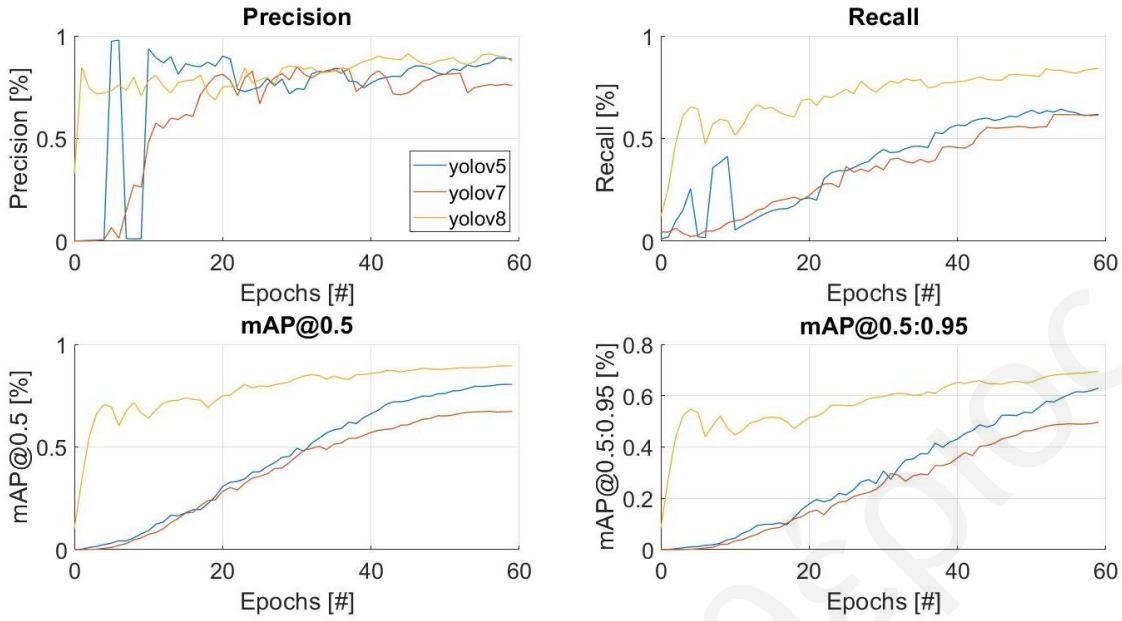
Σύμφωνα με τα αποτελέσματα του classification loss, οι καμπύλες των YOLOv5 και YOLOv7 ξεκινούν πολύ κοντά και καταλήγουν στο 1.7% και 2% αντίστοιχα. Παραδόξως το YOLOv8 έχει χειρότερη απόδοση ξεκινώντας με διαφορά 4.2 από το YOLOv7 και καταλήγοντας 0.52 περισσότερη απώλεια από τα άλλα δύο.



Εικόνα 3-22: Validation Comparison (YOLOv5 – YOLOv7 – YOLOv8).

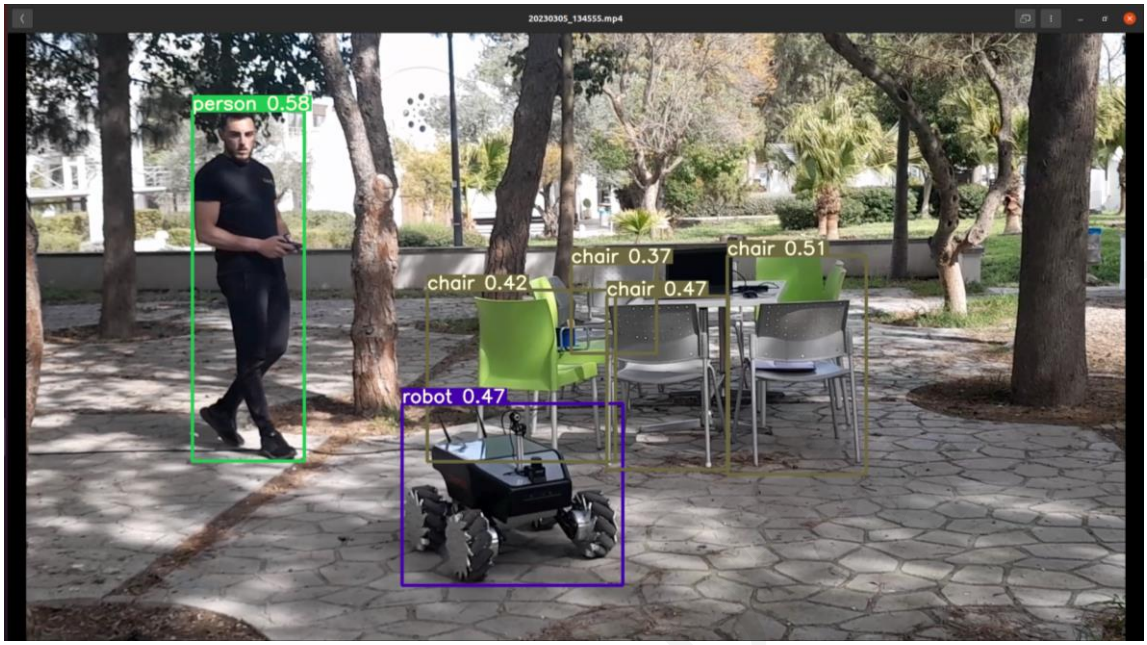
Κοιτάζοντας το γράφημα της ακρίβειας συναρτήσει των επαναλήψεων στην Εικόνα 3-23 είναι προφανές ότι όλα τα μοντέλα τείνουν να ξεπεράσουν το 90%. Πέραν των 50 επαναλήψεων παρουσιάζουν παρόμοια συμπεριφορά με το YOLOv7 να υπερέχει των υπολοίπων για ελάχιστο. Σε γενικές γραμμές αξίζει να σημειωθεί ότι η απόδοση του YOLOv8 μεταξύ των επαναλήψεων 0 και 50 υπερέχει κατά πολύ των υπολοίπων. Η διαφορά γίνεται πιο αισθητή στα διαγράμματα mean Average Precision, ωστόσο οι καμπύλες συγκλίνουν μετά την 50^{η} επανάληψη.

Από τα πιο πάνω αποτελέσματα προκύπτει ότι το YOLOv8 είναι καλύτερο από το YOLOv7, το οποίο υπερέχει του YOLOv5. Αυτό επιβεβαιώνεται και στον έλεγχο που διεξήχθη εφαρμόζοντας τα μοντέλα σε τυχαίο frame ενός βίντεο μερικών λεπτών κατά την χαρτογράφηση χρησιμοποιώντας το ρομπότ. Τα αποτελέσματα παρουσιάζονται διαδοχικά στις Εικόνες 3-24 με 3-26. Οι κώδικες για εκπαίδευση και ανίχνευση της κάθε έκδοσης παρατίθενται στο Παράρτημα ΙΙΙ.

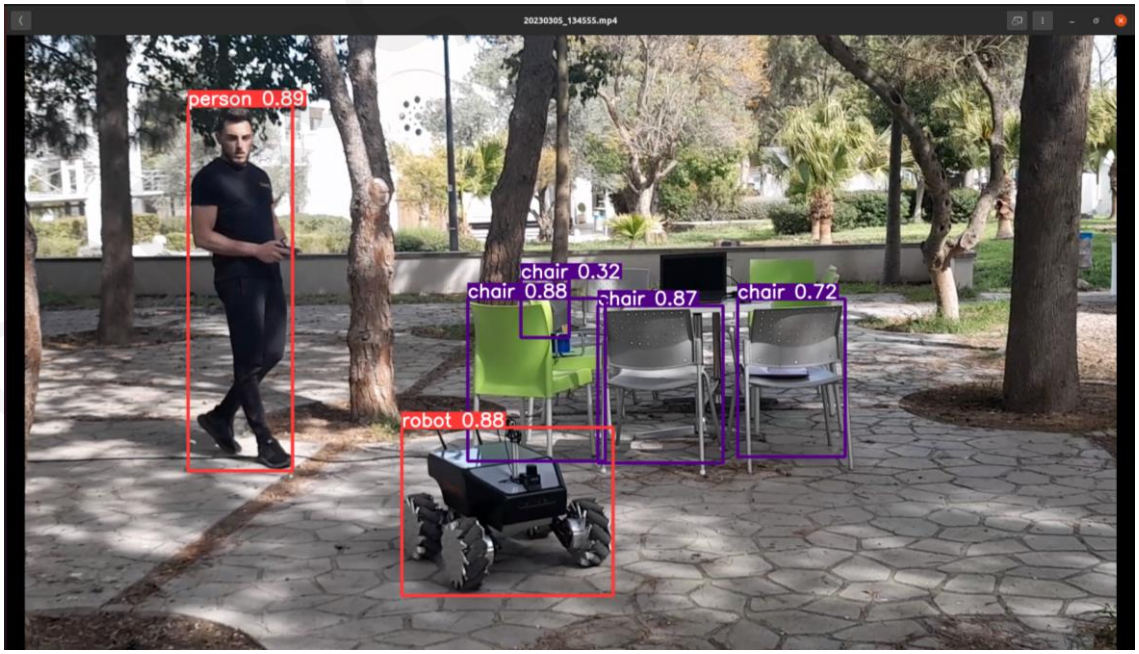


Εικόνα 3-23: Precision, Recall, mAP Comparison (YOLOv5 – YOLOv7 – YOLOv8).

Εικόνα 3-24: Έλεγχος YOLOv5.



Εικόνα 3-25: Έλεγχος YOLOv7.



Εικόνα 3-26: Έλεγχος YOLOv8.

Κεφάλαιο 4

4 ΚΑΤΗΓΟΡΙΟΠΟΙΗΣΗ

4.1 Ο κόσμος των Ρομπότ

Σε αυτό το Κεφάλαιο παρατίθενται οι κατηγορίες των συστημάτων που ήδη υφίστανται και εξυπηρετούν τον σκοπό για τον οποίο δημιουργήθηκαν, βασισμένα στην τεχνολογία αναγνώρισης των επιφανειών που επικεντρώνεται η παρούσα Διατριβή. Οι δύο βασικές κατηγορίες που χωρίζουν τα ρομπότ είναι τα ρομπότ εδάφους και τα ρομπότ αέρος (ιπτάμενα) και όπως μπορεί κανείς να συμπεράνει τα μεν περιορίζονται από την επαφή με το έδαφος, ενώ τα δε ίπτανται ως επί το πλείστων. Όσον αφορά την πρώτη ομάδα, αποτελείται τόσο από κινητά, όσο και από σταθερά ρομπότ. Η κατηγοριοποίηση αυτή βασίζεται εξ ολοκλήρου στο κατασκευαστικό μέρος του συστήματος και κατ' επέκταση στον τρόπο με τον οποίο ελίσσεται στον χώρο ενδιαφέροντος. Τα κινητά χωρίζονται με την σειρά τους σε τροχοφόρα, ερπυστριοφόρα κλπ. Σχετικά με την δεύτερη κατηγορία, αποτελείται σε σημαντικό βαθμό από drones. Ανάλογα με την ευελιξία, την σταθερότητα και την ακρίβεια που απαιτεί κάποια εφαρμογή, επιλέγεται αντίστοιχα και το κατάλληλο σύστημα για να υλοποιήσει τον στόχο της εφαρμογής.

Εικόνα 4-1: Ρομπότ από διάφορες κατηγορίες [25][26].

4.2 Κατηγοριοποίηση βάσει εφαρμογής

Τα συστήματα όμως, μπορούν ωστόσο να κατηγοριοποιηθούν ανάλογα και με την εφαρμογή για την οποία προορίζονται. Οι δύο βασικές κατηγορίες που απαρτίζουν τον τομέα της ασφαλείας είναι τα ρομπότ φρούρησης (security robots) και τα ρομπότ έρευνας και διάσωσης (search and rescue robots – S&R). Κάθε κατηγορία από τις δύο που μόλις ειπώθηκαν μπορούν να συμπεριλαμβάνουν ρομπότ εδάφους αλλά και αέρος.

Τα ρομπότ φρούρησης εστιάζουν σε συγκεκριμένο χώρο. Δηλαδή, έχουν εις γνώση τους ένα προκαθορισμένο χάρτη στον οποίο κινούνται και βασικός στόχος είναι η φύλαξη είτε κάποιου αντικειμένου, είτε κάποιου χώρου. Το ρομπότ κινείται στον χώρο και μόλις ανιχνεύσει ανθρώπινη φιγούρα ενημερώνει για την ύπαρξη ανθρώπου στον χώρο. Η ενημέρωση μπορεί να γίνει είτε με μήνυμα στην οθόνη ενός υπολογιστή, είτε με αυτόματη ενεργοποίηση κάποιου συναγερμού στον χώρο και ταυτόχρονη ενημέρωση κάποιου αρμόδιου προκειμένου να επιβληθεί του θέματος.

Εικόνα 4-2: Security Robot [27].

Από την άλλη, τα ρομπότ διάσωσης εργάζονται σε ένα πιο ευρύ περιβάλλον για τον εντοπισμό ανθρώπου. Αυτή είναι και μία από τις βασικές διαφορές των δύο κατηγοριών. Παρόλο που και τα δύο εργάζονται για τον εντοπισμό ανθρώπων, τα security robots δεν περιμένουν ότι θα συναντήσουν κάποιο άνθρωπο κατά τη διάρκεια της φρούρησής τους, ενώ τα S&R robots γνωρίζουν ότι κάπου στον χώρο ενδιαφέροντος τους υπάρχει ή πιθανόν να υπάρχει κάποιος άνθρωπος και προσπαθούν να τον εντοπίσουν. Μία άλλη βασική διαφορά, είναι ότι τα security robots δημιουργούν από μόνα τους τον χάρτη που θα χρειαστούν για να προσδιοριστεί η κίνησή τους, πριν τεθούν σε λειτουργία, είναι δηλαδή σαν να εξοικειώνονται με το περιβάλλον. Αντίθετα,

με τα S&R robots δεν είναι δυνατόν να ισχύσει κάτι τέτοιο από την στιγμή που δεν είναι γνωστός από πριν ο χώρος ενδιαφέροντος.

Ο τρόπος με τον οποίο λειτουργούν ποικίλει ανάλογα με την εφαρμογή και την κρισιμότητα ενός περιστατικού. Η μία εκδοχή που χαρακτηρίζεται ως ασφαλέστερη εκ των επιλογών, είναι η χαρτογράφηση περιοχών στις οποίες ενδέχεται να βρεθούν κάποτε τα ρομπότ για αναγνώριση. Η πιο πάνω διαδικασία ωστόσο δεν αποτελεί την πλέον πρακτική μέθοδο, εφόσον τα συστήματα περιήγησης θα πρέπει να ψηλαφίσουν από πριν κάθε περιοχή που μπορεί να ελλοχεύει κάποια απειλή. Με άλλα λόγια θα πρέπει να δημιουργηθεί ένα αρχείο στο οποίο να εμπεριέχονται όλοι οι χάρτες χωρισμένοι σε ομάδες ανά περιοχή για όλες τις δύσβατες περιοχές και περιοχές που εγκυμονούν κινδύνους στην μετακίνηση εντός των εν λόγω περιοχών. Οι πιο πάνω χάρτες θα πρέπει να έχουν δημιουργηθεί μέσω χαρτογράφησης από κάποιο ρομπότ με τρόπο ώστε ο κάθε χάρτης να μπορεί να αναγνωριστεί και από τα υπόλοιπα ρομπότ. Όπως υπάρχει η γκάμα με τα ρομπότ και σε κάθε περιστατικό επιλέγεται το κατάλληλο που θα φέρει σε πέρας την αποστολή στον κατάλληλο δυνατό βαθμό, έτσι θα υπάρχει και μία γκάμα με χάρτες της κάθε περιοχής, ώστε να επιλέγονται οι χάρτες σε μία περιοχή όπου πιθανό να βρίσκεται ο άνθρωπος που αναζητείται.

Μία δεύτερη πιο πρακτική εκδοχή είναι η εισαγωγή χάρτη από το Google Maps. Η συγκεκριμένη εξέλιξη ήρθε να λύσει τα χέρια όσων ασχολούνταν με το αντικείμενο αφού αποφεύγεται σε κάθε περίπτωση η χαρτογράφηση των περιοχών και οι χάρτες λαμβάνονται πλέον έτοιμοι. Παρόλα τα πλεονεκτήματα που μπορεί να παρουσιάζει η συγκεκριμένη εκδοχή – που δεν είναι λίγα – για ένα πράγμα έχει κερδίσει τον σεβασμό του συνόλου και έχει κεντρίσει το ενδιαφέρον αρκετών στελεχών του πληρώματος χειρισμού. Ο λόγος για την, ανά τακτικά χρονικά διαστήματα, αναβάθμιση των χαρτών. Δεν είναι μυστικό βέβαια ότι η μορφολογία του εδάφους αλλάζει με την πάροδο του χρόνου, κυρίως λόγω των καιρικών συνθηκών αλλά και της ανάπτυξης της βιοποικιλότητας. Θα αποτελούσε λοιπόν ζήτημα τεραστίων διαστάσεων η ανακρίβεια στον χάρτη, είτε με το να υπάρχουν στον χάρτη εμπόδια που δεν υφίστανται στην πραγματικότητα, είτε το αντίθετο. Ας μην ξεχνάμε ότι στις πλείστες των περιπτώσεων για να κρατήσεις ένα άνθρωπο στη ζωή μετά από τέτοιου είδους διασώσεις ο παράγοντας χρόνος είναι καθοριστικής σημασίας.

Εικόνα 4-3: Εναέρια φωτογραφία των εγκαταστάσεων του Πανεπιστημίου Κύπρου (Παράρτημα Καλλιπόλεως) [28].

4.3 Ρομπότ αέρος

Τον τελευταίο καιρό είχε ραγδαία ανάπτυξη η κατηγορία των ρομπότ αέρος ειδικά με την εμφάνιση των drones. Τα συγκεκριμένα συστήματα παρουσιάζουν το πλεονέκτημα ότι είναι αρκετά ευέλικτα στη χρήση, έτσι μπορεί με σχετική ευκολία να εξοικειωθεί κανείς μαζί τους. Δεν είναι τυχαίο άλλωστε που έχει γεμίσει η αγορά με διαφόρους τύπους σε ποικίλες μορφές και κόστος. Για ερευνητικούς σκοπούς ωστόσο, οι τιμές παραμένουν ψηλές, ανάλογα βέβαια και με την ποιότητα του προϊόντος, και ανεβαίνουν ακόμη περισσότερο καθώς προστίθενται τα διάφορα εξαρτήματα ανάλογα και με τις απαιτήσεις της εκάστοτε εφαρμογής. Ένα εξάρτημα στο οποίο αξίζει να γίνει αναφορά και προσδίδει σημαντικές δυνατότητες σε ένα σύστημα, είναι η θερμική κάμερα. Η χρωματική απεικόνιση των επιφανειών με βάση την θερμοκρασία τους συνεισφέρει αρκετά στον εντοπισμό ζωής σε ένα αφιλόξενο για τον άνθρωπο περιβάλλον.

Εικόνα 4-4: Λήψη από θερμική κάμερα σε drone [29].

Όπως όμως και κάθε συσκευή, που παρουσιάζει εύκολη εξοικείωση, η αλόγιστη χρήση από τον οποιονδήποτε αποτελεί κίνδυνο για το συλλογικό καλό. Με γνώμονα αυτό λοιπόν η κυβέρνηση και δη οι αρμόδιες υπηρεσίες προέτρεξαν να κατοχυρώσουν ένα σύνολο κανόνων και νομοθεσιών το οποίο οφείλει να γνωρίζει ο καθένας προτού θέσει σε λειτουργία την ιπτάμενη συσκευή. Η διαδικασία λοιπόν προκειμένου να εξασφαλίσει κάποιος άδεια χειρισμού drone δεν απέχει και πολύ από την διαδικασία για εξασφάλιση διπλώματος οδήγησης. Σε κάθε περίπτωση, ο χειρισμός ιπτάμενης συσκευής αποτελεί τέχνη, όπως άλλωστε και η οδήγηση προκειμένου να συντονιστεί η όποια δράση με το σύνολο, ώστε να συλλειτουργούν χωρίς επιπλοκές (βλ. Εικόνα 4-5).

Εικόνα 4-5: Νομοθεσία για drone [30].

Κεφάλαιο 5

5 ΑΙΣΘΗΤΗΡΕΣ ΤΟΥ ΡΟΜΠΟΤ

5.1 Αισθητήρες ανίχνευσης χώρου

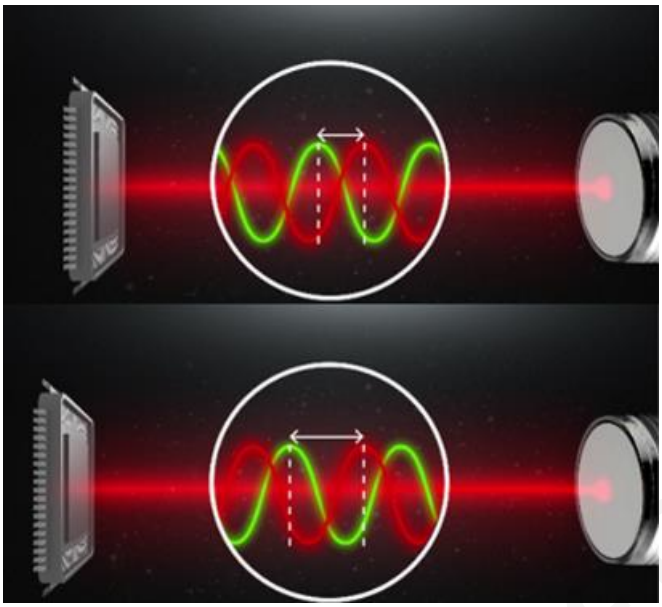
Όπως ειπώθηκε και στο προηγούμενο κεφάλαιο, η κάθε εφαρμογή για την οποία προορίζεται ένα σύστημα απαιτεί ένα σύνολο εξαρτημάτων τα οποία χρειάζεται να ρυθμιστούν ώστε η λειτουργία τους να προσδίδει το επιθυμητό αποτέλεσμα. Η λειτουργία τους μπορεί να είναι ταυτόχρονη, διαδοχική ή και τα δύο μαζί. Η εν λόγω θέση επιδιώκει τη βέλτιστη ρύθμιση των εξαρτημάτων ώστε να προκύψει το καλύτερο δυνατό αποτέλεσμα. Για να επέλθει όμως η απόλυτη συνεργασία μεταξύ των στοιχείων ενός συστήματος, θέμα μείζονος σημασίας αποτελεί η κατάλληλη επιλογή και έπειτα η ορθή ρύθμιση των παραμέτρων κάθε στοιχείου. Στην αγορά τη δεδομένη χρονική στιγμή υπάρχει πληθώρα από σύνολα εξαρτημάτων τα οποία στελεχώνουν τα διάφορα συστήματα. Έπειτα από σχολαστική έρευνα σε αρκετά από αυτά, η παρούσα βιβλιογραφική μελέτη εστιάζει σε ένα συγκρότημα εξαρτημάτων το οποίο συναντάται σε προϊόντα σημαντικών εταιρειών με μία από αυτές την Robotnik με έδρα την Βαλένθια, Ισπανία (βλ. Εικόνα 5-1).

Εικόνα 5-1: Robotnik offices [25].

Στην αγορά θα συναντήσει κανείς πολλούς αισθητήρες καθ΄ ένας από τους οποίους εξοπλίζει το σύστημα με διαφορετικές δυνατότητες. Ακόμη και αισθητήρες που προορίζονται για τον ίδιο σκοπό δύναται να προωθήσουν στον χρήστη συλλογή δεδομένων η οποία διαφέρει, από εξάρτημα σε εξάρτημα. Για σκοπούς Διατριβής παρουσιάζονται στην συνέχεια ορισμένοι αισθητήρες που κυκλοφορούν αυτή την στιγμή στην αγορά και επισυνάπτεται περιγραφή του τρόπου λειτουργίας, καθώς και μερικές σημαντικές προδιαγραφές. Η έρευνα επικεντρώθηκε σε εξοπλισμό το κόστος του οποίου ανέρχεται σε μερικές χιλιάδες ευρώ. Παρά το ότι υπήρχαν και συστήματα με εξαιρετικά αυξημένο κόστος και κατ' επέκταση αποδόσεις, η επιλογή έγινε ώστε η παρούσα Διατριβή να αποτελεί και ενδεικτικά παραδείγματα για εταιρείες ασφαλείας σύμφωνα με τα δεδομένα της Κυπριακής αγοράς. Τονίζεται ότι η υλοποίηση συστήματος με την αλληλεπίδραση των πιο κάτω εξαρτημάτων, μπορεί να εξυπηρετήσει τους σκοπούς που αναφέρθηκαν στην εισαγωγή με σχετικά χαμηλό κόστος και καλή ακρίβεια. Ο πιο κάτω κατάλογος λοιπόν αποτελεί την χρυσή τομή όσον αφορά το τρίπτυχο κόστος, ακρίβεια και ευελιξία.

5.1.12D laser sensor

Η αρχή λειτουργίας του αισθητήρα laser, κάθε άλλο παρά περίπλοκη μπορεί να χαρακτηριστεί. Ο αισθητήρας μέσω της δέσμης επιτυγχάνει τη διάδοση ενός διαμορφωμένου φωτεινού κύματος προς τον στόχο. Ακολουθεί ανάκλαση προς τον αισθητήρα και οι φάσεις των δύο σημάτων συγκρίνονται. Από την διαφορά φάσης των δύο υπολογίζεται ο χρόνος που χρειάζεται ώστε η δέσμη να ολοκληρώσει τη διαδρομή, προς και από τον στόχο, έτσι προσδιορίζεται η απόσταση μεταξύ των δύο επιφανειών (βλ. Εικόνα 5-2). Η τεχνολογία του αισθητήρα laser έχει βρει τεράστια ανταπόκριση σε πολλούς τομείς, ενώ ταυτόχρονα έχει εδραιωθεί για αρκετές εφαρμογές λόγω της αυξημένης ακρίβειας που παρέχει. Ανάλογα με την ποιότητα του αισθητήρα αλλά και το πόσο προσεκτικά γίνεται η σάρωση μίας επιφάνειας υπάρχει η δυνατότητα αποτύπωσης των ατελειών στις επιφάνειες μέχρι και σε κλίμακα χιλιοστού. Για τη συγκεκριμένη εφαρμογή θα αποτελούσε υπερβολή βέβαια, ενώ θα είχε κόστος σε χρόνο, χρήμα και ενέργεια χωρίς ουσιαστικό λόγο.

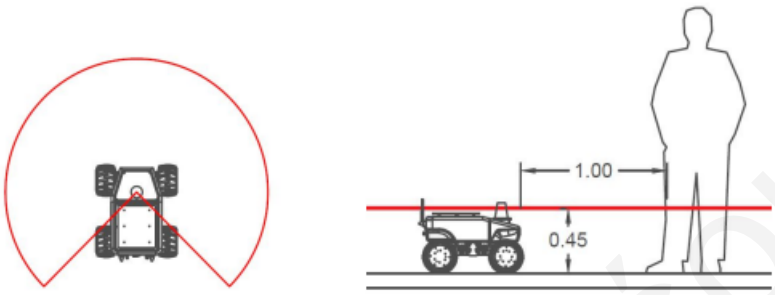


Εικόνα 5-2: Υπολογισμός απόστασης με laser sensor [31].

Ο αισθητήρας laser δύο διαστάσεων αποσκοπεί κατά κύριο λόγο στον εντοπισμό εμποδίων. Αυτό εξυπηρετεί τόσο το ρομπότ όσο και τον άνθρωπο. Μέσω του συγκεκριμένου αισθητήρα εξασφαλίζεται η ακεραιότητα της κινητής συσκευής εφόσον αποφεύγονται τυχόν συγκρούσεις με εμπόδια που αφίχθηκαν στον χώρο μετά την δημιουργία του χάρτη. Παράλληλα, μετά τον εντοπισμό του ανθρώπου, ο αισθητήρας επεξεργάζεται την ακριβή τοποθεσία του στον χώρο, ακόμη και εν κινήσει προστατεύοντάς τον. Σε καμία περίπτωση δεν είναι επιθυμητή η επαφή ανθρώπου – ρομπότ και αν αναλογιστεί κανείς ότι ρομπότ τέτοιου βεληνεκούς κυμαίνονται στα 50Kg και αναπτύσσουν μέχρι και $3^{m}/_{s}$, σε τυχόν σύγκρουση με τον άνθρωπο με αντίστοιχη ορμή μπορεί να προκαλέσει σοβαρούς τραυματισμούς.

Κάπου εδώ καλό θα ήταν να αναφερθεί ότι οι προδιαγραφές του αισθητήρα επιτρέπουν τη σάρωση στον χώρο μόνο σε δύο κατευθύνσεις αποτυπώνοντας ουσιαστικά στον χάρτη μόνο ένα επίπεδο του χώρου. Το επίπεδο που θα αποτυπωθεί εναπόκειται στην θέση που θα τοποθετηθεί ο αισθητήρας επί του ρομπότ. Η απόσταση από το έδαφος και η φορά με την οποία θα είναι στραμμένο θα καθορίσουν το εύρος που θα εξετάζει ανά πάσα στιγμή καθ’ όλη την διάρκεια της λειτουργίας. Οι πιο συνήθης εφαρμογές παρουσιάζουν τον αισθητήρα τοποθετημένο στο άνω μέρος της συσκευής (~30mm από το έδαφος) και στραμμένο με φορά τέτοια ώστε το τυφλό σημείο να είναι συμμετρικά η πίσω πλευρά του ρομπότ (βλ. Εικόνα 5-3). Σε περιπτώσεις όπου η οπτική γωνία των 270° δεν καλύπτει τις ανάγκες για τις οποίες προορίζεται το σύστημα, τοποθετούνται δύο αισθητήρες στην ίδια ευθεία αντίρροπα,

ώστε να υφίσταται πλήρης κάλυψη στο επίπεδο. Σε αντίστοιχη περίπτωση, οι δύο αισθητήρες πρέπει να συνεργάζονται ώστε τα αντικείμενα στο κοινό τους πεδίο κάλυψης να αντιμετωπίζονται σαν ένα σώμα. Δεν τίθεται τόσο θέμα ακρίβειας, όσο υπολογιστικό αφού θα είναι σαν να υπάρχουν δύο χάρτες ο ένας πάνω στον άλλο.



Εικόνα 5-3: 2D laser sensor range [25].

Κατά κύριο λόγο η εφαρμογή επιδιώκει την κάλυψη οριζοντίου επιπέδου και με μία πλήρη περιήγηση στον χώρο ανακτάται ο χάρτης της περιοχής ενδιαφέροντος. Με το πέρασμα του ρομπότ από κάποιο σημείο, ανιχνεύει τα διάφορα αντικείμενα και αποθηκεύει τα δεδομένα σύμφωνα με ένα αρχικό σύστημα συντεταγμένων ώστε να παραμένει γνωστό που είναι το καθετί σκιαγραφώντας με αυτό τον τρόπο τον χάρτη.

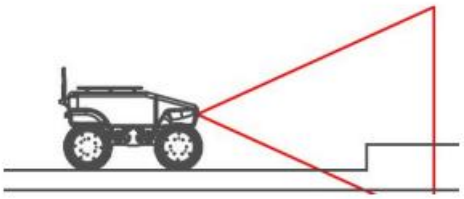
Παρόλο που ο συγκεκριμένος αισθητήρας παρέχει πολλές δυνατότητες, η αποκλειστική χρήση του δεν μπορεί να οδηγήσει σε ένα πλήρες αποτέλεσμα. Χωρίς να προδιαγράφεται απαραίτητη η χρήση ενός δεύτερου αισθητήρα, ορισμένες χωρικές παράμετροι δημιουργούν κάποιες ανησυχίες ως προς την αποτελεσματικότητα της αποτύπωσης του περιβάλλοντος σε ένα επίπεδο. Η ύπαρξη αντικειμένων εκατέρωθεν του επιπέδου πιθανόν να προκαλέσει προβλήματα τόσο στο ρομπότ, όσο και στις εσωτερικές ηλεκτρονικές εγκαταστάσεις, μετά από τυχόν σύγκρουση. Στην Εικόνα 5-4 για παράδειγμα, ο αισθητήρας θα αναγνωρίσει μόνο τον κορμό της καρέκλας που βρίσκεται στο επίπεδο που σαρώνει. Με αυτό τον τρόπο δύναται ο κίνδυνος σύγκρουσης με το πάνω μέρος του εμποδίου. Με κάποιο τρόπο δηλαδή ο υπολογιστής πρέπει να γνωρίζει τι ακριβώς βρίσκεται σε κάθε σημείο του χώρου, τουλάχιστον μέχρι το άνω άκρο του ρομποτικού συστήματος, ώστε να εξασφαλίζεται η ακεραιότητα της συσκευής. Λύση στο εν λόγω ζήτημα δίνει ο αισθητήρας που προδιαγράφεται στη συνέχεια.

Εικόνα 5-4: Εμπόδια εκτός εμβέλειας αισθητήρα 2D Laser Sensor.

5.1.23D camera sensor

Η αρχή λειτουργίας του συγκεκριμένου αισθητήρα είναι ελάχιστα πιο σύνθετη από του προηγούμενου και έχει ως εξής: η κάμερα εκπέμπει ένα μοτίβο IR το οποίο προβάλλεται στον χώρο. Στη συνέχεια το ίδιο μοτίβο ανιχνεύεται από μία κάμερα IR και το βάθος του μοτίβου υπολογίζεται με τριγωνισμό (Smisek et al., 2013 [60]). Όπως και προηγουμένως όμως, η βασική ιδέα παραμένει η ίδια. Το όποιο σήμα εκπέμπεται από τη συσκευή, στην οποία ενσωματωμένος βρίσκεται και ο δέκτης ώστε να το περισυλλέξει και ακολούθως προωθείται για επεξεργασία.

Ο αισθητήρας αυτός παρέχει μία διαφορετική κάλυψη του χώρου όπως φαίνεται και στην Εικόνα 5-5. Το γεγονός ότι εστιάζει περιορισμένα σε ένα τμήμα με εκπομπή μοτίβου ορθογώνιας διατομής δεν αποτελεί σημαντικό εμπόδιο, αφού η ελευθερία στις μετακινήσεις της συσκευής επιτρέπει την πλήρη κάλυψη του χώρου. Η παρουσίαση του χειρισμού ρομπότ στον χώρο ως τέχνη, που έγινε στην εισαγωγή, δεν είναι σε καμία περίπτωση τυχαία. Περιορισμοί όπως ο πιο πάνω μπορούν να αντιμετωπιστούν από τον χειριστή του συστήματος και το πόσο ικανός είναι σε αυτό θα επηρεάσει αυτόματα και το αποτέλεσμα. Σε περίπτωση που ο χειριστής αδυνατεί να υλοποιήσει τη χαρτογράφηση με τα συγκεκριμένα εργαλεία και τους περιορισμούς που προδιαγράφουν, δύναται να το κάνει με διαφορετικό συγκρότημα εξαρτημάτων που αφενός θα έχει λιγότερους περιορισμούς και αφετέρου θα εκτινάξει το κόστος αγοράς.



Εικόνα 5-5: 3D camera sensor [25].

Από τα σημεία που αξίζει να γίνει αναφορά, σχετικά με τις RGB κάμερες, είναι η λειτουργία υπό οποιοδήποτε επίπεδο φωτισμού. Από την άλλη αδυνατούν να προσδιορίσουν την ακριβή απόσταση σωμάτων που πλησιάζουν περισσότερο από το προκαθορισμένο διάστημα που ανέρχεται κοντά στα 40mm. Παρόμοιου τύπου περιορισμοί δεν περνάνε απαρατήρητοι. Παραταύτα, δεν επηρεάζουν το τελικό αποτέλεσμα, αφού κατά τη διάρκεια είτε χαρτογράφησης είτε εξερεύνησης προηγείται κάλυψη της περιοχής με ακτίνα 40mm, προτού αυτή προσέλθει στην τυφλή για τον αισθητήρα περιοχή.

5.2 Φωτεινές ενδείξεις και ηχητικά σήματα ασφαλείας

Ο τομέας της ασφάλειας επιβάλλεται να αποτελεί την No. 1 προδιαγραφή λειτουργίας του συστήματος. Στον κόσμο των ρομπότ αυτό εξυπηρετείται εν μέρει με ακουστικούς και φωτεινούς δείκτες. Ανάλογα με την λειτουργία τους, ο χειριστής αντιλαμβάνεται το καθεστώς που επικρατεί στην περιοχή που κεντρίζει το ενδιαφέρον για εξερεύνηση ή φρούρηση. Οι ενδείξεις επιτρέπουν στους παρατηρητές να ενημερώνονται καθ’ όλη την διάρκεια της επιχείρησης. Παράλληλα, αναφέρουν τυχόν προβλήματα στην λειτουργία της συσκευής και προειδοποιούν για συνθήκες που αποτελούν εμπόδιο ώστε να φέρουν σε πέρας την αποστολή. Κατά την περιήγηση του ρομπότ σε ένα περιβάλλον ενδέχεται να βρεθεί αντιμέτωπο με διάφορες επιπλοκές.

Σκοπός είναι με κάποιο πρακτικό τρόπο να γίνεται η ακριβής ενημέρωση με όσο το δυνατό λιγότερα εξαρτήματα. Η επιχειρησιακή αλληλεπίδραση ρομπότ – χειριστή κατέστη εφικτή μέσω ενός κωδικοποιημένου συνόλου ενδείξεων. Σε πρώτο στάδιο έχουν ομαδοποιηθεί οι συνθήκες που ενδέχεται να απασχολήσουν το ρομπότ κατά την λειτουργία. Οι κατηγορίες προέκυψαν σύμφωνα με τον βαθμό επικινδυνότητας κάθε κατάστασης και οι ενδείξεις προσαρμόστηκαν ανάλογα. Παρόλο που αυτό ακούγεται

απλό, σε περιπτώσεις έκτακτης ανάγκης δεν θα ήταν τόσο χρήσιμο για τυπικές προειδοποιήσεις να υφίσταται η ίδια ένδειξη με περιπτώσεις που χρίζουν υψηλής σημασίας. Επίσης, συχνά είναι προτιμότερο η ένδειξη να παρουσιάζεται είτε στο κέντρο ελέγχου του ρομπότ, είτε στην συσκευή ανάλογα με την εφαρμογή. Για παράδειγμα δεν θα εξυπηρετούσε σε κάτι εάν με το που ανιχνευόταν ο άνθρωπος σε μία επιχείρηση εντοπισμού να ηχήσει ενημερωτική ένδειξη στο ρομπότ, αντίθετα πολύ πιθανόν να τον τρόμαζε. Από την άλλη, κατά την διάρκεια φρούρησης ενός χώρου, με τον εντοπισμό ανθρώπου πιθανόν να ήταν πιο εξυπηρετικό εάν δινόταν σε αυτό να καταλάβει ότι έχει γίνει αντιληπτός.

Εικόνα 5-6: Acoustic and Light Indicators [32].

5.2.1Ακουστικοί δείκτες

Η διαμόρφωση των ήχων και των φωτεινών σημάνσεων δύναται να προσαρμόζεται ανάλογα με την εργασία του ρομπότ. Μερικές από τις πιο συνήθεις πληροφορίες που μεταδίδουν ακουστικές ενδείξεις είναι ο διακοπτόμενος ήχος με δεδομένη συχνότητα και ο συνεχής ήχος προκαθορισμένης έντασης. Ο διακοπτόμενος ήχος ενημερώνει ότι η συσκευή παρόλο που λαμβάνει κάποια εντολή, δεν λειτουργεί λόγω του ότι εντοπίζει κάποιο εμπόδιο. Ο συνεχής ήχος για δεδομένο αριθμό δευτερολέπτων ενημερώνει ότι το ρομπότ έχει αρχίσει να μετακινείται προειδοποιώντας για λόγους ασφαλείας.

Εικόνα 5-7: Acoustic Indicator [33].

5.2.2Φωτεινοί δείκτες

Για τις πιο σημαντικές καταστάσεις το ρομπότ προειδοποιεί με ήχο, ενώ για προειδοποιητικά σήματα με διαμόρφωση χρωμάτων. Αυτό κυρίως γιατί είναι πιο πιθανόν ο χειριστής να μην παρατηρήσει κάποια φωτεινή ένδειξη, από το να μην την ακούσει. Ο τρόπος λειτουργίας των φωτεινών ενδείξεων είναι ίδιος με αυτόν των ηχητικών. Ανάλογα με τη σοβαρότητα της ένδειξης προκαθορίζεται η διαμόρφωση των χρωμάτων, το χρονικό διάστημα που βρίσκεται σε λειτουργία, η συχνότητα για διακοπτόμενη λειτουργία και η ένταση του εκπεμπόμενου φωτός.

Εικόνα 5-8: Light Indicators [34].

5.3 PTZ camera

Η Pan Tilt Zoom κάμερα αποτελεί βασικό στοιχείο για την αναμετάδοση εικόνας κατά τις επιχειρήσεις διάσωσης και φρούρησης. Είναι ένα εύχρηστο εργαλείο που έχει την δυνατότητα τηλεχειρισμού της κατεύθυνσης του φακού και μεγέθυνσης. Μέσω μίας κατασκευαστικά απλής βάσης ενσωματώνεται στην κατασκευή σε θέση που

να εξυπηρετεί τους σκοπούς της εφαρμογής. Η εφαρμογή της μπορεί είτε να περιοριστεί αποκλειστικά για τον τηλεχειρισμό του ρομπότ επιτρέποντας την οπτική επαφή με το περιβάλλον, είτε για σκοπούς λήψης εικόνων από τον χώρο ενδιαφέροντος. Οι εικόνες που λαμβάνονται προωθούνται στον αλγόριθμο χωρίς κάποια επεξεργασία, κάτι που μειώνει τη χρονική απόκλιση από την στιγμή της λήψης έως την στιγμή της αναγνώρισης.

Εικόνα 5-9: PTZ camera [25].

5.4 Ultrasonic Sensor

Οι αισθητήρες υπερήχων έχουν σκοπό τον υπολογισμός της απόστασης και όπως έχει ειπωθεί για τους αισθητήρες που προηγήθηκαν, η βασική αρχή λειτουργίας είναι η αποστολή και επανασυλλογή σήματος, από την επεξεργασία του οποίου θα προκύψει το ζητούμενο αποτέλεσμα. Το μέσο μεταφοράς για την υλοποίηση της μέτρησης είναι τα υπερηχητικά κύματα, όπως άλλωστε υποδηλώνει και η ονομασία τους. Η τεχνολογία των υπερηχητικών κυμάτων παρουσιάζει το μεγάλο πλεονέκτημα της ανάκλασης σε διαφανή επιφάνειες.

Εικόνα 5-10: Ultrasonic Sensor [35].

Ο πομπός εκπέμπει υπερηχητικό κύμα προς την επιφάνεια του στόχου. Αυτό με τη σειρά του ανακλάται από την επιφάνεια και περισυλλέγεται εκ νέου από τον αισθητήρα. Το χρονικό διάστημα που μεσολαβεί μεταξύ εκπομπής και λήψης υπολογίζεται από τον αισθητήρα και στη συνέχεια χρησιμοποιείται ώστε να υπολογιστεί η απόσταση που χωρίζει τον αισθητήρα από τον στόχο.

Σε αντίθεση με τους οπτικούς αισθητήρες, οι αισθητήρες υπερήχων ανακλαστικού μοντέλου παρουσιάζουν μόνο ένα ταλαντωτή ο οποίος εναλλάξ εκπέμπει και λαμβάνει κύματα. Αυτό ελαττώνει σημαντικά το μέγεθος της κεφαλής του αισθητήρα. Η μαθηματική εξίσωση που αποτυπώνει την απόσταση είναι η εξής:

Εικόνα 5-11: Υπολογισμός Απόστασης.

$$L = \frac{1}{2} * T * C$$

όπου: L η απόσταση αισθητήρα – στόχου

T το χρονικό διάστημα μεταξύ εκπομπής και λήψης

C η ταχύτητα διάδοσης του κύματος

Σημειώνεται ότι η απόσταση πολλαπλασιάζεται με 0.5 εφόσον ο χρόνος που υπολογίζεται αντιστοιχεί από την στιγμή που εκπέμπεται το σήμα έως τη στιγμή που ανιχνεύεται ξανά.

Κεφάλαιο 6

6 ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΟΥ ΡΟΜΠΟΤ

6.1 Επιλογή πορείας

Σύμφωνα με το κάθε περιστατικό, τίθεται σε εφαρμογή και το αντίστοιχο σχέδιο δράσης. Στις πλείστες όμως των περιπτώσεων, για οικονομικούς κυρίως λόγους, εξασφαλίζεται ένα ρομποτικό σύστημα για να εξυπηρετεί περισσότερο από ένα σκοπό. Ιδιωτικές εταιρείες σε Ευρώπη και Αμερική επιστρατεύουν τον εξοπλισμό τις βραδινές ώρες ώστε να επιβλέπει εγκαταστάσεις ενώ την υπόλοιπη μέρα βρίσκεται σε ετοιμότητα για κάλυψη έκτακτων περιστατικών. Προηγείται κατάλληλη διαμόρφωση του εξοπλισμού στο σύστημα και βρίσκεται έτοιμο να ανταποκριθεί σε κάθε κάλεσμα ανά πάσα στιγμή.

Ανεξαιρέτως με την ανάθεση καθηκόντων, υπάρχει ένα σχέδιο δράσης το οποίο αρχικά ισχύει το ίδιο για τις πλείστες των περιπτώσεων και έπειτα διαμορφώνεται σύμφωνα με τον σκοπό λειτουργίας, που καλείται το ρομπότ να εξυπηρετήσει. Πρώτα απ' όλα πρέπει να γίνει χαρτογράφηση του χώρου ώστε να δημιουργηθεί ο χάρτης στον οποίο θα κινηθεί το ρομπότ. Στη συνέχεια θα ακολουθήσει αρχικοποίηση της συσκευής, να ενημερωθεί δηλαδή για το που ακριβώς βρίσκεται στον χάρτη. Έπειτα σκιαγραφείται η πορεία που θα ακολουθήσει στον χώρο και τέλος πλοήγηση. Για τις επιχειρήσεις έρευνας και διάσωσης, απαιτείται συνεχής επίβλεψη ώστε να υπάρχει άμεση ανταπόκριση και καθοδήγηση.

Οι εφαρμογές ασφαλείας χωρίζονται σε δύο μεγάλες κατηγορίες, τις εφαρμογές που γίνονται με σκοπό τη φρούρηση και φύλαξη χώρου ή εξοπλισμού και τις εφαρμογές διάσωσης. Κύρια διαφορά των δύο, όπως έχει ειπωθεί και πρωτύτερα, είναι ότι στη φρούρηση, νοουμένου ότι όλα κυλούν ομαλά, δεν θα εντοπιστεί άνθρωπος. Αντίθετα στη διάσωση, η επιχείρηση ολοκληρώνεται επιτυχώς με τον εντοπισμό του ανθρώπου.

6.2 Πορεία για εφαρμογή Φρούρησης

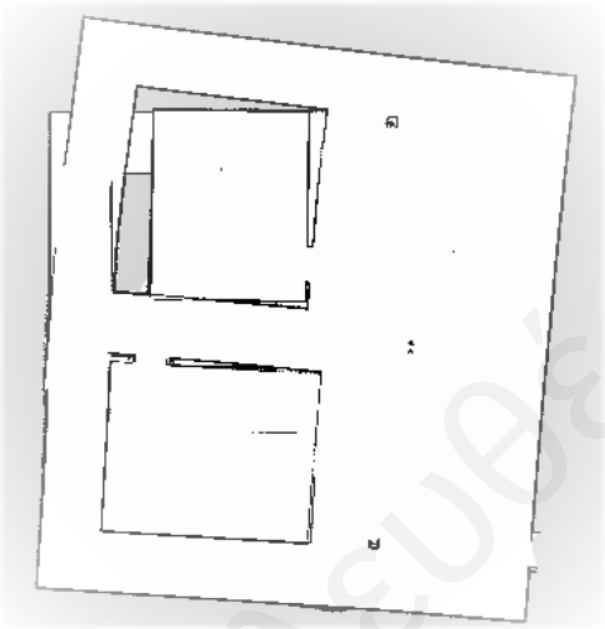
Η φύλαξη μίας περιοχής θα μπορούσε να πει κανείς ότι είναι από τις κατάλληλες αποστολές για ένα ρομπότ. Υπάρχουν μερικά βασικά βήματα, τα οποία εάν ακολουθηθούν σωστά είναι βέβαιο ότι θα προκύψει το επιθυμητό αποτέλεσμα. Δεν υπάρχει βιασύνη, τα πάντα σχεδιάζονται και εκτελούνται προσεκτικά, ενώ σε περίπτωση λάθους υπάρχει η δυνατότητα επανάληψης της διαδικασίας χωρίς ιδιαίτερο κόστος. Παράλληλα δεν απαιτείται κριτική σκέψη για την λήψη αποφάσεων. Τα πάντα είναι τυποποιημένα και εκτελούνται βάσει πρωτοκόλλων. Το μόνο που απαιτείται είναι ο σωστός προγραμματισμός. Απ' εκεί και πέρα είναι στην κρίση της κάθε εταιρείας, στην οποία ανήκει ο χώρος, για το πως θα εργαστεί σε τυχόν περιστατικό.

6.2.1Χαρτογράφηση (Mapping)

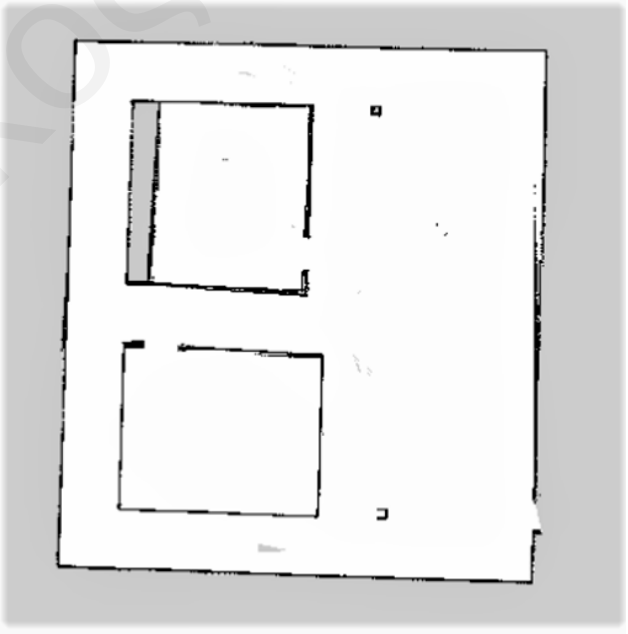
Το ρομπότ έχει όλο τον χρόνο να περιηγηθεί στον χώρο. Λόγω της εμβέλειας των αισθητήρων, αρκεί και μόνο ένα πέρασμα προκειμένου να εξασφαλίσει πληροφορίες για την εκάστοτε απόσταση που διένυσε αλλά και κάθε σημείο σε ακτίνα 1000mm απ' όπου πέρασε. Εκτελεί αρκετές διαδρομές μέχρις ότου να δημιουργηθεί ολοκληρωμένος χάρτης της περιοχής. Δεν είναι απαραίτητο να επαναλάβει την ίδια διαδρομή, ωστόσο συνηθίζεται να γίνεται. Με την επαναπροσέγγιση μιας περιοχής το ρομπότ αντιμετωπίζει ξανά τα ίδια στοιχεία στον χώρο. Δεν τα αναγνωρίζει από την μορφή τους αλλά επικαλείται το σημείο στον χάρτη που επισκέφθηκε πιο πριν και επιβεβαιώνει την ύπαρξή τους. Σε αρκετές περιπτώσεις δύναται να επαναλαμβάνεται ορισμένες φορές η πορεία για χαρτογράφηση εφόσον το ρομπότ επαληθεύει επανειλημμένα τα στοιχεία στον χώρο και τα παρουσιάζει στον χάρτη με περισσότερη ακρίβεια.

Πιο κάτω, στις Εικόνες 6-1 και 6-2 παρουσιάζονται δύο χάρτες της ίδιας περιοχής που επιβεβαιώνουν τα όσα ειπώθηκαν πιο πάνω. Αρχικά, παρατηρείται ο χάρτης που αποκτήθηκε με ένα μόνο πέρασμα. Παρόλο που είναι ξεκάθαρο το σκηνικό και μπορεί με ευκολία να διαπιστώσει κάποιος την τοποθεσία των διαφόρων αντικειμένων στον χώρο, με μία πιο προσεκτική ματιά διακρίνονται ασάφειες όσον αφορά το περίγραμμα των εμποδίων. Αυτό φαίνεται πιο ξεκάθαρα στα σημεία που είναι ξεθωριασμένα ή δύο γραμμές που αντικατοπτρίζουν ένα ενιαίο τοίχο, δεν συναντώνται μεταξύ τους. Δεν κατάφερε δηλαδή η συσκευή να προσδιορίσει με ακρίβεια το περίγραμμα της βάσης των αντικειμένων και να απομονώσει εξ ολοκλήρου το κάθε δωμάτιο. Η όλη συζήτηση αφορά μερικά εκατοστά, ωστόσο σε ένα ασφυκτικά

περιορισμένο περιβάλλον, όπως είναι για παράδειγμα μία αποθήκη ηλεκτρονικών συσκευών, είναι προτιμότερο να ανακτηθεί χάρτης όπως αυτόν που παρουσιάζεται στην Εικόνα 6-2.



Εικόνα 6-1: Χάρτης που ανακτήθηκε με ένα πέρασμα.



Εικόνα 6-2: Χάρτης που ανακτήθηκε με πολλαπλά συνεχόμενα περάσματα.

Στην συγκεκριμένη περίπτωση το ρομπότ ελίχθηκε στον χώρο για αρκετή ώρα εκτελώντας συνεχώς διαδρομές. Με αυτό τον τρόπο εξασφαλίστηκε ότι κάθε σημείο στον χώρο έχει σαρωθεί πάνω από μία φορά. Το αποτέλεσμα είναι εμφανές. Το περίγραμμα της βάσης των αντικειμένων και των διαχωριστικών τοίχων σκιαγραφήθηκε με χειρουργική ακρίβεια. Μετά από τέτοιο αποτέλεσμα είναι βέβαιο ότι δεν θα παρουσιαστούν οποιεσδήποτε επιπλοκές κατά τη φρούρηση του χώρου, τουλάχιστον όσο αφορά την αντιστοίχιση του χώρου με τον χάρτη.

Παρόλο που η ανακρίβεια στον χάρτη είναι λεπτομέρεια, και όπως έχει ειπωθεί πρόκειται για εκατοστά, ακόμα και χιλιοστά είναι παραπλανητικό και ύπουλο να κατοχυρωθεί ότι θεωρείται πάντα αμελητέα. Ακόμη και για μικρούς ανοικτούς χώρους με αρκετό διάστημα που επιτρέπουν μια πιο επιεική μετάβαση από ένα σημείο σε άλλο, η ολονύχτια πλοήγηση εξυπακούει και πολλές επαναλήψεις της ίδιας διαδρομής. Όσο μικρή κι αν είναι η απόκλιση, με τις απανωτές επαναλήψεις κάποια στιγμή θα γίνει αισθητή. Άρα, επιδιώκεται η πεπατημένη οδός με τη δημιουργία ξεκάθαρου χάρτη και την άρτια καθοδήγηση διαμέσου των διαδρόμων ώστε να αποφευχθούν προβλήματα κατά την πλοήγηση.

6.2.2Εντοπισμός (Localization)

Στο σημείο αυτό, έπειτα από την ολοκλήρωση του χάρτη, το ρομπότ χρειάζεται να ενημερωθεί σχετικά με την τρέχουσα του θέση σε σχέση με το περιβάλλον. Συγκεκριμένα, πρέπει να γνωρίζει την ακριβή διεύθυνση και τον προσανατολισμό του. Περιβάλλον πλέον για το ρομπότ αποτελεί ο χάρτης που δημιουργήθηκε πιο πριν. Η τρέχουσα κατεύθυνση του στον εκάστοτε χάρτη είναι γνωστή ως «θέση (pose)».

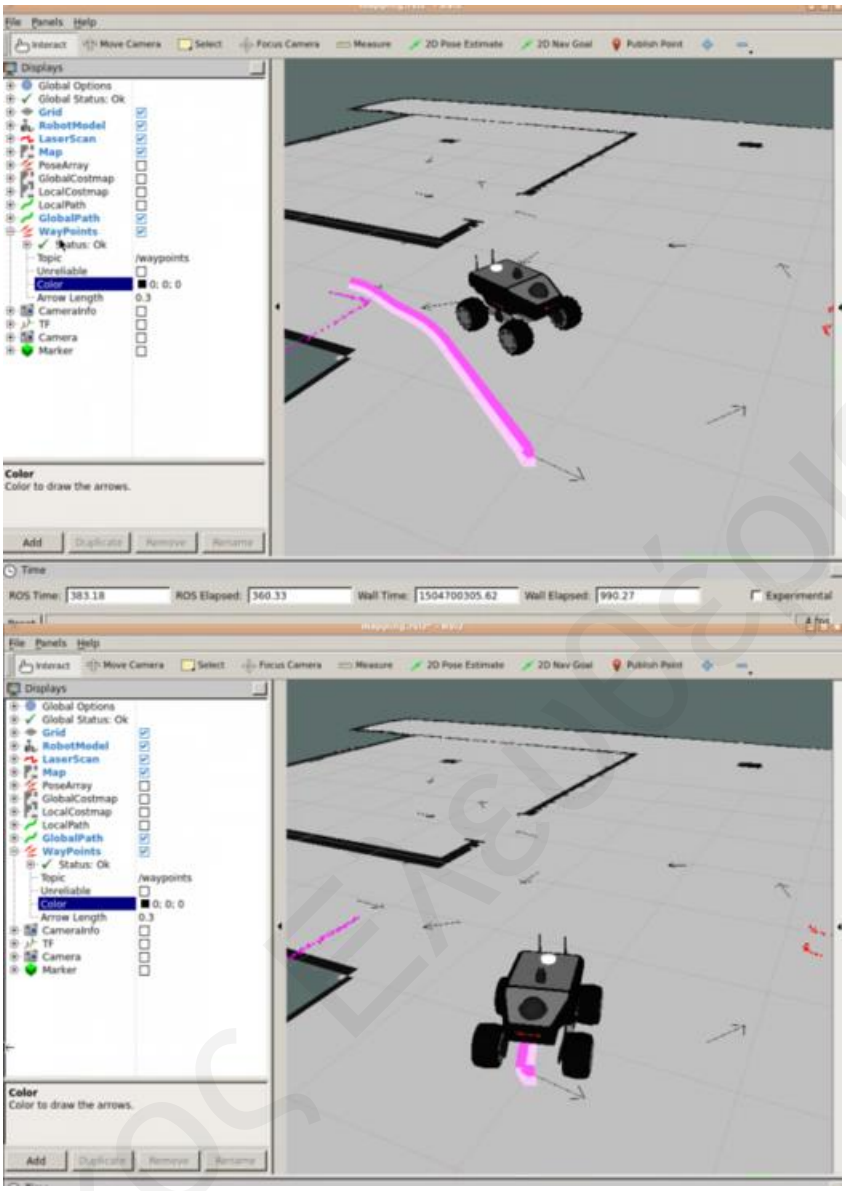
Ανάμεσα στα πολλά ρομπότ που κυκλοφορούν στην αγορά παρουσιάζεται μία πλούσια λίστα με εγκατεστημένους αλγορίθμους σκοπός των οποίων είναι το localization της συσκευής. Ένας αξιόπιστος αλγόριθμος που προσελκύει τις εντυπώσεις είναι ο AMCL (Adaptive Monte-Carlo Localization). Εγκατεστημένος σε συστήματα τελευταίας γενιάς, είναι από τους πιο διαδεδομένους με εξαιρετικά αποτελέσματα. Περιορίζεται προς το παρόν σε ρομπότ που κινούνται σε δύο διαστάσεις και πρόκειται για ένα πιθανολογικό σύστημα εντοπισμού. Με βάση την αρχή λειτουργίας του, υλοποιεί την προσαρμοστική (KLD-sampling) προσέγγιση εντοπισμού Monte Carlo (σύμφωνα με τον Dieter Fox) [38]. Η εν λόγω προσέγγιση κάνει χρήση ενός φίλτρου σωματιδίων για την παρακολούθηση της στάσης ενός ρομπότ έναντι μίας γνωστής χαρτογραφημένης περιοχής.

Η ποιότητα του συστήματος και κατ' επέκταση των μετρήσεων επηρεάζονται σε σημαντικό βαθμό από τα λογισμικά που έχει εγκατεστημένα. Όσο ικανό κι αν είναι το ρομπότ από μόνο του σαν συσκευή, όσο δυνατό υπολογιστή και να έχει εγκατεστημένο, πάντα θα υφίσταται η ανάγκη για ένα ικανό λογισμικό. Κάτι δηλαδή σαν τα ακριβά παπούτσια ποδοσφαίρου. Πάντα θα χρειάζονται ένα επιδέξιο ποδοσφαιριστή για να σκοράρουν.

6.2.3Σχεδιασμός Πορείας (Path Planning)

Έπειτα και από την αλληλεπίδραση ρομπότ – χάρτη, το επόμενο σημείο είναι η δημιουργία ενός μονοπατιού στο οποίο θα κινηθεί το ρομπότ. Το βήμα αυτό είναι υψίστης σημασίας εφόσον παραμονεύει, περισσότερο από κάθε άλλη φορά, ο κίνδυνος για να υποπέσει κανείς σε λάθος. Όπως και πριν έτσι και τώρα υπάρχει η δυνατότητα επαναπροσέγγισης της διαδικασίας από την αρχή, εντούτοις η συγκεκριμένη διαδικασία για να υλοποιηθεί απαιτεί χρονικό διάστημα το οποίο δεν μπορεί να θεωρηθεί αμελητέο.

Το μυστικό για την σκιαγράφηση πορείας που δεν θα μπερδέψει το ρομπότ είναι μερικά απλά βήματα τα οποία ο χειριστής πρέπει να ακολουθήσει πιστά. Πρώτα όμως να σημειωθεί ότι για κάθε σημείο που καλείται να προσεγγίσει το ρομπότ χρειάζεται η θέση (pose), όπως αυτή προσδιορίστηκε σε προηγούμενη ενότητα. Τα σημεία που θα αποτελέσουν μέρος της πορείας πρέπει να είναι κοντά το ένα στο άλλο. Επίσης, για σημεία μεταξύ των οποίων μεταβάλλεται ο προσανατολισμός περισσότερο από 90°, είναι προτιμότερο να εισαχθεί στο ενδιάμεσο ένα επιπλέον σημείο με τρόπο ώστε να αντιληφθεί το ρομπότ τη φορά με την οποία θα κινηθεί ώστε στο επόμενο σημείο να αποκτήσει την θέση που έχει οριστεί. Δηλαδή για μεταβολή προσανατολισμού 150° από ένα σημείο σε άλλο με απόσταση 1000mm μεταξύ τους, να μεσολαβεί στα 500mm σημείο που να υποδηλώνει αλλαγή προσανατολισμού κατά το ήμισυ, στις 75°. Τέλος, απαιτείται προσεκτική επιλογή σημείων ώστε να περνούν όσο γίνεται από το κέντρο των ελεύθερων διαδρόμων (βλ. Εικόνα 6-3). Αυτό θα βοηθήσει να βγουν στην επιφάνεια λάθη που έγιναν κατά την χαρτογράφηση, ή αν μη τι άλλο θα καθυστερήσει την εμφάνισή τους.



Εικόνα 6-3: Path Creation.

6.2.4Πλοήγηση (Navigation)

Όταν ολοκληρωθεί η πιο πάνω διαδικασία, ανάβει το πράσινο φως και το ρομπότ είναι πλέον σε θέση να κινηθεί αυτόνομα στον χώρο. Για συνεχή φρούρηση δύναται να γίνει επιλογή επαναλήψεων πλοήγησης στην ίδια πορεία, όπως επίσης και επ' άπειρον επαναλήψεις έως ότου τερματιστεί η λειτουργία από τον χειριστή.

Σε περίπτωση όπου εντοπίσει αντικείμενο που δεν προβλέπεται από τον χάρτη να υπάρχει, τότε λαμβάνονται εικόνες από τη συνεχή ροή ταινίας στην PTZ κάμερα. Οι εικόνες ελέγχονται μέσω του αλγόριθμου ανίχνευσης αντικειμένων YOLO. Σε ενδεχόμενο το ξένο σώμα να πρόκειται για άνθρωπο γίνεται ενημέρωση στο κέντρο ελέγχου αναφέροντας και το ποσοστό σιγουριάς, κατά πόσο ισχύει αυτό. Σε αντίθετη

περίπτωση, το ρομπότ επιχειρεί να αποφύγει το εμπόδιο με ένα μικρό ελιγμό και να επανέλθει στο αρχικό μονοπάτι. Όταν αυτό δεν καταστεί εφικτό, τότε ενημερώνει το κέντρο ελέγχου, ώστε ο υπεύθυνος να προβεί στις κατάλληλες ενέργειες για να ελευθερώσει το μονοπάτι.

6.3 Πορεία για εφαρμογή Διάσωσης

Η βασική διαφορά των επιχειρήσεων διάσωσης σε σύγκριση με τη φρούρηση είναι ο χρόνος. Στις περιπτώσεις όπου συγκροτείται ένα σύνολο προκειμένου να προβεί σε μία διάσωση, ο χρόνος που υπάρχει στη διάθεσή του είναι σημαντικά περιορισμένος. Πολλές φορές ακόμα και καθόλου διαθέσιμος χρόνος, εφόσον την ίδια στιγμή που γίνεται ενημέρωση υπάρχουν ήδη απώλειες. Άρα όσο γρηγορότερα λάβει χώρα η επιχείρηση, τόσο λιγότερη θα είναι η ζημιά. Ο τομέας της διάσωσης χωρίζεται σε δύο υποκατηγορίες. Ο λόγος για τις διασώσεις εσωτερικού χώρου και για τις διασώσεις εξωτερικού χώρου.

Όσον αφορά τις διασώσεις εσωτερικού χώρου, αυτές μπορεί να είναι πυρκαγιά σε ένα κτήριο, ή έλεγχος μετά από σεισμό και υπό τον φόβο μετασεισμού, ή ακόμη και έλεγχος κτηρίου για υποψία ύπαρξης εκρηκτικού μηχανισμού. Η πορεία είναι παρόμοια με τη φρούρηση εσωτερικού χώρου. Η διαφορά στην οποία αξίζει να γίνει αναφορά, είναι η διαδικασία χαρτογράφησης. Τα πλείστα κτήρια κατασκευάζονται βάσει σχεδίων και οι εργοληπτικές εταιρείες οφείλουν να υποβάλουν τα σχέδια προκειμένου ένα υποστατικό να εξασφαλίσει άδεια οικοδομής. Με το που σταλεί σήμα για περιστατικό σε κάποιο κτήριο, αμέσως ανακτώνται τα σχέδια και δημιουργείται ο χάρτης. Τότε η πλοήγηση γίνεται χειροκίνητα με βάση τον εν λόγω χάρτη και εικόνα εντός του κτηρίου προσδίδει η PTZ κάμερα. Σε περίπτωση που δεν υφίσταται χάρτης η πλοήγηση γίνεται στα τυφλά με όποια εικόνα προβάλλεται μέσω της κάμερας. Ο αλγόριθμος εντοπισμού ανθρώπου εξακολουθεί να βρίσκεται σε λειτουργία εφόσον μπορεί να εντοπίσει ανθρώπινη φιγούρα ακόμη και σε καθεστώτα ομίχλης, καπνού ή σκόνης με χαμηλότερο βέβαια ποσοστό αξιοπιστίας.

Αναφορικά με τις διασώσεις εξωτερικού χώρου, αυτές διαφέρουν πολύ περισσότερο σε σύγκριση με την πορεία φρούρησης.

6.3.1Χαρτογράφηση (Mapping)

Το γεγονός ότι το ρομπότ θα εργαστεί σε εξωτερικό χώρο, προσδίδει το πλεονέκτημα ότι υφίσταται κάλυψη από δορυφόρο. Σε αυτή την περίπτωση η δημιουργία χάρτη παραλείπεται και ο χάρτης λαμβάνεται έτοιμος από το δίκτυο. Απαιτείται βέβαια προσεκτική χρήση και ορθή αντιστοίχιση προκειμένου ο χάρτης να αποτελεί αντιπροσωπευτικό δείγμα της περιοχής ενδιαφέροντος.

Εάν όμως για οποιοδήποτε λόγο η λήψη του χάρτη δεν είναι εφικτή υπάρχουν εναλλακτικές λύσεις, η επιλογή των οποίων εναπόκειται στην κρίση του συντονιστή της επιχείρησης. Ορισμένες φορές μάλιστα αποφεύγεται η καθοδήγηση αποκλειστικά από τον έτοιμο χάρτη. Όπως παρουσιάζεται μέσω εφαρμογής στη συνέχεια της Διατριβής, η απευθείας λήψη του χάρτη εγκυμονεί σωρεία κινδύνων. Ως εκ τούτου, στις περιπτώσεις όπου εφαρμόζεται ακολουθείται μία διαδικασία από ελέγχους ώστε να πιστοποιηθεί το ποσοστό αξιοπιστίας.

Δεδομένου ότι θα προηγηθεί χειροκίνητη χαρτογράφηση, εκτιμάται αρχικά ο χώρος και αφότου πιστοποιηθούν και εξεταστούν όλοι οι παράγοντες οι οποίοι θα καθορίσουν την ποιότητα του χάρτη, τότε αρχίζει η χαρτογράφηση. Παράγοντες που πιθανόν να αποτελούν ορόσημο στην ποιότητα του χάρτη είναι η έκταση, η τραχύτητα και οι τρέχουσες καιρικές συνθήκες. Αρχής γενομένης με την έκταση του χωρίου, ο χειριστής θα αποφασίσει την ταχύτητα χαρτογράφησης, όπως επίσης και το σύνολο των επαναλήψεων. Κάτι που επηρεάζει ακόμη περισσότερο την απόφαση που θα κληθεί να λάβει ο συντονιστής σχετικά με την ταχύτητα είναι η ευελιξία εναλλαγής ταχύτητας. Η βασική διαφορά μεταξύ γρήγορης και αργής χαρτογράφησης είναι η εξής: σε μία γρήγορη χαρτογράφηση υφίσταται η δυνατότητα λειτουργίας και σε χαμηλότερες ταχύτητες. Αντίθετα, εάν ανακτηθεί σημαντικό μέρος του χάρτη με αργή ταχύτητα, τότε τυχόν επιτάχυνση κατά την χαρτογράφηση ενδέχεται να αφήσει ατέλειες στον χάρτη. Ιδανικά, χρειάζονται 4 – 6 επαναλήψεις προκειμένου να αποκτηθεί ένας καλός χάρτης στον οποίο μπορεί να βασιστεί κανείς για τη δημιουργία πορείας. Έτσι, για σχετικά ελεγχόμενες εκτάσεις, το ρομπότ εκτελεί όσες επαναλήψεις χρειαστούν με σχετικά γρήγορη ταχύτητα. Κάθε επιπλέον επανάληψη που προστίθεται στην πορεία χαρτογράφησης βελτιώνει σημαντικά τον χάρτη και αφαιρεί τυχόν ατέλειες που προκλήθηκαν σε προηγούμενα περάσματα. Από την άλλη όμως, όταν πρόκειται για ευρύτερες εκτάσεις σε ανοικτές περιοχές, συνηθίζεται το ρομπότ να κάνει ένα μόνο πέρασμα – ή το πολύ δύο – με πολύ χαμηλή ταχύτητα. Σε αντίστοιχες περιπτώσεις το περιεχόμενο του χάρτη διακρίνεται με άνεση παρόλο που ενδέχεται να μην έχει

διεξαχθεί καμία επανάληψη. Συν τοις άλλοις, σε μεγάλες εκτάσεις δεν δίνεται η ίδια σημασία στις λεπτομέρειες όσο δίνεται σε κοντινές κλειστές δομές.

Ο περιορισμός όμως στην ταχύτητα κατά την αργή χαρτογράφηση αυξημένων εκτάσεων δεν αφήνει περιθώρια λάθους. Με το που ανακτάται ο χάρτης, τυγχάνει επεξεργασίας και αξιολόγησης προκειμένου να προχωρήσει η επιχείρηση με τη δημιουργία της πορείας πλοήγησης. Το ενδεχόμενο ατελειών παραμένει στο προσκήνιο έως ότου η αρμόδια ομάδα κρίνει τον χάρτη κατάλληλο. Ακόμη και 8 ή 10 επαναλήψεις να έχουν διεξαχθεί, ο χάρτης που θα προωθηθεί για σκοπούς πλοήγησης ελέγχεται από φυσικό άτομο. Τότε και μόνο τότε μπορεί να συνεχίσει η διαδικασία με την εισαγωγή του χάρτη για την δημιουργία του μονοπατιού. Σε περίπτωση όμως που εντοπιστούν σημεία των οποίων η ακρίβεια υστερεί, τότε ο χάρτης δεν μπορεί να αποτελέσει αξιόπιστο στοιχείο προκειμένου να υλοποιηθεί η πορεία πλοήγησης. Άρα, η ανακατασκευή του χάρτη αποτελεί αδήριτη ανάγκη.

Το πόσο ανώδυνα μπορεί να υλοποιηθεί αυτό εξαρτάται αποκλειστικά από την παρούσα κατάσταση του τρέχοντος χάρτη, σε συνδυασμό με την ταχύτητα του ρομπότ στη χαρτογράφηση που έγινε. Όταν πρόκειται για χαρτογράφηση υπό γρήγορη ταχύτητα τότε τα πράγματα είναι απλά. Εντοπίζονται αρχικά τα σημεία του χάρτη που υστερούν σε ακρίβεια, ακολούθως καταγράφονται προκειμένου να διαφανεί εάν από τη δεύτερη φάση της χαρτογράφησης προέκυψαν καινούριες ατέλειες. Τότε το ρομποτικό σύστημα ελίσσεται στο χώρο με γρήγορες μεταβιβάσεις, προσεγγίζει τις εν λόγω περιοχές και καλύπτει τοπικά την περιοχή με απανωτά συνεχόμενα περάσματα σε αργή ή γρήγορη ταχύτητα.

Τίθεται όμως τεράστιο ζήτημα όταν πρόκειται για χαρτογράφηση που έγινε σε χαμηλή ταχύτητα. Το γεγονός ότι ο χάρτης έχει υλοποιηθεί με ένα ή το πολύ δύο μόνο περάσματα τον καθιστά «εύθραυστο» και τυχόν επανάληψη της χαρτογράφησης για ανανέωση του υφιστάμενου χάρτη ενδέχεται να διαιωνίσει το πρόβλημα, ή να δημιουργήσει άλλο αναπτύσσοντας ατέλειες σε διαφορετικά σημεία πέρα από εκείνα που εντοπίστηκαν αρχικά. Γι’ αυτό εξάλλου, όπως ειπώθηκε και πιο πάνω ο ιδανικός χάρτης θα προκύψει μετά από 4 – 6 περάσματα στην ίδια χαρτογράφηση. Νοουμένου ότι η έκταση δεν το επιτρέπει, ο χειριστής καλείται να βελτιώσει τα σημεία που δεν είναι ευδιάκριτα χωρίς να γίνει χαρτογράφηση από το μηδέν. Παρόλο που θα έλεγε κανείς ότι αυτό είναι προτιμότερο, ωστόσο ελλοχεύει ο κίνδυνος ο νέος χάρτης να παρουσιάζει επίσης ασάφειες. Γενικότερα, η δημιουργία ευδιάκριτου, στο σύνολο

χάρτη, με ένα ή δύο περάσματα είναι πρακτικά αδύνατο άσχετα με την ταχύτητα και το πόσο προσεκτικά γίνεται η διαδικασία.

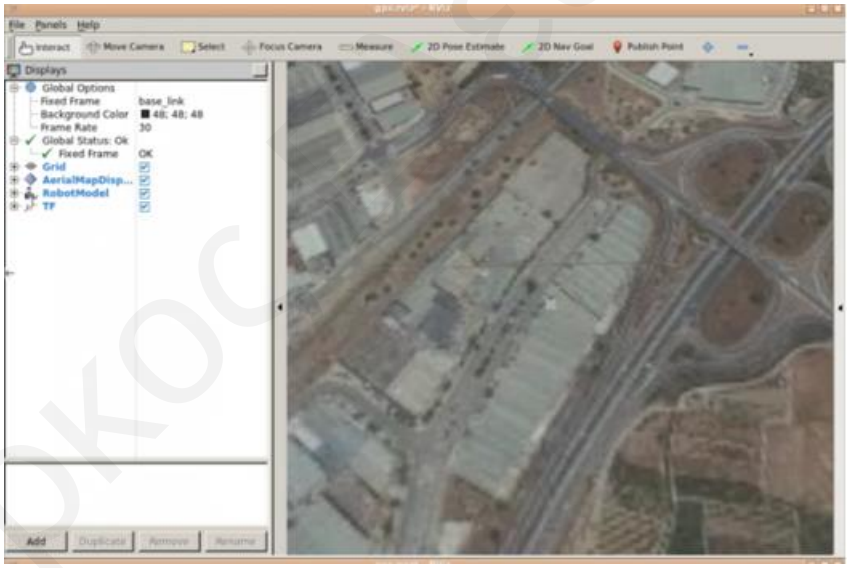
Παρόλο που υπάρχει ξεκάθαρη καθοδήγηση για το πως μπορεί να αντιμετωπιστεί το πιο πάνω πρόβλημα, κατά τη διάρκεια των δοκιμών εντοπίστηκε ένας ακόμη τρόπος ο οποίος δύναται να περιορίσει τον χρόνο ανακατασκευής του χάρτη σε μεγάλο βαθμό. Οι δύο τρόποι παρατίθενται στη συνέχεια, εντούτοις ο πιο αξιόπιστος είναι ο πρώτος τρόπος κάτι που συστήνεται ανεπιφύλακτα ειδικά για χειριστές που δεν έχουν εξοικειωθεί με το σύστημα και έχει ως εξής: το ρομπότ προσεγγίζει τις περιοχές ενδιαφέροντος του χάρτη με ταχύτητα όχι μεγαλύτερη από εκείνη με την οποία διεξήχθη η χαρτογράφηση. Στη συνέχεια εκτελεί σημαντικό αριθμό κινήσεων κατά μήκος και πλάτος της περιοχής με την ίδια ταχύτητα έως ότου η περιοχή στον χάρτη καταστεί ευδιάκριτη. Ο μεγαλύτερος περιορισμός στο σημείο αυτό είναι η ταχύτητα. Για παράδειγμα δύο μεμονωμένες περιοχές που χρειάζονται ανακατασκευή ενδέχεται να απέχουν μεταξύ τους πολλές δεκάδες μέτρα. Το ρομπότ επιβάλλεται, κατά τη μετακίνηση από την μία περιοχή στην άλλη, να κινείται με την ίδια χαμηλή ταχύτητα. Η διαδικασία αυτή αυξάνει το συνολικό κόστος, τις φθορές, καθώς επίσης και τον χρόνο διεξαγωγής της επιχείρησης.

Ο δεύτερος τρόπος είναι η επέμβαση στο .rviz file του χάρτη. Όπως φαίνεται στην Εικόνα 6-4 στην επιλογή File – Open υπάρχει η επιλογή ανάκλησης του χάρτη που δημιουργήθηκε και αποθηκεύτηκε προηγουμένως. Το αρχείο περιέχει σημαντικές πληροφορίες για την αντίληψη του χάρτη και θα κληθεί αργότερα να σχηματίσει την περιοχή στην οποία θα δημιουργηθεί η επιθυμητή πορεία πλοήγησης του ρομπότ. Αποτελείται κατά κύριο λόγο από γραμμές που αντικατοπτρίζουν τα εμπόδια που συνάντησε το ρομπότ κατά τη χαρτογράφηση. Οποιαδήποτε επέμβαση στο αρχείο αντιστοιχεί σε μετάλλαξη του χάρτη σύμφωνα πάντα με τα νέα δεδομένα. Σε αυτό το τελευταίο μπορεί να βασιστεί κανείς και να αναδιαμορφώσει τον χάρτη με απλή επεξεργασία του αρχείου. Η διαδικασία αυτή φαίνεται απλή, ωστόσο επιφυλάσσει αρκετούς κινδύνους. Αυτό που επιβάλλεται να ειπωθεί είναι ότι ο χάρτης εξυπηρετεί περισσότερο τον άνθρωπο στην αντίληψη του χώρου. Ακόμη δηλαδή και με μολύβι μπορεί να καταγραφεί ολόκληρη η περιοχή και να εισαχθεί στη θέση του χάρτη που δημιούργησε το ρομπότ. Επίσης, ακόμη και να βελτιωθεί ο χάρτης και οπτικά να μοιάζει με την περιοχή, οι διαστάσεις είναι πολύ δύσκολο να επιτευχθούν με ακρίβεια. Ακόμη και μία μικρή αλλαγή στην κλίμακα, στις περιοχές που ανακατασκευάστηκαν επεμβαίνοντας στο αρχείο μπορεί να επιφέρουν σοβαρές επιπλοκές κατά την πλοήγηση.

Η επέμβαση στον χάρτη γίνεται μέσω του προγράμματος GIMP 2.8 [61] στο οποίο θα ανακατασκευαστούν τα εμπόδια στην ήδη χαρτογραφημένη περιοχή, επιτρέποντας έτσι στον χρήστη να γνωρίζει κάθε λεπτομέρεια καθώς σχεδιάζει στον χάρτη την πορεία πλοήγησης.

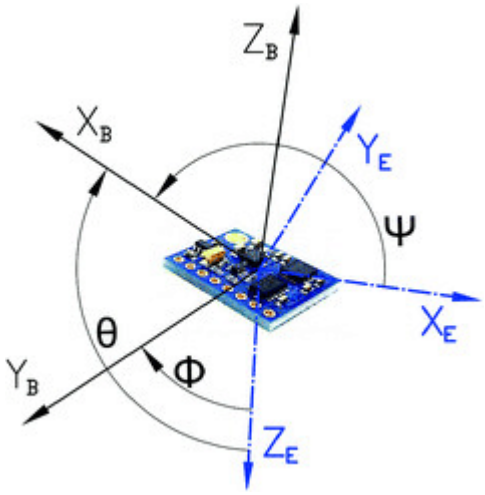
6.3.2Εντοπισμός (Localization)

Ο εντοπισμός του ρομπότ στον χάρτη είναι προτιμότερο να γίνει μέσω GPS. Το αποτέλεσμα θα είναι πανομοιότυπο με την Google Maps (βλ. Εικόνα 6-4). Τα δεδομένα GPS ωστόσο έχουν ορισμένους περιορισμούς. Κατ' αρχάς δεν είναι αρκετά ακριβή. Παρουσιάζουν δηλαδή θόρυβο, κάτι σαν ταλάντευση γύρω από την εκάστοτε τιμή και η θέση μετακινείται σημαντικά. Παρόλο που τα δεδομένα οδομετρίας έρχονται να μετριάσουν ορισμένα από τα σφάλματα, η ολίσθηση που προκύπτει από τους τροχούς (εάν είναι τροχοφόρο), επιφέρει επιπλέον σφάλματα στην οδομετρία.



Εικόνα 6-4: Προσαρμογή χάρτη από Google Maps.

Παράλληλα, το GPS δεν δίνει τον προσανατολισμό. Αρκείται μόνο σε ένα σημείο στον χάρτη, με αυτό να σημαίνει ότι με κάποιο άλλο τρόπο πρέπει να ανακτηθεί η συγκεκριμένη πληροφορία. Συνηθίζεται να γίνεται με κάποιο είδος πυξίδας, με τις περισσότερες εφαρμογές να κάνουν λόγο για αισθητήρα IMU (βλ. Εικόνα 6-5). Ανάμεσα στις ικανότητές του, το εν λόγω εξάρτημα δύναται να υπολογίζει τον προσανατολισμό της συσκευής στην οποία βρίσκεται εγκατεστημένο ανά πάσα στιγμή.



Εικόνα 6-5: IMU Sensor [36].

Το συμπέρασμα που προκύπτει είναι ότι το GPS αποτελεί ένα ακατέργαστο αλλά πολύ χρήσιμο σύστημα προκειμένου μια συσκευή να τοποθετηθεί και να καθοδηγήσει το ρομπότ σε εξωτερικό χώρο στον οποίο δεν προηγήθηκε χαρτογράφηση. Κατά την περιήγηση υπάρχει η δυνατότητα να δημιουργηθεί χάρτης και σε περίπτωση αποτυχίας της επιχείρησης να επαναληφθεί η διαδικασία εντοπισμού με τον χάρτη που προέκυψε μέσω της χαρτογράφησης. Το μόνο σίγουρο είναι ότι ο δεύτερος είναι πιο ακριβής, η δημιουργία του όμως εναπόκειται στην κρισιμότητα της κλήσης εντοπισμού.

6.3.3Πλοήγηση (Navigation)

Με την αντιμετώπιση των πιο πάνω ζητημάτων, απομένει η περιήγηση στον χώρο ενδιαφέροντος. Η πιθανότητα ύπαρξης συνωστισμού στο κέντρο ελέγχου λόγω της κρισιμότητας του περιστατικού, επιτρέπει την παράλειψη δημιουργίας μονοπατιού στο οποίο θα κινηθεί το ρομπότ. Είναι πιο πρακτικό η συσκευή να μετακινείται χειροκίνητα ώστε να επικεντρώνεται ανά πάσα στιγμή σε ό,τι κεντρίσει το ενδιαφέρον χωρίς να διακόπτεται η προκαθορισμένη πορεία. Επίσης, σημειώνεται ότι σε περίπτωση που διακοπεί η πορεία, είτε θα πρέπει να επιστρέψει το ρομπότ στην αρχική θέση, είτε θα πρέπει να ανανεωθεί το μονοπάτι ξεκινώντας από το σημείο στο οποίο βρίσκεται, κάτι που επιβραδύνει τη διαδικασία εντοπισμού.

Πέραν των περιστατικών διάσωσης όμως, ο σχεδιασμός πορείας πλοήγησης ενίοτε προσέλκυε περισσότερα βλέμματα σε μία εφαρμογή εξωτερικού χώρου παρά σε εσωτερικού. Αυτό συμβαίνει λόγω της διαφοράς του εδάφους στις δύο περιπτώσεις. Ένας εσωτερικός χώρος, ακόμη και να μην έχει εξερευνηθεί προηγουμένως από τον χειριστή δεν μπορεί να είναι οτιδήποτε άλλο εχτός από επίπεδος. Ακόμη και

μεμονωμένα σκαλοπάτια να υπάρχουν, αυτά διακρίνονται μέσω των σχεδίων του κτηρίου και αντιμετωπίζονται κατάλληλα.

Σε εξωτερικό χώρο όμως και δη σε ανώμαλο έδαφος η διαμόρφωση της πορείας δεν μπορεί να γίνει με την ίδια άνεση. Ο χειριστής εφόσον γνωρίζει καλύτερα από τον καθένα το ρομποτικό σύστημα, πρέπει να είναι σε θέση να προβλέψει και να αποφύγει τυχόν παγίδες στο έδαφος που ενδέχεται να πλήξουν το ρομπότ. Η μεγαλύτερη πλάνη στην οποία μπορεί να υποπέσει κάποιος είναι να βασιστεί εξ ολοκλήρου σε έτοιμο χάρτη (π.χ., από Google Maps), χωρίς να επιβλέψει ο ίδιος την περιοχή. Ο λόγος διότι ο χάρτης, είτε αυτός που κατασκευάζεται είτε αυτός που λαμβάνεται έτοιμος είναι δισδιάστατος. Πιο συγκεκριμένα σε ένα δισδιάστατο χάρτη δεν μπορούν να διακριθούν λακκούβες και σημεία στο έδαφος με κλίση. Η πτώση του ρομπότ σε λακκούβα μπορεί να προκαλέσει ζημιά στο ρομποτικό σύστημα. Από την άλλη η κλίση στο έδαφος περιορίζει την ταχύτητα περιήγησης. Τυχόν αύξηση πέρα του ορίου ενδέχεται να επιφέρει ανατροπή του ρομπότ, κάτι που για τα περισσότερα ρομπότ είναι μη αναστρέψιμο χωρίς ανθρώπινη παρέμβαση. Σε αφιλόξενο για τον άνθρωπο περιβάλλον γίνεται επίταξη δεύτερου ρομπότ ώστε να βοηθήσει στην ανάκτηση του πρώτου.

Όλα τα πιο πάνω μπορούν να αποφευχθούν με εξερεύνηση της περιοχής και αναλυτική καταγραφή των ιδιαιτεροτήτων της. Όσο κρίσιμο και αν είναι το περιστατικό που λαμβάνει χώρα, η αποτύπωση του μοντέλου της περιοχής είναι απαραίτητη. Αυτό εξασφαλίζει την ακεραιότητα του εξοπλισμού, ενώ παράλληλα επιτρέπει στην ομάδα συντονισμού της επιχείρησης να εκτελεί τις όποιες ενέργειες βάσει πραγματικών δεδομένων.

6.4 Προσομοιωτής Gazebo

Το Gazebo είναι ένα open-source λογισμικό προσομοιωτή που χρησιμοποιείται κυρίως στο χώρο της ρομποτικής για την ανάπτυξη και τον έλεγχο ρομποτικών συστημάτων. Προσομοιώνει τη φυσική συμπεριφορά των ρομποτικών συστημάτων σε διαφορετικά περιβάλλοντα και συνθήκες, όπως επίσης και την λειτουργία αισθητήρων και άλλων στοιχείων ελέγχου. Χρησιμοποιείται συνήθως για τη δοκιμή και τη βελτιστοποίηση των ρομποτικών αλγορίθμων και την ανάπτυξη νέων συστημάτων, χωρίς να απαιτείται πρόσβαση σε πραγματικά ρομπότ.

Κεφάλαιο 7

7 ΠΕΔΙΟ ΕΦΑΡΜΟΓΗΣ – ΑΝΑΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ

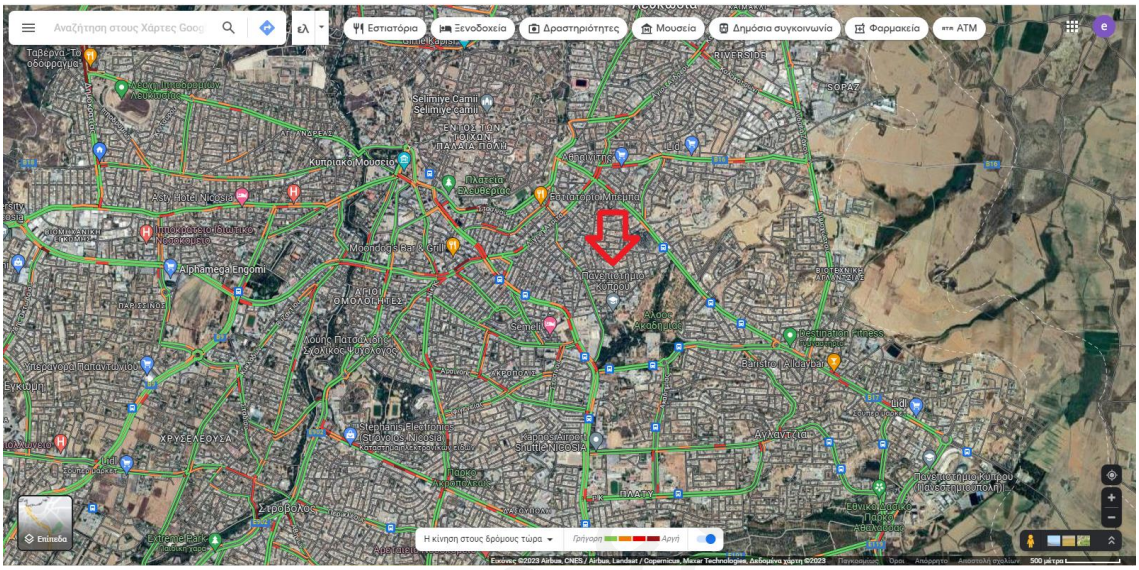
7.1 Το πρόβλημα

Για την επιλογή παρουσίασης του συνολικού όγκου εργασίας, έγινε προσπάθεια εύρεσης μίας ολοκληρωμένης εφαρμογής η οποία θα επέτρεπε την επίδειξη όσο το δυνατό περισσότερων πτυχών από τις ικανότητες των μοντέλων που ολοκληρώθηκαν. Η επιλογή στην πορεία που θα ακολουθούσε το ρομπότ ήταν κατά κάποιο τρόπο προδιαγεγραμμένη, ωστόσο το σύστημα με τον τρόπο που στήθηκε, επιτρέπει την λειτουργία τόσο σε συνθήκες φρούρησης, όσο και σε επιχειρήσεις διάσωσης. Για σκοπούς διατριβής έγινε επιλογή πορείας φρούρησης εξασφαλίζοντας με αυτό τον τρόπο την ελάχιστη δυνατή καταπόνηση του εξοπλισμού.

Το αρχικό σενάριο επικεντρώνεται στην φρούρηση και φύλαξη των υποδομών μίας κοινότητας κατά τις ώρες που δεν βρίσκεται σε λειτουργία. Το χρονικό αυτό διάστημα ενδέχεται να περιλαμβάνει βραδινές, απογευματινές αλλά και πρωινές ώρες ανάλογα με την ημέρα. Στόχος της συγκεκριμένης εφαρμογής είναι η άμεση ενημέρωση του εκάστοτε αρμόδιου λειτουργού για την ύπαρξη ανθρώπου στον χώρο. Ακολούθως ο λειτουργός χρειάζεται να αποκτήσει πρόσβαση στο οπτικό υλικό στο οποίο και ανιχνεύθηκε ο άνθρωπος ούτως ώστε να παρατηρήσει και να κρίνει την σοβαρότητα του περιστατικού.

Στις αρχικές επιλογές υπήρξαν διάφορες υποδομές όπως νοσοκομεία, σχολεία, πάρκα, στρατόπεδα καθώς επίσης και αποθήκες φύλαξης εξοπλισμού. Στην φάση αυτή τέθηκαν ορισμένα κριτήρια ώστε ο χώρος που θα επιλεχθεί να συνδυάζει όσο το δυνατό περισσότερες πτυχές των δεξιοτήτων που αναπτύχθηκαν. Χρειαζόταν λοιπόν ένας χώρος που να μπορούσε να συνδυάσει υπαίθρια αυλή στεγασμένη και μη, εγκαταστάσεις με τουλάχιστον ένα όροφο, όπως επίσης και εσωτερικοί χώροι. Παράλληλα χρειαζόταν να υπάρχει ένας σταθμός στον οποίο θα εγκαθίστανται όλος ο εξοπλισμός και θα αντιπροσώπευε τον χώρο εργασίας του αρμόδιου λειτουργού. Με γνώμονα πάντα τα όσα επισημάνθηκαν μόλις, αλλά και τη διαφύλαξη του εξοπλισμού σε συνδυασμό με την εξοικονόμηση πόρων στον μέγιστο βαθμό, επιλέχθηκαν οι

κτηριακές εγκαταστάσεις των Κεντρικών κτηρίων του Πανεπιστημίου Κύπρου. Η εν λόγω περιοχή βρίσκεται στο κέντρο της Λευκωσίας και είναι παρά το εργαστήριο ρομποτικής του τμήματος Μηχανικών Μηχανολογίας και Κατασκευαστικής του Πανεπιστημίου Κύπρου (UCY Robotics Lab).



Εικόνα 7-1: Χάρτης Περιοχής Λευκωσίας [28].

Η συγκεκριμένη κοινότητα παρουσιάζει μία μεγάλη ανοικτή αυλή, η οποία περιβάλλεται από εγκαταστάσεις σε ισόγειο και πρώτο όροφο. Σε κοντινή απόσταση βρίσκεται μία αυλή η οποία χρησιμοποιείται σαν υπαίθριος χώρος του εστιατορίου κατά τους καλοκαιρινούς μήνες και επικαλύπτεται από δέντρα. Εντός της ίδιας ακτίνας βρίσκεται το γραφείο του φύλακα όπως επίσης και το εργαστήριο ρομποτικής. Τα σενάρια φρούρησης της συγκεκριμένης κοινότητας δύναται να τροποποιούνται ανάλογα με την μεταβολή σημαντικών παραγόντων. Ορισμένοι από αυτούς είναι οι καιρικές συνθήκες, η ώρα της ημέρας που η φρούρηση κρίνεται αναγκαία, καθώς επίσης και η εκάστοτε ορατότητα.

Για την εκπλήρωση των απαιτήσεων της εφαρμογής τα ρομπότ, SUMMIT XL και MAVIC 2, φρουρούν τις δύο περιοχές. Την ανοικτή αυλή μεγάλης έκτασης, όπως επίσης και τον υπαίθριο χώρο της καφετέριας. Με την αναγνώριση ανθρώπου, το σύστημα ενημερώνει για την ύπαρξη μέσω ειδοποίησης σε τοπικό δίκτυο. Παράλληλα, αναφέρει το ποσοστό αξιοπιστίας, το πόσο σίγουρο δηλαδή είναι το μοντέλο για αυτό που ανιχνεύει. Υπάρχουν περιπτώσεις που αυτό μπορεί να φανεί εξαιρετικά χρήσιμο για το άτομο που διαχειρίζεται το περιστατικό.

Θέμα μείζονος σημασίας αποτελεί το γεγονός διαχείρισης των πληροφοριών αφού το υλικό που ανακτάται από το ρομπότ θα πρέπει να τυγχάνει προσεκτικής αντιμετώπισης. Πέραν των προσωπικών δεδομένων και του περί διαφύλαξης νόμου, στο τραπέζι τέθηκε και το ζήτημα μεταφοράς δεδομένων μέσω διαδικτύου. Το γεγονός να αποκτήσει κάποιος πρόσβαση σε αυτά τα δεδομένα εγκυμονεί δύο μεγάλους κινδύνους.

Ο πρώτος μεγάλος κίνδυνος διαρροής πληροφοριών αφορά την αξιοπιστία και αποτελεσματικότητα της εφαρμογής. Σε ενδεχόμενη απόπειρα προσέγγισης των εγκαταστάσεων από μη εξουσιοδοτημένα άτομα είναι σημαντικό ο εισβολέας να μην γνωρίζει ότι οι αρμόδιες αρχές έχουν ενημερωθεί για την παρουσία του. Εάν όμως ο δράστης αποκτήσει λαθραία πρόσβαση στο σύστημα, τότε θα μπορεί και ο ίδιος να ενημερωθεί μόλις εντοπιστεί από το ρομπότ, οδηγώντας τον έτσι να τραπεί σε φυγή. Εν πάση περιπτώσει προκειμένου το μοντέλο να πληροί τα κριτήρια και να αντεπεξέρχεται σε όσο το δυνατό περισσότερες εφαρμογές η μετάδοση των πληροφοριών θα γίνεται μέσω τοπικού δικτύου. Παρόλα αυτά όμως υπάρχει η δυνατότητα εγκατάστασης φωτεινών ή ηχητικών δεικτών σε περιπτώσεις που κριθεί απαραίτητο και εφαρμόζεται συνήθως για εκφοβισμό του εισβολέα.

Το δεύτερο μεγάλο ζήτημα στο θέμα διαρροής πληροφοριών επικεντρώνεται στον περί προστασίας προσωπικών δεδομένων νόμο. Με τον τρόπο που στήθηκε το μοντέλο, ο υπολογιστής αποθηκεύει φωτογραφίες και βίντεο για λόγους ασφαλείας. Αυτό εξυπηρετεί υπηρεσίες στις οποίες κατά τις ώρες μη λειτουργίας υπάρχει μόνο ένα άτομο για φρούρηση. Το γεγονός ότι αποθηκεύονται εικόνες και ταινίες μικρού μήκους, χρίζει προσεκτικής αντιμετώπισης.

Λαμβάνοντας σοβαρά υπόψη τους παράγοντες ασφαλείας αποφασίστηκε πως η μεταφορά πληροφοριών θα γίνει μέσω τοπικού δικτύου. Παράλληλα, χρειάστηκε να συμβαδίσουν και να επικοινωνούν η γλώσσα προγραμματισμού του ρομπότ μαζί με τον αλγόριθμο για την ανίχνευση αντικειμένων. Η σύνδεση αυτή είναι αδήριτη ανάγκη αφού σε αντίθετη περίπτωση δεν θα μπορούσε να επιτευχθεί η πλοήγηση του ρομπότ ταυτόχρονα με την ανίχνευση ανθρώπων. Ως εκ τούτου, το σύστημα παρουσιάζεται φτωχό και αδυνατεί να ικανοποιήσει τις απαιτήσεις ακόμη και των πιο απλών εφαρμογών.

Στο παρόν στάδιο τίθεται ως στόχος αφενός η παράλληλη λειτουργία των δύο λογισμικών από τον ίδιο υπολογιστή και αφετέρου η μεταξύ τους επικοινωνία. Παρόλο

που το όλο έργο μπορεί να υλοποιηθεί με δύο ξεχωριστούς υπολογιστές και να λειτουργεί χωρίς πρόβλημα, εντούτοις η λύση αυτή δεν είναι πρακτική.

Σε αυτό το σημείο πρέπει να σημειωθεί ότι τόσο ο υπολογιστής, όσο και το ROS με το YOLO μπορούν να συνυπάρξουν και να λειτουργούν στο 100% χωρίς ενδοιασμούς μόνο σε ορισμένες εκδόσεις. Λόγου χάρη ο υπολογιστής που χρησιμοποιήθηκε λειτουργεί με Linux Ubuntu 20.04. Οι συμβατές εκδόσεις του ROS που συμβαδίζουν είναι το Noetic και το Foxy. Από την άλλη το YOLOv7 μπορεί να συμβαδίσει με ROS Humble και Noetic. Από την στιγμή που το ROS Humble συμβαδίζει με επιτυχία μόνο σε Ubuntu 22.04, απομένει μόνο μία γραμμή που να επιτρέπει και στα τρία λογισμικά να συνυπάρχουν και να λειτουργούν ταυτόχρονα στην ίδια συσκευή. Οι εκδόσεις που χρησιμοποιήθηκαν ώστε να λειτουργούν χωρίς ζήτημα είναι οι εξής:

- Ubuntu 20.04
- ROS Noetic
- YOLOv7

Πέραν των τριών όμως υπάρχουν και άλλες βιβλιοθήκες που πρέπει να συμφωνούν με τα πιο πάνω όπως είναι το TensorFlow, tensorboard και NumPy. Όσον αφορά την κάθε έκδοση προκειμένου να συμβαδίζει με το YOLOv7, κατά τη διάρκεια της εγκατάστασης του αλγορίθμου υπάρχει ένα επιπρόσθετο βήμα στο οποίο εγκαθίσταται αρχείο με όνομα requirements.txt όπου υπάρχουν οι εκδόσεις που απαιτείται να υφίσταται η κάθε βιβλιοθήκη προκειμένου να συμβαδίζει με τον αλγόριθμο (βλ. Εικόνα 7-2). Στις πλείστες μάλιστα δίνεται και εύρος εκδόσεων ώστε ο χειριστής να επιλέξει.

7.2 Τρόπος Επίλυσης

Έχοντας λοιπόν τις δύο περιοχές που φαίνονται στην Εικόνα 7-3, το ρομποτικό σύστημα χρειάζεται να παρέχει εξ ολοκλήρου κάλυψη στην ανοικτή περιοχή που είναι σχετικά μεγάλης έκτασης, στους διαδρόμους παρά την ανοικτή αυλή, όπως επίσης και στους διαδρόμους του πρώτου ορόφου και τέλος στην περιοχή που καλύπτεται με τα φυλλώματα των δέντρων. Παράλληλα επιτεύχθηκε σύνδεση του SUMMIT XL με φορητό υπολογιστή, κάτι που προσδίδει ευελιξία στην τοποθέτηση του σταθμού ελέγχου. Στην Εικόνα 7-3 παρουσιάζεται η συνολική περιοχή για την οποία χρειάστηκε να υπάρχει κάλυψη.

Open requirements.txt ~/yolov7 Save

```
# Usage: pip install -r requirements.txt

# Base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.1.1
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
tqdm>=4.41.0
protobuf<4.21.3

# Logging -------------------------------------
tensorboard>=2.4.1
# wandb

# Plotting ------------------------------------
pandas>=1.1.4
seaborn>=0.11.0

# Export --------------------------------------
# coremltools>=4.1  # CoreML export
# onnx>=1.9.0  # ONNX export
# onnx-simplifier>=0.3.6  # ONNX simplifier
# scikit-learn==0.19.2  # CoreML quantization
# tensorflow>=2.4.1  # TFLite export
# tensorflowjs>=3.9.0  # TF.js export
# openvino-dev  # OpenVINO export

# Extras --------------------------------------
ipython  # interactive notebook
psutil  # system utilization
thop  # FLOPs computation
# albumentations>=1.0.3
# pycocotools>=2.0  # COCO mAP
# roboflow
```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

Εικόνα 7-2: Αρχείο requirements.txt.

Εικόνα 7-3: Περιοχή φρούρησης για τη δοκιμασία.

Η κάθε περιοχή παρουσιάζει ιδιαιτερότητες και εξετάζονται τρεις διαφορετικές επιλογές για την παρακολούθησή τους που παρουσιάζονται και εφαρμόζονται παρακάτω.

7.2.1Επιλογή 1: Χρήση drone με κάμερα

Η αυλή που περικλείεται από το κτήριο αντιστοιχεί στην ανοικτή ευρεία περιοχή χωρίς πολλά εμπόδια. Ένα πιο κοντινό πλάνο φαίνεται στην Εικόνα 7-4, όπου και διακρίνονται σημαντικές λεπτομέρειες της περιοχής. Σε πρώτη φάση διακρίνονται δύο μεγάλες εκτάσεις γρασιδιού, ένα μικρό σιντριβάνι, παγκάκια, μερικά δέντρα και λάμπες που στηρίζονται σε πασσάλους. Με μία πιο προσεκτική ματιά όμως διακρίνονται αρκετές σχάρες και αριθμός μεμονωμένων σκαλοπατιών ειδικά κοντά στο σιντριβάνι. Πέρα όμως από αυτά άλλη ιδιαιτερότητα είναι η ύπαρξη γάτων στον χώρο.

Εικόνα 7-4: Ανοικτός εξωτερικός χώρος μεγάλης έκτασης.

Λαμβάνοντας υπόψη όλα τα πιο πάνω, σε συνδυασμό με την έκταση που καταλαμβάνει η περιοχή, αποφασίστηκε πως για κάλυψη της θα χρησιμοποιηθούν εναέρια ρομπότ. Με τον τρόπο αυτό εξασφαλίζεται η κάλυψη και στους διαδρόμους σε

ισόγειο και πρώτο όροφο που παρουσιάζονται στην Εικόνα 7-5. Για την ανίχνευση όμως ανθρώπων σε φωτογραφίες και βίντεο από drone χρειάστηκε να γίνει εκπαίδευση του μοντέλου με αρχικό σύνολο δεδομένων που περιείχε εικόνες από εναέρια μέσα. Παρόλο που οι αρχικές δοκιμές έγιναν με weights από φωτογραφίες εδάφους, εντούτοις το ποσοστό αξιοπιστίας ήταν αρκετά ικανοποιητικό. Σαφώς και με το νέο σύνολο δεδομένων η αξιοπιστία αυξήθηκε σε απόλυτα ικανοποιητικό βαθμό, ωστόσο εν τέλει έγινε συνένωση των δύο συνόλων προκειμένου να υλοποιηθεί ένα μοναδικό αρχείο με weights που να ικανοποιεί κάθε περίπτωση.

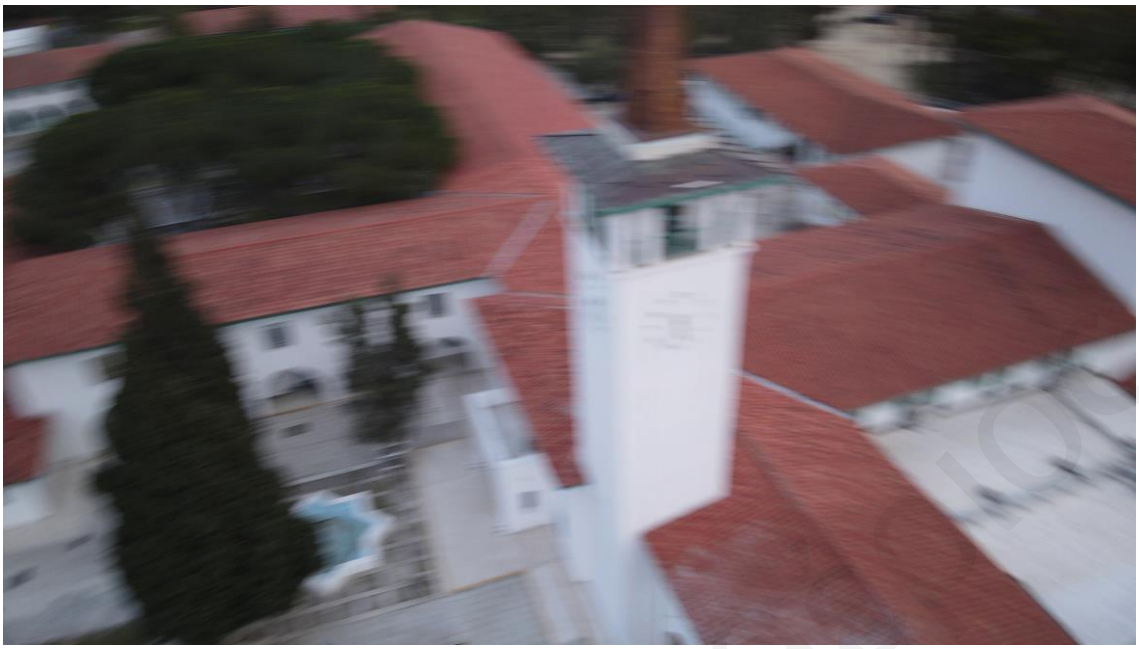
Εικόνα 7-5: Διάδρομοι σε ισόγειο και πρώτο όροφο πέριξ της αυλής.

Κατά την διάρκεια των δοκιμών εντοπίστηκαν ορισμένα «κενά» σημεία τα οποία χρειάστηκε να ξεπεραστούν ώστε το σύστημα να αποτελεί σημαντικό εμπόδιο για κάποιον που θα επιχειρήσει να εισέλθει στους χώρους του Πανεπιστημίου Κύπρου με κακόβουλες προθέσεις. Ένα από αυτά τα σημεία είναι το προστατευτικό τοίχωμα κατά μήκος ολόκληρου του διαδρόμου στον πρώτο όροφο της υποδομής. Το συγκεκριμένο ζήτημα αντιμετωπίστηκε με κατάλληλη λήψη του διαδρόμου όπως φαίνεται στο πλάνο που απεικονίζει η Εικόνα 7-6. Μπορεί ο τρόπος προσέγγισης του διαδρόμου να αφήνει ένα μικρό τυφλό διάστημα κατά μήκος της γραμμής ωστόσο εάν αναλογιστεί κανείς ότι το drone θα καλύπτει την περιοχή σε μερικά δευτερόλεπτα και θα επιστρέφει στη θέση που φαίνεται στην Εικόνα 7-6, ακόμη και να επιχειρήσει κάποιος να διασχίσει τον διάδρομο είναι δύσκολο να μην γίνει αντιληπτός ούτε μία στιγμή.

Εικόνα 7-6: Κάλυψη διαδρόμου πρώτου ορόφου.

Απ’ εκεί και έπειτα έπρεπε να αποφασιστεί το ύψος και η ταχύτητα κίνησης του ρομπότ. Αναφορικά με το ύψος έγιναν αρκετές δοκιμές σε διαφορετικά επίπεδα, ενώ υπήρξαν περιπτώσεις στις οποίες το ρομπότ μετέβαινε σε διαφορετικά ύψη κατά την διάρκεια της ίδιας πτήσης. Λεπτομερής αναφορά για την ρύθμιση των παραμέτρων του drone παρατίθεται στο Κεφάλαιο 8.3.2. Από την άλλη η ταχύτητα που αναπτύσσεται πρέπει να επιτρέπει στον αλγόριθμο να εκτελεί αναγνώριση. Όπως έχει σημειωθεί το μοντέλο αναλύει ένα βίντεο σε πολλά frames και στη συνέχεια εκτελεί αναγνώριση στο καθένα από αυτά. Εάν για παράδειγμα το ρομπότ καλύπτει απόσταση 100 μέτρων σε μερικά δευτερόλεπτα τότε ελλοχεύει ο κίνδυνος τα frames να είναι χαμηλής ακρίβειας. Σε καμία περίπτωση εικόνες όπως αυτή της Εικόνας 7-7 δεν είναι επιθυμητές.

Η συγκεκριμένη εφαρμογή γίνεται αποδεκτή σε περιπτώσεις όπου η έκταση που χρειάζεται να καλυφθεί είναι αρκετά μεγάλη. Παράλληλα εξυπηρετεί χώρους στους οποίους απαιτείται ταυτόχρονη κάλυψη πέραν του ενός ορόφου κάτι το οποίο αδυνατεί να πράξει ένα ρομπότ εδάφους. Για την ιδιαιτερότητα της εφαρμογής που προκύπτει έπειτα και από την ύπαρξη γάτων στον χώρο, αφαιρέθηκε τόσο η εν λόγω κατηγορία από την λίστα κλάσεων, όσο και αρκετές άλλες κατηγορίες. Αυτό εξυπηρετεί κατά κύριο λόγο τον λειτουργό ασφαλείας, διότι αποφεύγεται με αυτό τον τρόπο η εμφάνιση περιττών ειδοποιήσεων στο παράθυρο του υπολογιστή. Έτσι, κατά την διάρκεια της ανίχνευσης στο παράθυρο με τα αντικείμενα που εντοπίστηκαν θα εμφανίζονται μόνο όσα χρειάζεται να απασχολήσουν την ασφάλεια του κτηρίου.



Εικόνα 7-7: Κακή ποιότητα εικόνας μετά από ανάλυση βίντεο.

Εν τέλει, μπορεί να χαρακτηριστεί στο σύνολο εξαιρετικά χρήσιμη εφαρμογή και ειδικότερα σε ανοικτές δομές οι οποίες χωρίζονται από φράκτες. Αυτό επιβεβαιώθηκε στην περίπτωση όπου ο αλγόριθμος εντόπισε ανθρώπινη φιγούρα σε γειτονική αυλή. Μπορεί ο συγκεκριμένος χώρος να μην εμπίπτει στους οριοθετημένους χώρους προς παρακολούθηση, εντούτοις το μόνο που είναι σίγουρο είναι ότι αυτός που θα επιχειρήσει να προσεγγίσει την κεντρική αυλή θα το πράξει μεταβαίνοντας μέσω ενός χώρου που γειτνιάζει με αυτή. Θα ήταν λοιπόν βοηθητικό εάν υπήρξε ενημέρωση για την ύπαρξη ανθρώπου σε κοντινή απόσταση στην οποία επίσης δεν συντρέχει λόγος για να βρίσκεται κάποιος εκεί τις ώρες που το πανεπιστήμιο δεν λειτουργεί.

Στην Εικόνα 7-8 που ακολουθεί εντοπίστηκε από τον αλγόριθμο άνθρωπος με ποσοστό αξιοπιστίας 0.84, κάτι το οποίο είναι πραγματικά εντυπωσιακό εάν αναλογιστεί κανείς την απόσταση που πάρθηκε η εικόνα και τη θέση του ανθρώπου στον χώρο. Την ίδια στιγμή στο βάθος καλύπτεται ο χώρος στάθμευσης κάτι που επίσης μπορεί να ενδιαφέρει την ασφάλεια. Παρά το ότι η βλάστηση περιμετρικά του χώρου στάθμευσης επικαλύπτει σημαντικό μέρος του χώρου, ωστόσο ο αλγόριθμος έχει εντοπίσει τα οχήματα κάτι που επιβεβαιώνει την ακρίβεια του μοντέλου. Ο εντοπισμός οχημάτων στον χώρο στάθμευσης κατά τις ώρες που το πανεπιστήμιο δεν λειτουργεί μπορεί επίσης να βοηθήσει στο έργο του λειτουργού ασφαλείας.

Εικόνα 7-8: Χώρος που γειτνιάζει με την κεντρική αυλή.

7.2.2Επιλογή 2: Χρήση κινητού ρομπότ με κάμερα

Ο εξωτερικός χώρος του εστιατορίου αποτελεί επίσης περιοχή ενδιαφέροντος για την οποία πρέπει να υπάρξει κάλυψη. Η διάταξη του χώρου σε συνδυασμό με την βλάστηση και τις πυκνές φυλλωσιές των δέντρων δεν επιτρέπει την προσέγγιση της περιοχής από το drone σε κοντινή απόσταση. Στην Εικόνα 7-9 παρουσιάζεται φωτογραφία της περιοχής που λήφθηκε από drone. Η ανίχνευση ανθρώπων στον χώρο από παρόμοιο υλικό δεν είναι δυνατό εφόσον η βλάστηση καλύπτει εξ ολοκλήρου τον χώρο. Λόγω των πολλαπλών εμποδίων κρίθηκε προτιμότερο να δημιουργηθεί ένας χάρτης βάσει του οποίου θα κινηθεί ένα ρομπότ εδάφους, πιο συγκεκριμένα το SUMMIT XL.

Παρά τη μικρή έκταση προκύπτουν ορισμένοι παράγοντες οι οποίοι καλό θα ήταν να ληφθούν υπόψη ώστε το ρομποτικό σύστημα να αποδίδει στον μέγιστο βαθμό. Το γεγονός ότι ο χώρος περιέχει μεγάλο αριθμό κινητών εμποδίων μετατρέπει το έργο σε μεγάλη πρόκληση. Ο λόγος για τα τραπέζια και τις καρέκλες τα οποία μετακινούνται συνεχώς είτε από φοιτητές, ή από το προσωπικό του εστιατορίου προκειμένου να καθαριστεί ο χώρος. Στην Εικόνα 7-10 παρουσιάζεται ο χώρος ενδιαφέροντος μία τυχαία ημέρα. Παρατηρείται ότι υπάρχουν τμήματα στον χώρο που το ρομπότ μπορεί

Εικόνα 7-9: Εξωτερικός χώρος εστιατορίου στα Κεντρικά Πανεπιστημίου Κύπρου.

να κινηθεί χωρίς οποιοδήποτε πρόβλημα, αλλά και σημεία στα οποία βρίσκονται μαζεμένα αρκετά τραπέζια.

Συν τοις άλλοις, στον χώρο υφίσταται και αριθμός στατικών εμποδίων όπως κορμοί δέντρων, στηρίγματα φωτιστικών, κάδοι απορριμμάτων όπως επίσης και πεζούλια που δημιουργούν λεκάνες. Παρά τις διακυμάνσεις στο έδαφος, κυρίως στις μεταβάσεις από πετρώματα σε χαλίκια, το ρομπότ δεν αντιμετώπισε ιδιαίτερο θέμα στη μετάβαση από το ένα στο άλλο. Σε αντίθετη περίπτωση είτε θα περιοριζόταν η κάλυψη σε τμήματα που το ρομπότ μπορεί να προσεγγίσει, είτε θα γινόταν χρήση των τροχών που φέρουν ελαστικά προκαλώντας άλλα ζητήματα. Η επιλογή των τροχών, οι περιορισμοί αλλά και ικανότητες που προσδίδουν αμφότεροι, αναλύονται εκτενέστερα σε μετέπειτα κεφάλαιο.

Αυτό για το οποίο αξίζει να γίνει αναφορά είναι η προσέγγιση του χώρου με σκοπό τη δημιουργία του χάρτη. Λαμβάνοντας υπόψη τον τρόπο με τον οποίο το ρομπότ αντιμετωπίζει εμπόδιο το οποίο δεν υφίσταται στον αρχικό χάρτη, θα κριθεί και η κατασκευή του χάρτη. Στην ουσία όμως δύο βασικές επιλογές βρίσκονται στο τραπέζι. Η πρώτη επιλογή παραπέμπει στη διαμόρφωση του χώρου όπως θα έπρεπε να

Εικόνα 7-10: Τμήμα εξωτερικού χώρου εστιατορίου του Πανεπιστημίου.

είναι υπό καθημερινές συνθήκες. Αυτό παραπέμπει σε ομοιόμορφη διάταξη των τραπεζιών στον χώρο προτού το ρομπότ σαρώσει την περιοχή. Η αξιοποίηση του εν λόγω χάρτη επιτρέπει τη δημιουργία μονοπατιού το οποίο δεν θα αποκλίνει σημαντικά από την πορεία που θα ακολουθήσει το ρομπότ. Ανάλογα και με τον τρόπο που θα σκιαγραφηθεί η πορεία το ρομπότ ενδέχεται να μην παρεκκλίνει της αρχικής διαδρομής ακόμη και αν ο χώρος έχει υποστεί αλλαγές όσον αφορά τη θέση των κινητών εμποδίων. Τα δύο σημεία λοιπόν, που θα καθορίσουν την αποτελεσματικότητα στην πλοήγηση είναι πρώτα η ορθή αναδιάταξη του χώρου με την τοποθέτηση των κινητών εμποδίων στην πιο πιθανή θέση που ενδέχεται να βρίσκονται και έπειτα η καταχώρηση σημείων πορείας που θα απέχουν από τα κινητά εμπόδια σε μία απόσταση ασφαλείας. Η απόσταση αυτή εξασφαλίζει ότι ακόμη και να μετακινηθούν τα τραπέζια είτε από θαμώνες, είτε από το προσωπικό, δεν θα επηρεάσουν την πλοήγηση ή τουλάχιστο θα το κάνουν στον ελάχιστο βαθμό.

Η δεύτερη επιλογή ακολουθεί κατά κάποιο τρόπο διαφορετική νοοτροπία, και υπακούει στην απομάκρυνση όλων των κινητών εμποδίων. Ο χώρος σαρώνεται και ο χάρτης που δημιουργείται περιέχει αποκλειστικά τα στατικά εμπόδια. Απ’ εκεί και πέρα ο χειριστής είτε σκιαγραφεί το μονοπάτι με βάσει την εκάστοτε διάταξη του χώρου,

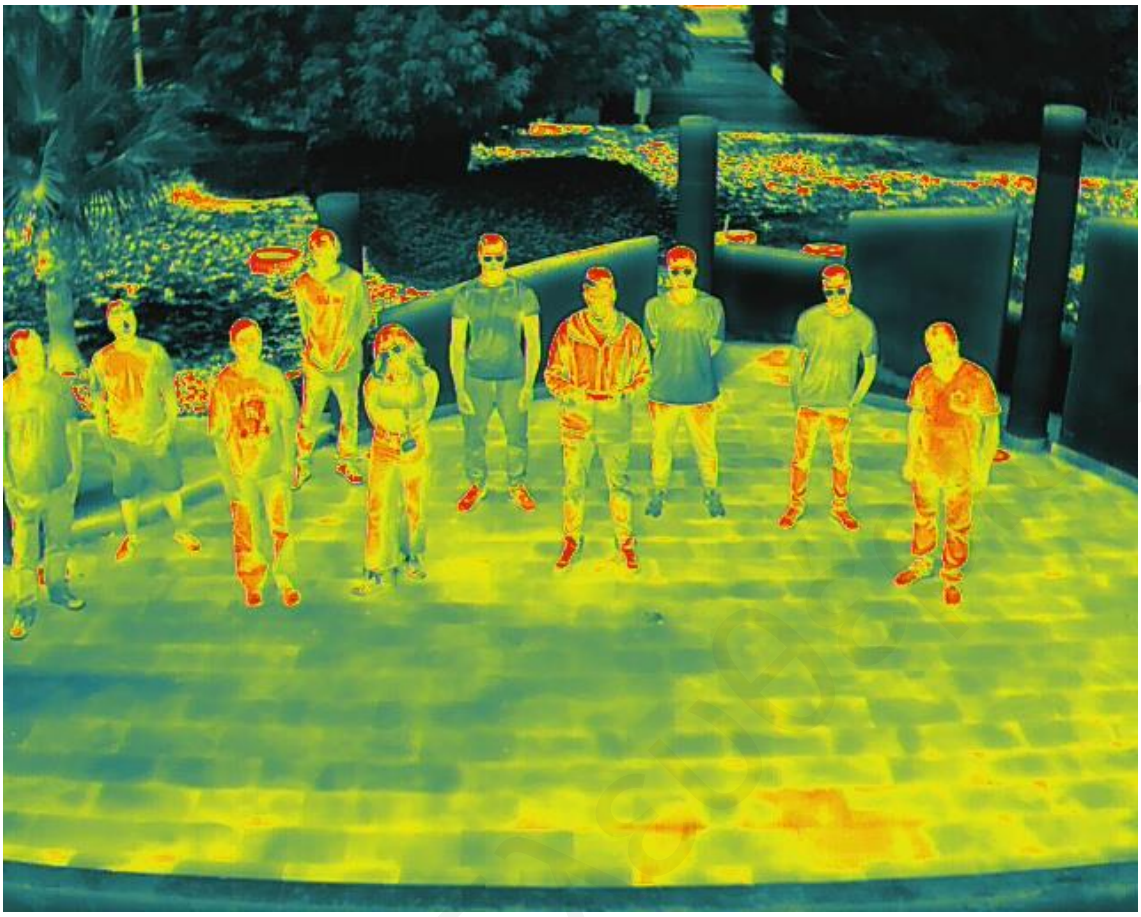
εμπειρικά κατά κάποιο τρόπο, είτε καταχωρεί μία ανεξάρτητη διαδρομή ευελπιστώντας ότι το SUMMIT δεν θα συναντήσει πληθώρα εμποδίων στη διαδρομή του. Με το που συναντά εκ παραδρομής κάποιο εμπόδιο, το ρομπότ σταματά την πλοήγηση στο επίπεδο και εκτελεί περιστροφική κίνηση αναζητώντας την καλύτερη δυνατή διαδρομή με σκοπό να επιστρέψει στο μονοπάτι, όσο το δυνατό νωρίτερα. Για τη δοκιμασία ακολουθήθηκε αυτή η προσέγγιση.

7.2.3Επιλογή 3: Χρήση θερμικής κάμερας

Η τρίτη περίπτωση υλοποιήθηκε στα πλαίσια εξεύρεσης λύσης σχετικά με την ανίχνευση κατά τις βραδινές ώρες. Παρά το γεγονός ότι ο εξοπλισμός είναι επαρκής ώστε να εξασφαλίζονται φωτογραφίες υψηλής ευκρίνειας ακόμη και σε συνθήκες χαμηλής ορατότητας, εντούτοις υπήρξε το εξής θέμα. Σε περίπτωση που κάποιος ήθελε να εισέλθει στους χώρους και γνώριζε για τα συστήματα ασφαλείας των δύο προηγούμενων περιπτώσεων πιθανό να επιχειρούσε να το πράξει σε ώρες περιορισμένης ορατότητας, είτε κατά τη διάρκεια της νύχτας, είτε υπό ομίχλη.

Ακόμη όμως και αν οι καιρικές συνθήκες επιτρέπουν την ορατότητα, υπάρχει η πιθανότητα πρόκλησης κακόβουλης βλάβης στις ηλεκτρικές εγκαταστάσεις, ή πολύ πιο απλά αποκτώντας πρόσβαση στον γενικό διακόπτη να διακοπεί η παροχή ηλεκτρικού ρεύματος έως ότου ο φύλακας το επαναφέρει.

Την λύση στο πιο πάνω ζήτημα έρχεται να δώσει η παρούσα περίπτωση εκτελώντας την λήψη από εναέρια μέσα με θερμική κάμερα. Η θερμική κάμερα αναπαριστά χρωματικά βάσει κλίμακας την θερμοκρασία σε κάθε σημείο της εικόνας. Εάν θεωρηθεί ότι κατά τις βραδινές ώρες η θερμοκρασία κυμαίνεται κοντά στους 20℃ και ότι ο ανθρώπινος οργανισμός υπό φυσιολογικές συνθήκες στους 36,7℃, τότε η διαφορά μεταξύ του ανθρώπου και των υπόλοιπων επιφανειών στην εικόνα θα είναι σαφώς ορατή. Με καλή εκπαίδευση του μοντέλου δεν θα είναι καθόλου δύσκολο να εντοπίσει τον άνθρωπο εφόσον σε σύγκριση με την υπόλοιπη εικόνα θα αναπαρίσταται με διαφορετικά χρώματα. Ακόμη όμως και σε συνθήκες καύσωνα όπου οι επιφάνειες σε μία ανοικτή περιοχή αποκτούν θερμοκρασία ίδια με το ανθρώπινο σώμα παρατηρείται εκ νέου διαφορά.



Εικόνα 7-11: Φωτογραφία ανθρώπων από θερμική κάμερα.

Στην Εικόνα 7-11 παρουσιάζεται φωτογραφία μελών του UCY Robotics Lab. Παρά την ηλιοφάνεια που επικρατεί, οι ανθρώπινες φιγούρες διακρίνονται ξεκάθαρα κάτι το οποίο εξασφαλίζει αξιοπιστία της εφαρμογής καθ' όλη την διάρκεια της ημέρας. Αξίζει επίσης να γίνει αναφορά στη διαφορετική αναπαράσταση από άνθρωπο σε άνθρωπο. Όπως μπορεί κανείς να παρατηρήσει τέσσερα από τα μέλη των ομάδων παρουσιάζονται διαφορετικά από τους υπόλοιπους, των οποίων η φιγούρα έχει κατά κύριο λόγο έντονο κόκκινο χρώμα. Ο λόγος για τον οποίο τα τέσσερα άτομα αναπαρίστανται σε πιο ήπια κλίμακα είναι η ενδυμασία που φέρουν. Ενδεδυμένοι λοιπόν από λευκά ή ανοιχτόχρωμα ρούχα απορροφούν λιγότερη ακτινοβολία, ωστόσο ακόμη και έτσι μπορούν να ανιχνευτούν από το μοντέλο.

Με αυτό τον τρόπο εξαλείφεται κάθε πιθανή περίπτωση κάλυψης του δράστη από την ενδυμασία του αφού με σκουρόχρωμο ρουχισμό οποιαδήποτε στιγμή της ημέρας μπορεί να ανιχνευτεί σε εικόνα από τη θερμική κάμερα, ενώ σε φωτογραφίες με ανοιχτόχρωμο ρουχισμό ενδέχεται η συγκεκριμένη περίπτωση να μην είναι η πλέον αξιόπιστη ωστόσο μπορεί να συνδυαστεί με τις δύο πρώτες.

Πέρα όμως από την ανίχνευση ανθρώπου η εφαρμογή αυτή δύναται να αξιοποιηθεί και σε αρκετές περιπτώσεις για σκοπούς ασφαλείας πέραν από την παράνομη είσοδο ανθρώπων σε χώρους ενδιαφέροντος. Το γεγονός ότι καταχωρείται και ακριβής θερμοκρασία για κάθε σημείο της εικόνας επιτρέπει πρόληψη και βέλτιστη διαχείριση κρίσιμων περιστατικών. Η υπερθέρμανση εξοπλισμού καθώς επίσης και τυχόν βλάβες που μπορεί να προκύψουν είναι μόνο μερικά από αυτά. Στο Κεφάλαιο 10 αναλύονται εκτενέστερα τρόποι αξιοποίησης της θερμικής κάμερας για σκοπούς πρόληψης και ασφάλειας.

Κεφάλαιο 8

8 ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ – ΧΡΗΣΗ DRONE

Σε αυτό το κεφάλαιο παρουσιάζονται πειραματικά αποτελέσματα με τη χρήση drone που αφορούν την Επιλογή 1 που παρουσιάστηκε στο Κεφάλαιο 7.

8.1 Drone

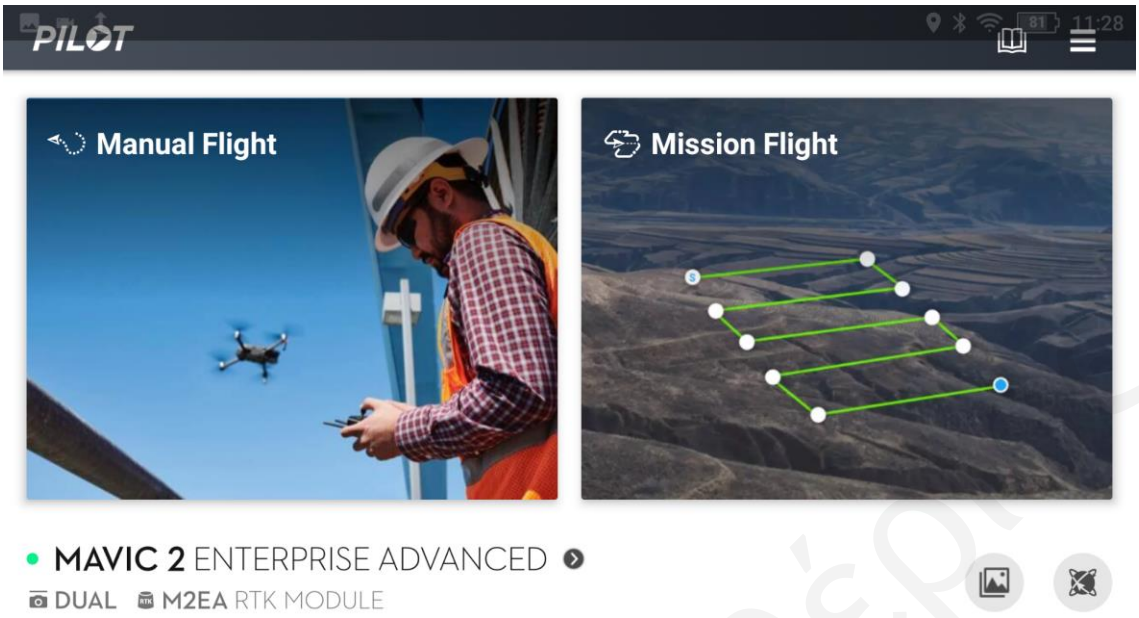
Το ρομπότ που χρησιμοποιήθηκε για την εφαρμογή είναι το MAVIC 2 (βλ. Εικόνα 8-1) και ανήκει στο UCY Robotics Lab. Τεχνικά χαρακτηριστικά παρουσιάζονται στο Παράρτημα Ι. Βασικά χαρακτηριστικά είναι η ανίχνευση εμποδίων όπως επίσης και το προηγμένο σύστημα υποβοήθηση πιλότου. Παράλληλα, υπάρχει ενσωματωμένο σύστημα αντιστάθμισης αέρα και προστασία μέσω κωδικού πρόσβασης

Εικόνα 8-1: Το MAVIC 2 με όλα τα παρελκόμενα.

για φύλαξη τόσο των δεδομένων που καταγράφονται και αποθηκεύονται, όσο και του ελέγχου του ρομπότ.

Το φορητό τηλεχειριστήριο φέρει οθόνη αφής από την οποία ο χειριστής έχει τη δυνατότητα να παρακολουθεί τα όσα καταγράφει η κάμερα. Παράλληλα μέσω της οθόνης πραγματοποιείται ένα σύνολο από λειτουργίες που χρειάζονται ώστε το ρομπότ να είναι σε θέση να καταγράφει δεδομένα και να τα στέλνει σε πραγματικό χρόνο για επεξεργασία. Η μεταφορά δεδομένων δεν κατέστη εφικτή μέσω του δικτύου που παρέχει το Πανεπιστήμιο Κύπρου λόγω περιορισμών ασφαλείας σε αυτό. Ο πιο πάνω περιορισμός οδήγησε στη λειτουργία εξωτερικού δικτύου για μεταφορά των δεδομένων κάτι που εν τέλει αποδείχθηκε κατάλληλο εφόσον λειτουργεί αυτόνομα και αποκλειστικά για σκοπούς της εφαρμογής.

Οι φωτογραφίες και τα βίντεο προβάλλονται απευθείας στην οθόνη, έτσι ο χειριστής αποκτά την ευκαιρία άμεσης αξιολόγησης των αποτελεσμάτων. Με τον τρόπο αυτό έχει την δυνατότητα παρέμβασης στην διαδικασία λήψης προκειμένου να κάνει βελτιώσεις. Σχετικά με την μεταφορά του υλικού, αυτή μπορεί να επιτευχθεί με δύο τρόπους. Είτε μέσω καλωδίου το οποίο συνδέεται σε θύρα που υπάρχει στο ρομπότ, είτε ασύρματα μέσω Bluetooth από το τηλεχειριστήριο προς την επιθυμητή συσκευή. Αξίζει να σημειωθεί ότι τα δεδομένα που συλλέγονται αποθηκεύονται στην μνήμη του ρομπότ (20Gb) και όχι σε αυτή του τηλεχειριστηρίου ή την επιπλέον κάρτα μνήμης. Λόγω του ότι το ρομπότ έχει μόλις 20Gb μνήμη, είναι ρίσκο να παραμείνει αυτή ως προκαθορισμένη επιλογή. Εάν υπάρχει κάρτα μνήμης καλό θα ήταν τα δεδομένα να πηγαίνουν κατευθείαν εκεί ή και εκεί, εφόσον υπάρχει η δυνατότητα αποθήκευσης τόσο στην μνήμη του τηλεχειριστηρίου όσο και στην κάρτα ταυτόχρονα. Παράλληλα, η κάρτα μνήμης αφαιρείται εύκολα και γρήγορα και μπορεί να εφαρμόσει αμέσως σε άλλη συσκευή όπως τηλέφωνο ή υπολογιστή, κάτι που θα επιτάχυνε και την διαδικασία μεταφοράς δεδομένων.



Εικόνα 8-2: Αρχική οθόνη τηλεχειριστηρίου MAVIC 2.

8.1.1Χειροκίνητη Πλοήγηση

Πέρα από την μεταφορά δεδομένων, χρειάζεται να καθοριστεί η διαδρομή στην οποία θα κινείται αυτόνομα το drone. Η δημιουργία πορείας πλοήγησης του ρομπότ γίνεται με σχετικά εύκολο για τον χρήστη τρόπο λόγω των ενσωματωμένων προγραμμάτων. Με το που τίθεται σε λειτουργία το χειριστήριο, η αρχική οθόνη παρουσιάζει δύο επιλογές με τον τρόπο που παρουσιάζεται στην Εικόνα 8-2. Η πρώτη επιλογή αφορά την χειροκίνητη πλοήγηση κατά την οποία ο χειριστής έχει τον απόλυτο έλεγχο του drone και με τον ανάλογο συνδυασμό κομβίων και μοχλών το καθοδηγεί να εκτελέσει την επιθυμητή πορεία. Στο πάνω μέρος του τηλεχειριστηρίου υπάρχει κομβίο για την λήψη εικόνων, όπως επίσης και για την εκκίνηση αλλά και αποθήκευση των πλάνων από την πτήση. Δίνεται παράλληλα η δυνατότητα κατεύθυνσης της κάμερας με δυνατότητα περιστροφής έως και 90°, να λαμβάνεται δηλαδή λήψη ακριβώς κάτω από τη συσκευή. Επιπρόσθετα, μέσω του τηλεχειριστηρίου μπορεί να καθοριστεί το ποσοστό εστίασης της κάμερας με μεγάλη ευκρίνεια ακόμη και για x32 εστίαση.

Οι δύο μοχλοί που βρίσκονται στο τηλεχειριστήριο επιτρέπουν την κίνηση σε δύο από τα τρία επίπεδα του τρισδιάστατου χώρου. Με τον κατάλληλο συνδυασμό κινήσεων όμως το ρομπότ μπορεί να μετακινηθεί σε οποιοδήποτε σημείο και στο τρίτο επίπεδο αποκτώντας έτσι ευελιξία προς κάθε σημείο του χώρου.

8.1.2Προκαθορισμένη Πλοήγηση

Η δεύτερη επιλογή, που εμφανίζεται στα δεξιά της οθόνης αφής έχει να κάνει με τη δημιουργία πορείας πλοήγησης η οποία μπορεί να αποθηκευτεί και να επαναχρησιμοποιηθεί οποιαδήποτε στιγμή. Η εν λόγω αποστολή, εκτός από επανεκτέλεση μπορεί να υποστεί και επεξεργασία. Εάν λόγου χάρη η πορεία στο μεγαλύτερό της μέρος αφήνει ικανοποιημένο τον χειριστή και χρειάζεται απλώς βελτιστοποίηση, τότε αυτό μπορεί να επιτευχθεί μέσω τροποποίησης των κατάλληλων παραμέτρων.

Η δημιουργία της πορείας χρίζει ιδιαίτερης αντιμετώπισης καθώς τα σημεία καταχωρούνται διαδοχικά. Πρώτα απ' όλα καταχωρείται το σημείο αναφοράς. Η αφετηρία με άλλα λόγια, από την οποία θα ξεκινήσει η πλοήγηση. Με το που δίνεται εντολή για συγκεκριμένη πορεία η οποία έχει καταχωρηθεί εκ των προτέρων, το ρομπότ απογειώνεται, εάν βρίσκεται στο έδαφος, και προσεγγίζει την αφετηρία για σκοπούς αρχικοποίησης. Ακόμη όμως και να βρίσκεται κάπου τυχαία στον εναέριο χώρο, ή στην μέση άλλης αποστολής, το ρομπότ υπακούει στην τελευταία εντολή και επικεντρώνεται στο να κατευθυνθεί στο σημείο αναφοράς με την ίδια φορά που έχει καταχωρηθεί. Πέραν της φοράς, της ταχύτητας και του ύψους, το ρομπότ συγκρατεί και την γωνία λήψης. Απαραίτητο λοιπόν είναι κατά την διάρκεια δημιουργίας της εκάστοτε αποστολής να λαμβάνονται υπόψη και να ρυθμίζονται ανάλογα όλες οι σχετικές παράμετροι.

8.2 Περιορισμοί

Λόγω της χωρικής διάταξης αλλά και άλλων παραγόντων όπως για παράδειγμα τα νομοθετικά πλαίσια και η επιρροή που έχει η συσκευή στο περιβάλλον προκύπτει ένα σύνολο περιορισμών που επεξηγούνται παρακάτω.

8.2.1Περιορισμός στο ύψος πτήσης

Η απαίτηση ταυτόχρονης κάλυψης σε δύο επίπεδα, ισόγειο και πρώτο όροφο, προσδίδει ένα ακόμη περιορισμό στο ύψος πτήσης του drone. Πέραν των περιορισμών λόγω θορύβου, καθ' όλη την διάρκεια της πτήσης χρειάζεται να υφίσταται οπτική επαφή και στα δύο επίπεδα. Αναφορικά με τον ήχο, η συσκευή κατά την διάρκεια πτήσης εκπέμπει ένα βουητό το οποίο γίνεται εντονότερο με την αύξηση της ταχύτητας και κατά την διάρκεια αλλαγής κατεύθυνσης του drone. Ο εν λόγω ήχος σε συνεχόμενη

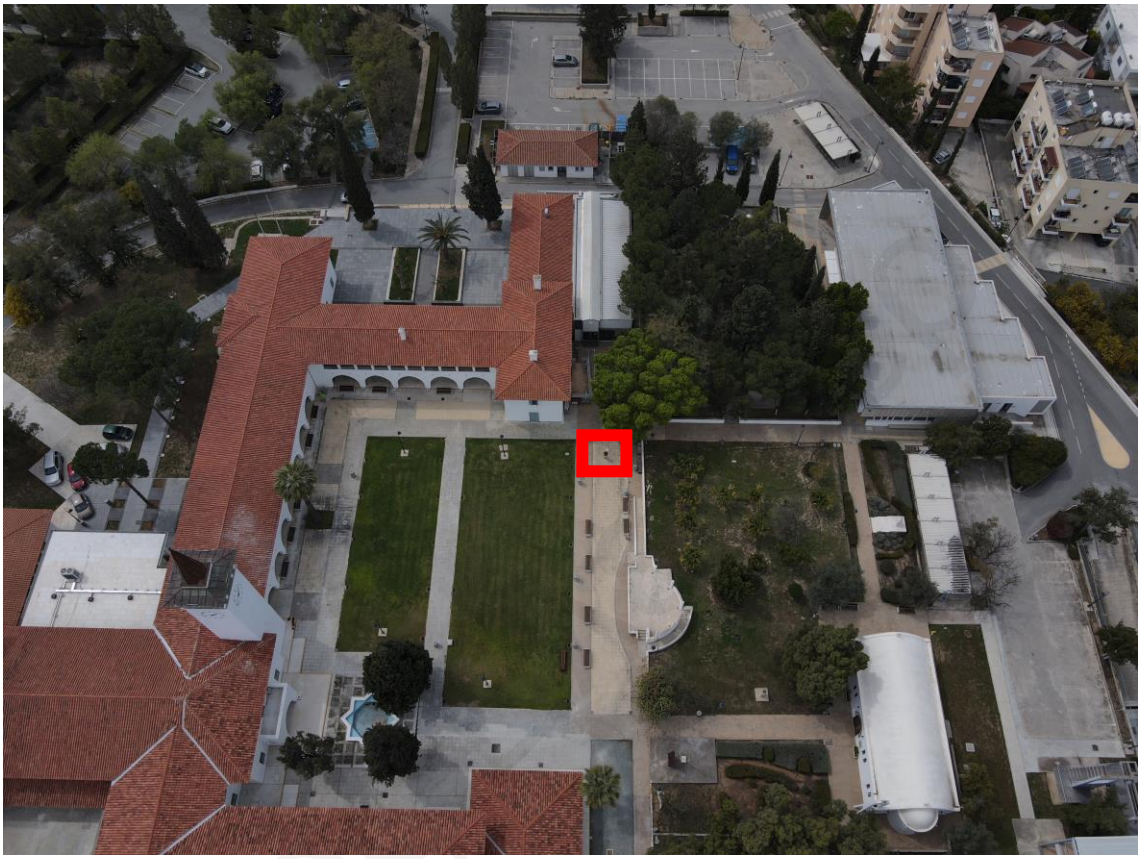
βάση μπορεί να καταστεί ενοχλητικός, ως εκ τούτου η συσκευή χρειάζεται να ίπταται πάνω από τον πρώτο όροφο για να μην επηρεάζει τις αίθουσες. Ταυτόχρονα όμως πρέπει να μην είναι πολύ ψηλά, ώστε το πλάνο που λαμβάνεται να είναι όσο το δυνατό καθαρότερο και πιο κοντινό για να γίνει η ανίχνευση όσο το δυνατό καλύτερα.

Με βάσει τα πιο πάνω διεξήχθη μία σειρά δοκιμών ώστε να βρεθεί το ελάχιστο ύψος στο οποίο ο θόρυβος να μην επηρεάζει τις αίθουσες διδασκαλίας. Ακολούθως έλαβε χώρα δεύτερη σειρά δοκιμών. Ναι μεν χρειάζονται κοντινά πλάνα, ωστόσο το drone χρειάζεται να καλύψει ολόκληρη την περιοχή με μόλις μερικά περάσματα. Αυτό δεν μπορεί να καταστεί εφικτό με χαμηλές πτήσεις αφού για να καλύψει η κάμερα όλα τα σημεία του χώρου τουλάχιστον μία φορά θα χρειαστούν αλλεπάλληλα περάσματα. Το ζήτημα με τη μεγάλη διαδρομή είναι ότι χρειάζεται σημαντικά μεγαλύτερο χρονικό διάστημα διότι εμπεριέχει αυξημένο αριθμό γωνιών τις οποίες η συσκευή χρειάζεται να περάσει προσεκτικά ώστε να μην αλλοιωθεί η ποιότητα της εικόνας. Το να χρειάζεται μία εφαρμογή σχετικά μεγάλο χρονικό διάστημα για την ολοκλήρωση ενός κύκλου μπορεί και να μην είναι ιδιαίτερα αρνητικό.

Έπειτα από σωρεία δοκιμών κρίθηκε καταλληλότερο ύψος πτήσης τα 16.4ft από το έδαφος. Από πλευράς νομοθεσίας και θορύβου δεν τίθεται κάποιο ζήτημα, ωστόσο τα αποτελέσματα από την ανίχνευση δεν ήταν και τόσο ενθαρρυντικά. Προφανώς διότι όλες οι εικόνες στο αρχικό σύνολο δεδομένων που απεικόνιζαν ανθρώπους πάρθηκαν από αρκετά πιο κοντινή απόσταση και από εντελώς διαφορετική γωνία λήψης. Στην ανίχνευση υπήρχαν σημεία που η ανθρώπινη φιγούρα εντοπιζόταν και μάλιστα είχε και υψηλή ακρίβεια. Υπήρχαν όμως και στιγμές όπου ο άνθρωπος περνούσε εντελώς απαρατήρητος από το μοντέλο, κυρίως κατά την διάρκεια όπου η λήψη γινόταν με μικρή γωνία, δηλαδή ο άνθρωπος βρισκόταν κάτω από την συσκευή και φαινόταν στην εικόνα σαν κουκίδα.

Στην Εικόνα 8-3 επισημαίνεται η παρουσία ανθρώπου στον χώρο ενδιαφέροντος. Η πλοήγηση του ρομπότ έγινε σε μια δοκιμή αρκετά πιο ψηλά από τα 16.4ft και όπως ήταν λογικό η ανθρώπινη φιγούρα κάθε άλλο παρά ευδιάκριτη είναι. Παρά το γεγονός ότι το εν λόγω ζήτημα επιλύθηκε με κατάλληλη διαμόρφωση του συνόλου δεδομένων του μοντέλου, εντούτοις θεωρήθηκε πιο φρόνιμο να υπάρχει και το κατάλληλο φωτογραφικό υλικό ώστε η παρουσία ανθρώπου στον χώρο να επιβεβαιώνεται και από φυσικό πρόσωπο. Αυτό εξ υπακούει ότι από την φωτογραφία στην οποία γίνεται η ανίχνευση από το μοντέλο, ο φρουρός ασφαλείας θα μπορεί να διακρίνει και να επιβεβαιώσει την ύπαρξη ανθρώπου στον χώρο. Την ίδια στιγμή θα

υπάρχει η δυνατότητα ελέγχου του εισβολέα για τυχόν μεταφορά επικίνδυνων αντικειμένων εντός του Πανεπιστημίου, κάτι το οποίο δεν διακρίνεται από την πανοραμική φωτογραφία στην Εικόνα 8-3.



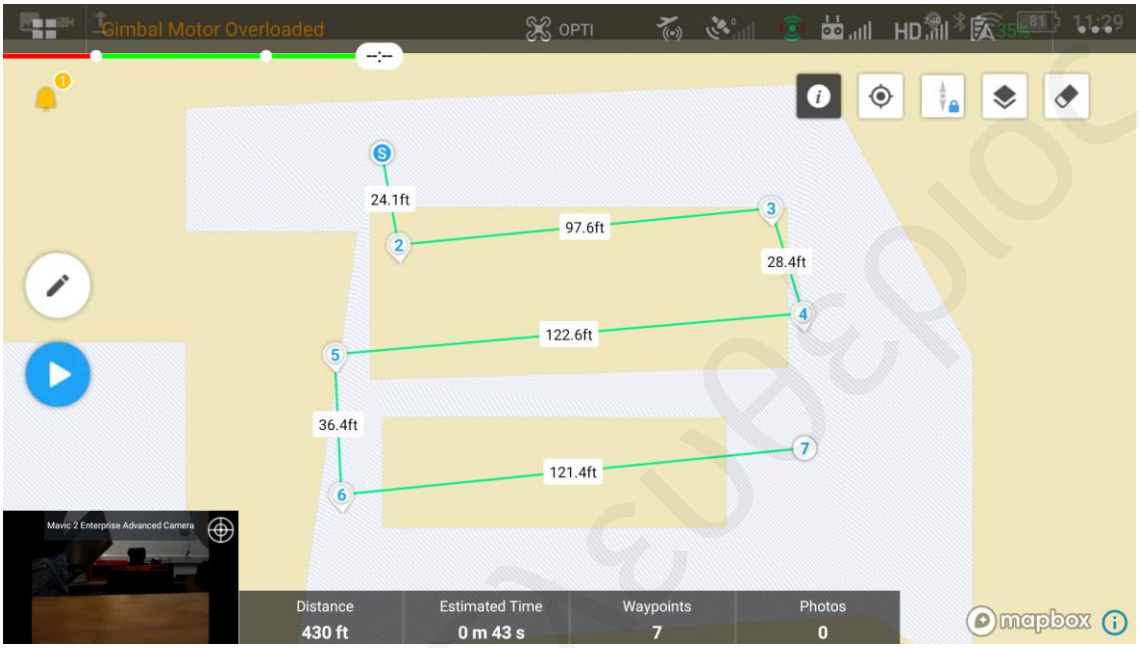
Εικόνα 8-3: Επισήμανση ανθρώπινης παρουσίας από πτήση σε μεγάλο ύψος.

8.2.2Περιορισμός στην ταχύτητα πτήσης

Πέρα από το ύψος, εξίσου σημαντική είναι και η ταχύτητα με την οποία θα κινείται το ρομπότ. Η ποιότητα των εικόνων που έχει ειπωθεί πιο πάνω είναι εξαιρετικά σημαντική για την ανίχνευση ανθρώπου στον χώρο. Υπάρχουν όμως και ορισμένοι άλλοι παράγοντες για τον καθορισμό της. Αρχής γενομένης από το χρονικό περιθώριο κάλυψης του χώρου, τον χρόνο δηλαδή που χρειάζεται ώστε να διεκπεραιωθεί ένας πλήρης κύκλος με το ρομπότ να καταλήγει εκεί απ' όπου ξεκίνησε. Εφόσον έχει βρεθεί το βέλτιστο ύψος πλοήγησης, μπορεί πλέον να καθοριστεί η διαδρομή ώστε το drone να διανύσει την ελάχιστη απόσταση καλύπτοντας κάθε γωνία του χώρου.

Έπειτα από μια σειρά λήψεων ολόκληρη η περιοχή ενδιαφέροντος δύναται να καλυφθεί με μόλις τρία ευθύγραμμα περάσματα σε διαφορετικές γραμμές του εναέριου

χώρου. Στην Εικόνα 8-4 φαίνεται η πορεία που καθορίστηκε στην οποία θα κινείται επανειλημμένα το ρομπότ. Οι λευκές περιοχές αντιστοιχούν στην πλακόστρωτη επιφάνεια του δαπέδου, ενώ τα δύο πορτοκαλί πλαίσια στα ορθογώνια τμήματα γρασιδιού.



Εικόνα 8-4: Πορεία πλοήγησης του drone.

8.2.3Περιορισμός στην γωνία λήψης

Η γωνία λήψης και κατ' επέκταση η ανίχνευση ανθρώπου σε μία εικόνα επηρεάζονται σημαντικά και από τον φωτισμό στον χώρο. Θα ήταν παράλειψη να τεθεί το ρομπότ σε μία επανειλημμένη τροχιά και να αφεθεί σε αυτή την ρουτίνα καθ' όλη την διάρκεια της ημέρας. Χρειάζεται να ληφθεί υπόψη η θέση του ηλίου ώστε η λήψη να μην επηρεάζεται. Αντίθετα, με σωστή τοποθέτηση του drone στον εναέριο χώρο, ο ήλιος μπορεί να βοηθήσει σε μεγάλο βαθμό προσφέροντας στο μοντέλο υψηλή ποιότητα εικόνων. Όπως όμως και αν γίνει η πλοήγηση, σε καμία περίπτωση η κάμερα δεν πρέπει να βρεθεί αντιμέτωπη με τις ακτίνες του ήλιου, κάτι που χρίζει ιδιαίτερης προσοχής κυρίως κατά τις πρωινές ώρες κατά την ανατολή, αλλά και τις απογευματινές κατά την δύση, περιπτώσεις στις οποίες ο ήλιος βρίσκεται στο επίπεδο της πτήσης.

Οι Εικόνες 8-5 μέχρι 8-8 παρουσιάζουν λήψη εικόνων της περιοχής ενδιαφέροντος από διαφορετικές κατευθύνσεις.

Εικόνα 8-5: Λήψη από βορειοανατολικά προς νοτιοδυτικά.

Εικόνα 8-6: Λήψη από νοτιοδυτικά προς βορειοανατολικά.

Εικόνα 8-7: Λήψη προς τα βορειοδυτικά.

Εικόνα 8-8: Λήψη προς τα νοτιοανατολικά.

8.3 Επιλογή και Διαμόρφωση Μοντέλου

Βάσει των αποτελεσμάτων στο Κεφάλαιο 3.6, αποφασίστηκε το μοντέλο να γίνει με το YOLOv8. Πέραν όμως επιλογής της έκδοσης αλγορίθμου κατέστη αναγκαίο να διεξαχθεί έρευνα τόσο για το σύνολο δεδομένων, όσο και για την ακρίβεια που απαιτεί μία τέτοια εφαρμογή ασφαλείας.

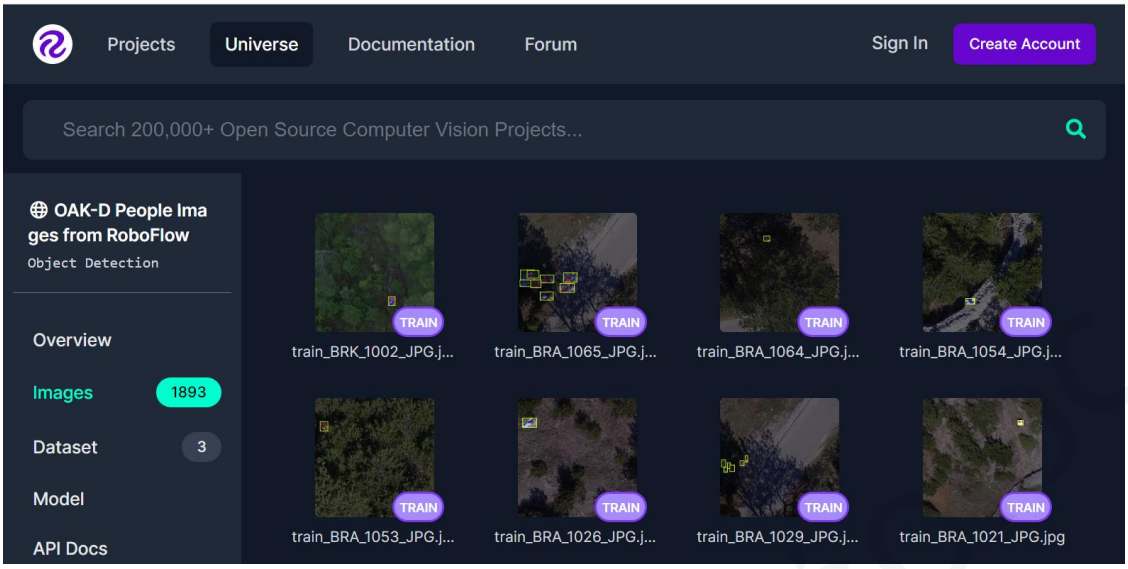
8.3.1Επιλογή Συνόλου Δεδομένων

Η πλέον κρίσιμη παράμετρος για την καλύτερη δυνατή ανίχνευση είναι τα weights που θα αξιοποιηθούν. Τα weights είτε μπορεί να παρθούν έτοιμα (pre-trained), είτε να προκύψουν μέσω ενός καλού training. Τέτοιου είδους απαιτητική εφαρμογή με τις ιδιαιτερότητες που παρουσιάζει επιβάλλει αναλυτική προεπισκόπηση και ενδελεχή έρευνα σε όσο το δυνατό περισσότερες πλατφόρμες έτοιμων συνόλων δεδομένων.

Σε πρώτη φάση χρειάστηκε να ακολουθηθεί κάπως αντίστροφα η διαδικασία εκπαίδευσης του δικτύου. Πιο αναλυτικά, μέσω εκτεταμένων λήψεων ανακτήθηκαν ολιγόλεπτα βίντεο από ποικίλες γωνίες λήψης και διαφορετική θέσπιση παραμέτρων όπως η ταχύτητα και το ύψος. Στη συνέχεια εφαρμόστηκε σε αυτά ανίχνευση χρησιμοποιώντας pre-trained weights του ιδίου του μοντέλου αρχής γενομένης με το yolov8n.pt με την πρώτη φάση της διαδικασίας να ολοκληρώνεται με χρήση του yolov8x.pt. Σε μία πρόχειρη αξιολόγηση των αποτελεσμάτων που προέκυψαν διαπιστώθηκε ότι στο σύνολό τους ήταν αρκετά φτωχά με έντονη την παρουσία αρνητικών bounding boxes.

Αδιαμφισβήτητα τα καλύτερα δυνατά αποτελέσματα προέκυψαν χρησιμοποιώντας το x-large πακέτο, ωστόσο ακόμη και σε αυτή την περίπτωση δεν ήταν ικανοποιητικά. Η λήψη κατά την διάρκεια κίνησης με εναλλαγή από γραμμική σε γωνιακή ταχύτητα και αντίστροφα, η αυξημένη απόσταση από το αντικείμενο ενδιαφέροντος, όπως επίσης και η έκθεση σε διαφορετικά επίπεδα ηλιοφάνειας είναι μερικοί από τους βασικούς παράγοντες που συνέβαλαν στην εξαγωγή υποδεέστερων αποτελεσμάτων. Η ανάγκη λοιπόν για πιο ακριβή αποτελέσματα οδήγησε στην αναζήτηση συνόλων δεδομένων με εικόνες που αν μη τι άλλο να έχουν ληφθεί από παρόμοια γωνία λήψης.

Σε μια προσπάθεια αναζήτησης έτοιμων συνόλων δεδομένων διαπιστώθηκε ότι η πλατφόρμα Roboflow περιλαμβάνει μερικά. Με μια γρήγορη ματιά επιλέχθηκε σύνολο δεδομένων πέραν των 4500 εικόνων με τις πλείστες από αυτές να έχουν ληφθεί είτε από drone, είτε από δορυφόρο, ενώ περιλαμβάνει και αρκετές λήψεις από κάμερες



Εικόνα 8-9: Σύνολο δεδομένων με λήψεις από εναέρια μέσα [24].

ασφαλείας. Όλως παραδόξως από το συγκεκριμένο σύνολο δεδομένων δεν προέκυψαν weights που να προσδίδουν ικανοποιητικά αποτελέσματα σχετικά με ανίχνευση σε ολιγόλεπτα βίντεο. Σε μια προσπάθεια επίλυσης του ζητήματος αυτού έγινε διαμόρφωση των παραμέτρων, χωρίς όμως αποτέλεσμα. Ο λόγος απόκλισης του επιθυμητού αποτελέσματος από το πραγματικό, σε τόσο έντονο βαθμό, είναι η πλειοψηφία του συνόλου δεδομένων που εστιάζει σε επισήμανση ανθρώπινης ύπαρξης από απόσταση δραματικά μεγαλύτερη από αυτή της εφαρμογής. Αυτό διαπιστώθηκε από το ότι το μοντέλο μετά την εκπαίδευση εντόπιζε ανθρώπινες φιγούρες στα 300 – 400 μέτρα με σχετικά καλή ακρίβεια, την ίδια στιγμή που αγνοούσε ανθρώπους μόλις μερικά μέτρα μακριά.

Κατόπιν μελέτης των αποτελεσμάτων εκάστων επιλογών προέκυψε η ιδέα συνένωσης των δύο τους. Το γεγονός ότι το RoboCoco σύνολο δεδομένων του μοντέλου εντόπιζε την ύπαρξη αντικειμένων ενδιαφέροντος σε κοντινή και ευδιάκριτη εικόνα, σε συνδυασμό με την αποκλειστικά απομακρυσμένη ανίχνευση του συνόλου στην Εικόνα 8-9, επέτρεπε τουλάχιστον να γίνει μία προσπάθεια εξαγωγής weights συνδυάζοντας τα δύο σύνολα στην εκπαίδευση. Ακόμη και τότε όμως τα αποτελέσματα δεν ήταν ακριβή στον απαιτούμενο βαθμό. Υπήρχαν πολλές στιγμές κατά τη διάρκεια του βίντεο στις οποίες παρουσιάζονταν αρνητικά bounding boxes.

Οριστική λύση στο πρόβλημα προήλθε με την εκπαίδευση του μοντέλου σε υλικό από τον χώρο ενδιαφέροντος. Μετά από αλλεπάλληλες προσπάθειες, χρησιμοποιώντας έτοιμα σύνολα δεδομένων, με αποτελέσματα κατώτερα των περιστάσεων η μόνη πιθανή λύση ήταν η εκπαίδευση του μοντέλου σε πανομοιότυπο

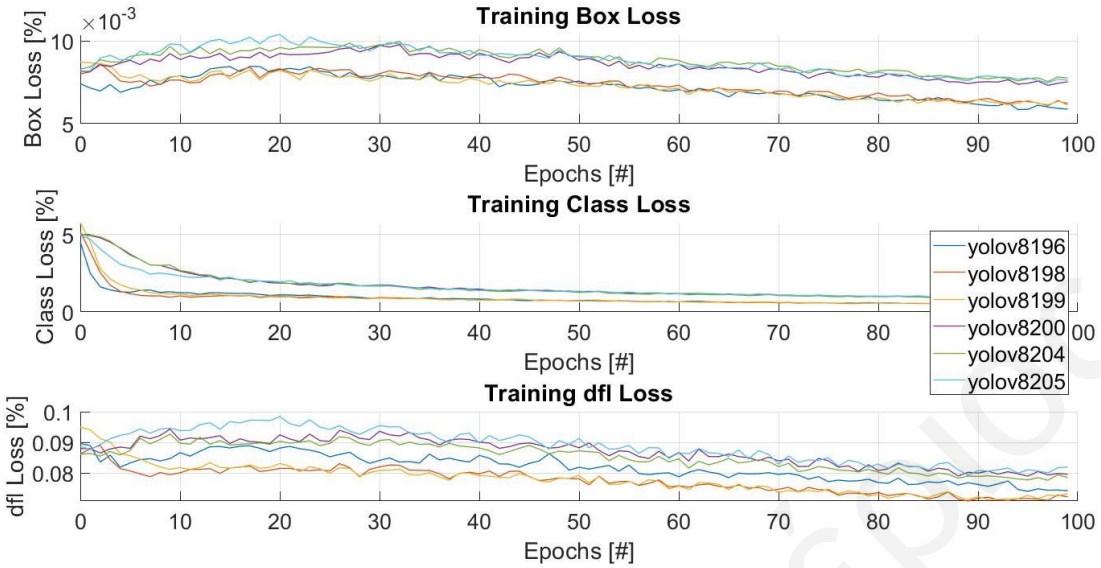
υλικό όχι όμως στον ίδιο χώρο. Για τον σκοπό αυτό χρησιμοποιήθηκαν τα ολιγόλεπτα βίντεο από τις δοκιμές, τα οποία με την σειρά τους χωρίστηκαν σε frames και δημιουργήθηκαν ετικέτες για το καθένα από αυτά.

Αξιοποιώντας στο έπακρο το περιεχόμενο του συνόλου δεδομένων RoboCoco, σε συνδυασμό με μέρος του συνόλου OAK-D People Images from Roboflow [24] οι εικόνες από τις δοκιμές ήρθαν να ολοκληρώσουν το πακέτο καλύπτοντας κάθε πτυχή των λήψεων. Τα αποτελέσματα, που παρουσιάζονται αναλυτικά στη συνέχεια είναι αρκετά ακριβή και με μεγάλο ποσοστό αξιοπιστίας.

8.3.2Καθορισμός Παραμέτρων

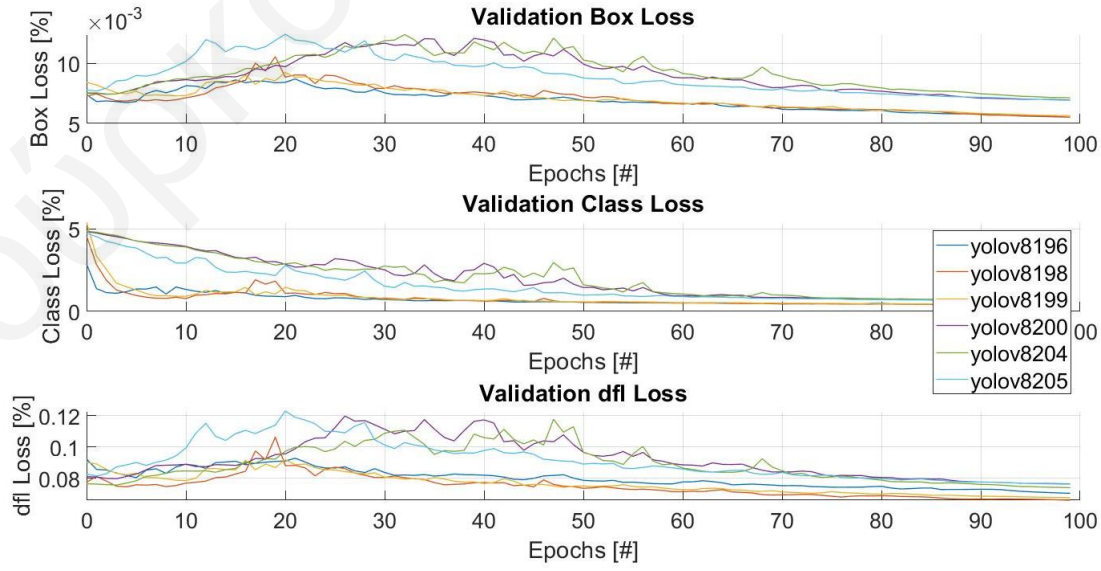
Σε εφαρμογές ασφαλείας όπως η συγκεκριμένη επιβάλλεται ακρίβεια, αξιοπιστία και αμεσότητα. Σε μια προσπάθεια υλοποίησης του καλύτερου δυνατού μοντέλου προηγήθηκε μία σειρά από trainings του δικτύου υπό διαφορετικές παραμέτρους και hyperparameters. Από τη σύγκριση μεταξύ των τριών εκδόσεων του YOLO βάσει της δημοσίευσης [63], μία από τις βασικές διαφορές του YOLOv8 σε σύγκριση με τα YOLOv5 και YOLOv7 είναι ότι στο πρώτο είναι πιο εμφανής ο κίνδυνος over-train. Χρειάζεται λοιπόν να προηγηθεί προσεκτική εκπαίδευση του μοντέλου ούτως ώστε αυτά να μπορούν να αξιοποιηθούν σωστά στη συνέχεια.

Έπειτα από μία σειρά δοκιμών μεταβολής παραμέτρων αξιολογήθηκαν τα αποτελέσματα της εκπαίδευσης κάθε περίπτωσης και αφού αποκλείστηκαν όσα με γυμνό μάτι δεν ικανοποιούσαν τις απαιτήσεις απέμειναν ένα σύνολο μόλις μερικών. Κατόπιν, τα αποτελέσματα αυτά συγκρίθηκαν μεταξύ τους με χρήση του λογισμικού MatLab ώστε να διαφανεί η καλύτερη δυνατή επιλογή. Στις Εικόνες 8-10 και 8-11 παρουσιάζονται οι γραφικές παραστάσεις των αποτελεσμάτων όπως αυτά προέκυψαν από την εκάστοτε εκπαίδευση και επικύρωση. Στο υπόμνημα κάθε γραφήματος αναγράφεται ο κωδικός αριθμός κάθε περίπτωσης. Στο Παράρτημα IV – Πίνακας Σύγκρισης Παραμέτρων YOLOv8 επισυνάπτονται οι Πίνακες 1-3 με τις παραμέτρους της κάθε περίπτωσης και η αντιστοίχιση προκύπτει μέσω του κωδικού αριθμού από κάθε εκπαίδευση.

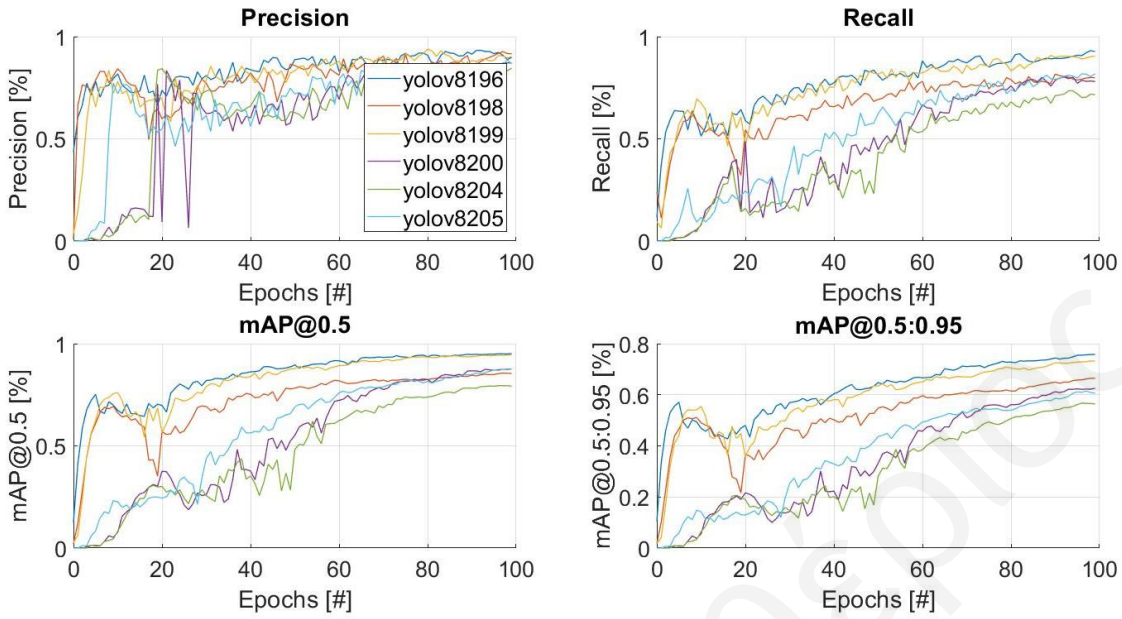


Εικόνα 8-10: Training Box, Class, dfl Loss.

Τα αποτελέσματα της εκπαίδευσης και επιβεβαίωσης επιτρέπουν την απόρριψη των 200 και 204 λόγω των υψηλότερων τιμών που παρουσιάζουν σε Box Loss και Class Loss, αντίστοιχα. Οι τιμές των υπολοίπων συγκλίνουν, έτσι για να προκύψει η καλύτερη δυνατή επιλογή χρειάζεται να ληφθούν υπόψη τα γραφήματα ακρίβειας στην Εικόνα 8-12.



Εικόνα 8-11: Validation Box, Class, dfl Loss.



Εικόνα 8-12: Precision, Recall, mAP plots.

Με μία προσεκτική ματιά στα γραφήματα ακρίβειας διαπιστώνεται ότι η καλύτερη δυνατή επιλογή είναι οι καμπύλες του συνόλου 196. Σημειώνεται ότι η συγκεκριμένη περίπτωση μαζί με την 198 είναι οι μόνες που έχουν τρέξει με το μεγάλο πακέτο pre-trained weights. Στις υπόλοιπες χρησιμοποιήθηκαν είτε το small ή το nano.

8.4 Αποτελέσματα

Τα αποτελέσματα που προέκυψαν από τον εντοπισμό ανθρώπων σε πραγματικό χρόνο μετατράπηκαν σε ταινίες μικρού μήκους και αποθηκεύτηκαν για σκοπούς αξιολόγησης. Στη συνέχεια ακολουθούν μερικά στιγμιότυπα οθόνης που πάρθηκαν κατά τη διάρκεια της ανίχνευσης. Τα στιγμιότυπα πάρθηκαν τυχαία σε πραγματικό χρόνο, ενώ παράλειψη θα αποτελούσε να μην αναφερθεί το γεγονός ότι υπάρχουν λήψεις από διαφορετικές μέρες και ώρες.

Εικόνα 8-13: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο από μακρινή απόσταση.

Εικόνα 8-14: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο από μεγάλο ύψος.

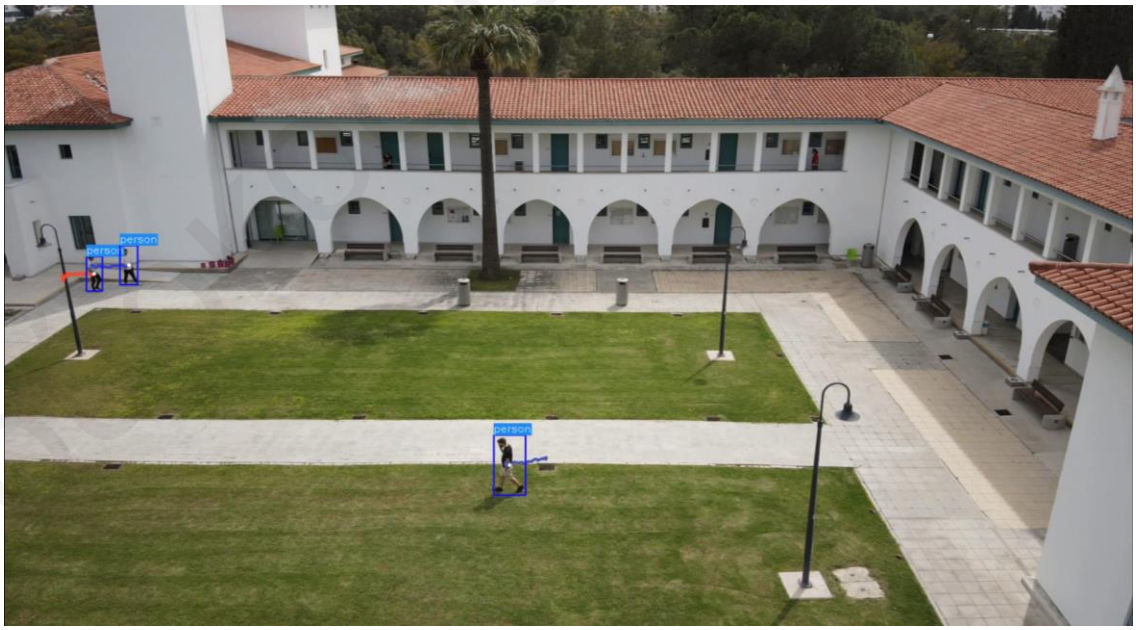
Εικόνα 8-15: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο με άλλα αντικείμενα στο προσκήνιο.

Εικόνα 8-16: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο και διατήρηση του bounding box (το επιβεβαιώνουν οι γραμμές πορείας).

Εικόνα 8-17: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο εστιάζοντας στον πρώτο όροφο.

Εικόνα 8-18: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο από διαγώνια λήψη του πρώτου ορόφου.

Εικόνα 8-19: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο διαφορετικών κατηγοριών.



Εικόνα 8-20: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο σε μακρινή και κοντινή απόσταση.

Εικόνα 8-21: Στιγμιότυπο από εντοπισμό σε πραγματικό χρόνο σε έντονη ηλιοφάνεια.

Σε μια σειρά από εννέα τυχαία στιγμιότυπα που έχουν δειχτεί παραπάνω, το να μην υπάρξει κανένα αρνητικό bounding box σημαίνει ότι η εκπαίδευση που έχει τύχει το μοντέλο ήταν πραγματικά καλή. Το συγκεκριμένο σύνολο δεδομένων δημιουργήθηκε με σκοπό την ανίχνευση ανθρώπου, ωστόσο υπάρχουν ακόμη 80 κατηγορίες που μπορεί να εντοπίσει. Κάποιες από αυτές εντοπίζονται κιόλας σε ορισμένα στιγμιότυπα. Ο λόγος για τα παγκάκια και το ρολόι κατά κύριο λόγο. Το γεγονός όμως ότι κατά τον εμπλουτισμό του συνόλου δεδομένων RoboCoco, τα επιπλέον αρχεία αφορούσαν αποκλειστικά την κατηγορία «άνθρωπος», οδήγησε σε ένα σύνολο δεδομένων «με προτίμηση». Αυτό πάει να πει ότι το δίκτυο εντοπίζει με μεγαλύτερη ευκολία τη συγκεκριμένη κατηγορία αντίθετα με τις υπόλοιπες, κάτι που έχει διαφανεί και μέσω των αποτελεσμάτων.

Αξιοσημείωτο των αποτελεσμάτων είναι η γραμμή κίνησης των αντικειμένων ενδιαφέροντος με σημείο αναφοράς τις εκάστοτε συντεταγμένες του drone. Η εν λόγω πληροφορία που ανακτάται από τις γραμμές αυτές είναι εξαιρετικά σημαντική.

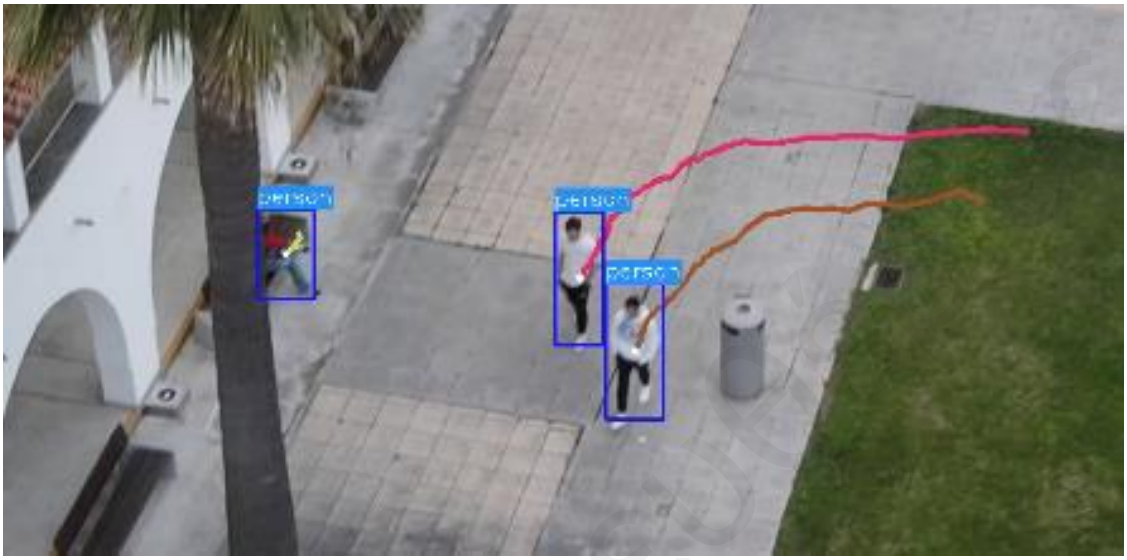
Αρχής γενομένης από την πορεία κίνησης, με σταθερή κάμερα μπορεί να αποτυπωθεί σε ένα μόνο στιγμιότυπο ολόκληρη η πορεία κίνησης του ανθρώπου στον χώρο. Σαφώς με αυτό τον τρόπο ο αρμόδιος λειτουργός δύναται να έχει μία πανοραμική αποτύπωση των κινήσεων του δράστη. Παράλληλα, ολόκληρη η πληροφορία συγκροτείται σε ένα μόνο αρχείο μικρού όγκου και μπορεί να διαβιβαστεί γρήγορα και εύκολα. Εν συνεχεία ο παραλήπτης μπορεί να αντιληφθεί σε μικρό χρονικό διάστημα

την κρισιμότητα του περιστατικού έχοντας έτσι την ευκαιρία άμεσης και αποτελεσματικής υποστήριξης.

Πέραν όμως από το προφανές που είναι η αποτύπωση της κίνησης, με την λειτουργία αυτή διασφαλίζεται μία ακόμη πληροφορία. Την ποσότητα των αντικειμένων ενδιαφέροντος στον χώρο. Αυτό μπορεί να γίνει πλήρως κατανοητό με την βοήθεια της Εικόνας 8-22. Από την εικόνα μπορεί κανείς να εξαγάγει την πορεία που διένυσαν τα δύο παιδιά με τις λευκές μπλούζες όπως επίσης και το γεγονός ότι ο ποδηλάτης είναι ακίνητος. Ενδιαφέρον είναι το ότι ο άνθρωπος με την κόκκινη μπλούζα ενώ αρχικά βρισκόταν σε ανοικτή για το drone θέα, την στιγμή που τραβήχτηκε το στιγμιότυπο τοποθετείται πίσω από το δέντρο σε σημείο που το δίκτυο δεν είναι ικανό να τον εντοπίσει, κάτι που φαίνεται και από την απώλεια του bounding box. Παρόλα αυτά όμως η γραμμή κίνησης διατηρείται στο τελευταίο σημείο που εντοπίστηκε ο άνθρωπος από το μοντέλο. Όταν το μοντέλο εντοπίσει ξανά το ίδιο αντικείμενο αναγνωρίζει από την υφιστάμενη γραμμή κίνησης ότι πρόκειται για τον ίδιο άνθρωπο

Εικόνα 8-22: Διατήρηση γραμμής πορείας κίνησης.

που απώλεσε προ λίγου, όπως φαίνεται στην Εικόνα 8-23. Έτσι, το χρώμα στην γραμμή κίνησης παραμένει το ίδιο με πριν, κάτι που επιβεβαιώνει την διατήρηση παρουσίας του ιδίου ατόμου. Με τον τρόπο αυτό γίνεται γνωστό το πόσοι άνθρωποι υπάρχουν ανά πάσα στιγμή στον χώρο και αν είναι οι ίδιοι κάθε φορά.



Εικόνα 8-23 Διατήρηση γραμμής πορείας κίνησης με την εφαρμογή bounding box.

Όσον αφορά την συγκεκριμένη λειτουργία μπορεί επίσης να αποτελέσει την βάση για υλοποίηση μοντέλου καταγραφής διερχομένων. Αυτό μπορεί να εφαρμοστεί τόσο σε παρόμοια περίπτωση όσο και σε εντελώς διαφορετικό περιβάλλον. Τον τελευταίο καιρό έχει εφαρμοστεί σε αρκετά σημεία τόσο της Λευκωσίας. όσο και άλλων επαρχιών με σκοπό την καταγραφή της κίνησης των οχημάτων και ενημέρωση του οδικού δικτύου για τη συμφόρηση στην εκάστοτε περιοχή.

8.5 Αξιολόγηση Εφαρμογής

Μεταξύ άλλων παρατηρήθηκε ο εντοπισμός και επισήμανση τόσο ανθρώπων όσο και άλλων αντικειμένων. Η ανίχνευση ανθρώπων έγινε σχεδόν με απόλυτη ακρίβεια και ελάχιστα αρνητικά bounding boxes ανά αραιά χρονικά διαστήματα. Στο σημείο αυτό να σημειωθεί ότι λόγω της φύσης της εφαρμογής και των υψηλών επιπέδων ασφαλείας, είναι προτιμότερο να εντοπίζονται λανθασμένα μία στο τόσο άνθρωποι χωρίς να υπάρχουν, από το να μην εντοπίζονται καθόλου ενώ πραγματικά υπάρχουν. Αυτό βέβαια αποτελεί λεπτομέρεια εντούτοις καλό θα ήταν να ληφθεί υπόψη.

Το πιο σημαντικό επίτευγμα που προέκυψε και επιβεβαιώνει την άρτια δουλειά που έχει γίνει κατά την εκπαίδευση είναι ο συνεχής εντοπισμός καθ’ όλη την διάρκεια της λήψης σε πραγματικό χρόνο. Ακόμη όμως και οι μεμονωμένες περιπτώσεις που το μοντέλο αδυνατεί να εντοπίσει το αντικείμενο για κάποιο στιγμιότυπο, το bounding box ανακτάται αμέσως και το κενό αντισταθμίζεται από την γραμμή πορείες με τον τρόπο που περιγράφεται στην Ενότητα 8.4 Αποτελέσματα.

Κεφάλαιο 9

9 ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ – ΧΡΗΣΗ ΚΙΝΗΤΟΥ ΡΟΜΠΟΤ

Σε αυτό το κεφάλαιο παρουσιάζεται η πειραματική δοκιμασία και τα αποτελέσματα που αφορούν τη χρήση κινητού τροχοφόρου ρομπότ. Αναφέρεται στην Επιλογή 2 που παρουσιάστηκε στο Κεφάλαιο 7.

9.1 SUMMIT XL

Το συγκεκριμένο ρομπότ αποκτήθηκε από την Robotnik έπειτα από παραγγελία και είναι ιδιοκτησία του εργαστηρίου ρομποτικής του Τμήματος Μηχανικών Μηχανολογίας και Κατασκευαστικής του Πανεπιστημίου Κύπρου. Το μοντέλο του είναι το SUMMIT XL GEN, έκδοση η οποία περιλαμβάνει εφαρμοζόμενο βραχίονα. Για τη συγκεκριμένη εφαρμογή ωστόσο ο βραχίονας δεν συμπεριλήφθηκε στο ρομποτικό σύστημα.

Παρόλο που το SUMMIT περιέχει ένα ολοκληρωμένο σύνολο αισθητήρων προκειμένου να εκτελεί βασικές λειτουργίες όπως η χαρτογράφηση και ανίχνευση εμποδίων, στην περίπτωση αυτή δεν παρουσιάζει στην φαρέτρα του όλους τους αισθητήρες για τους οποίους έγινε αναφορά στο Κεφάλαιο 5. Εντούτοις τα 2D laser sensor, 3D camera sensor σε συνδυασμό με τους αισθητήρες υπέρηχων είναι ικανά να εξάγουν ένα άκρως ικανοποιητικό αποτέλεσμα.

9.2 Συνδεσιμότητα του ρομπότ SUMMIT XL και υπολογιστή

Ο υπολογιστής που χρησιμοποιήθηκε για τον έλεγχο του ρομπότ ήταν φορητός υπολογιστής ASUS με επεξεργαστή intel CORE i7 8th Gen και κάρτα γραφικών NVIDIA GEFORCE GTX 1080. Τεχνικά χαρακτηριστικά του υπολογιστή παρουσιάζονται αναλυτικά στο Παράρτημα Χ.

Η σύνδεση μεταξύ υπολογιστή και ρομπότ επιτεύχθηκε ασύρματα μέσω δικτύου Wi-fi, ενώ χρειάστηκε να καταχωρηθούν τα σωστά IP address στα /etc/hosts files

έκαστων. Περισσότερες πληροφορίες για τη διαδικασία σύνδεσης παρατίθενται στο Παράρτημα VII – Network Configuration – Remote PC. Αυτό που χρειάζεται να προσεχθεί κατά τη διάρκεια της πρώτης σύνδεσης είναι ο υπολογιστής να ενωθεί στο δίκτυο του ρομπότ και παράλληλα σε δεύτερο δίκτυο, οι περιορισμοί του οποίου να μην επηρεάζουν τη διαδικασία. Για παράδειγμα η πρώτη σύνδεση δεν μπόρεσε να γίνει με τον υπολογιστή ενωμένο στο δίκτυο του Πανεπιστημίου λόγω των περιορισμών, ωστόσο με την ολοκλήρωση της πρώτης σύνδεσης μπορεί να χρησιμοποιηθεί στη συνέχεια οποιοδήποτε δίκτυο για ανταλλαγή πληροφοριών.

Κάτι που επίσης χρειάζεται να σημειωθεί στο σημείο αυτό είναι το γεγονός ότι και το ρομπότ είναι εξοπλισμένο με δικό του υπολογιστή. Χρειάζεται λοιπόν οι εκδόσεις των λογισμικών τόσο του φορητού υπολογιστή, όσο και του ρομπότ να συμβαδίζουν προκειμένου να λειτουργούν αφού μεταφερθούν δεδομένα από τον ένα υπολογιστή στον άλλο. Νοουμένου ότι το ρομπότ δεν έχει οθόνη, η ολοκλήρωση της διαδικασίας έγινε στον φορητό υπολογιστή. Ακολούθως μεταφέρθηκαν τα πακέτα στον υπολογιστή του SUMMIT και αφού επιτεύχθηκε η σύνδεση μέσω του φορητού υπολογιστή κατέστη δυνατό να τρέξουν οι κώδικες στον υπολογιστή του ρομπότ.

Η διαδικασία μεταφοράς δεδομένων είναι συγκεκριμένη και εξαιρετικά σημαντική αφού εκτός του ότι τα πακέτα χρειάζεται να μεταφερθούν στη σωστή μορφή, είναι πολύ σημαντικό να αποθηκευτούν και στη σωστή θέση. Σε αντίθετη περίπτωση όταν ο χρήστης επιχειρήσει να τρέξει ένα κώδικα θα εμφανίζεται σφάλμα στην οθόνη με την περιγραφή ότι το συγκεκριμένο αρχείο δεν υφίσταται. Η μεταφορά των δεδομένων στη συγκεκριμένη περίπτωση έγινε ξεχωριστά για κάθε πακέτο με τον τρόπο που φαίνεται στην Εικόνα 9-1.

```
user@user-Zephyrus-GX501GI:~$ scp -r /home/user/catkin_ws/src/follow_waypoints r
obot@SXL00-200708AA:/catkin_ws/src/follow_waypoints
```

Εικόνα 9-1: Εντολή μεταφοράς δεδομένων από τον φορητό υπολογιστή στο ρομπότ.

9.3 Περιγραφή χώρου ενδιαφέροντος

Ο χώρος ενδιαφέροντος είναι ο εξωτερικός χώρος του εστιατορίου του Πανεπιστημίου. Συνοψίζοντας τα όσα ειπώθηκαν προηγουμένως, είναι μία ανοικτή δομή με σημαντικές ιδιαιτερότητες. Το πιο βασικό είναι ο μεγάλος αριθμός δέντρων

που βρίσκονται σκορπισμένα στον χώρο με τις φυλλωσιές τους να αναμιγνύονται και να καλύπτουν σχεδόν ολόκληρο τον εναέριο χώρο. Πέραν των δέντρων αλλά και κάποιων θάμνων περιμετρικά, υφίστανται στον χώρο κάδοι απορριμμάτων, βάσεις στήριξης των φωτιστικών ενώ τα τσιμεντένια περιτοιχίσματα των λεκανών συμπληρώνουν το πακέτο των σταθερών εμποδίων. Όσον αφορά τα κινητά εμπόδια, αυτά μπορεί να είναι είτε καρέκλες, είτε τραπέζια. Στην Εικόνα 9-2 παρουσιάζονται αρκετά από αυτά.

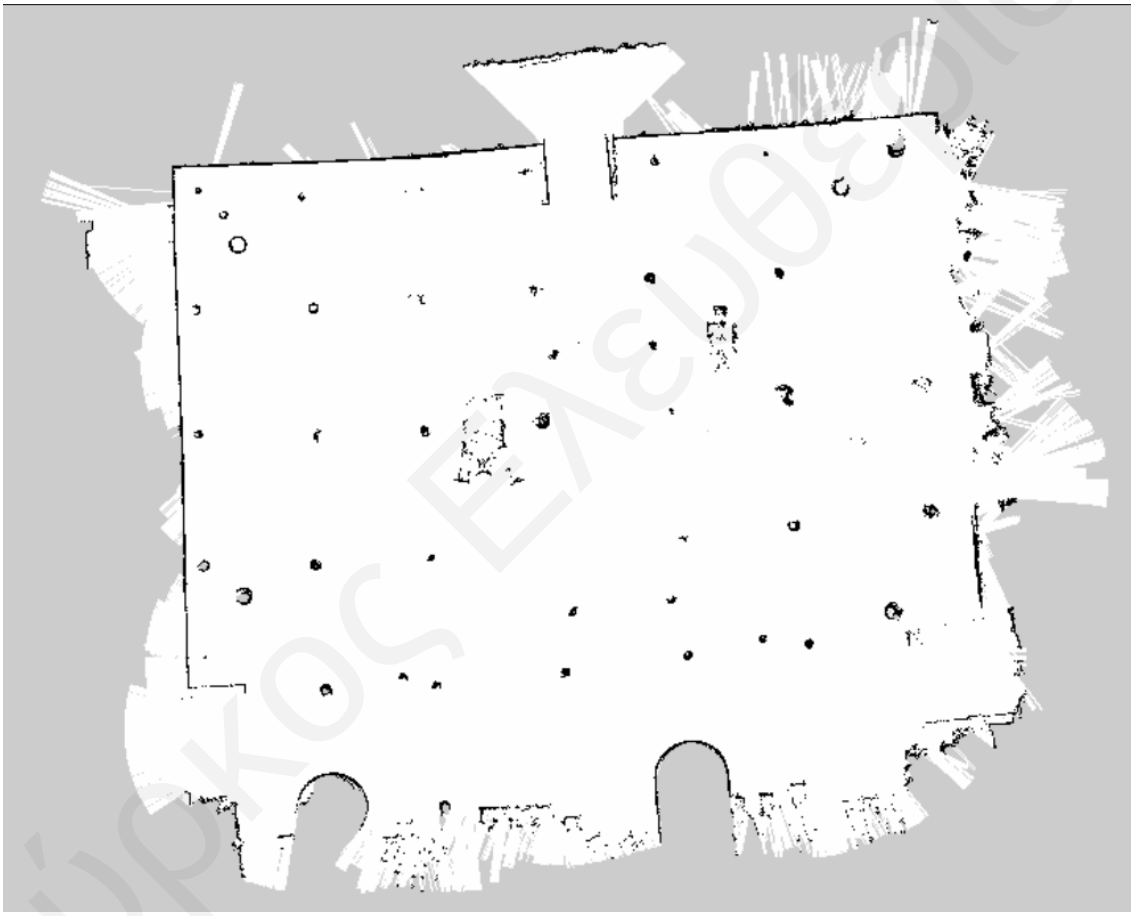
Εικόνα 9-2: Τμήμα περιοχής ενδιαφέροντος (εξωτερικός χώρος εστιατορίου Πανεπιστημίου Κύπρου).

9.4 Δημιουργία Χάρτη

Για τη χαρτογράφηση της περιοχής ακολουθήθηκε η διαδικασία, όπως αυτή περιεγράφηκε στο Κεφάλαιο 6. Τρέχοντας τον κώδικα start_mapping.launch (βλ. Παράρτημα Ι) και με σημείο αναφοράς το κέντρο περίπου της αυλής, το ρομπότ περιηγήθηκε χειροκίνητα στον χώρο. Λόγω των αυξημένων και άτακτα κατανεμημένων εμποδίων, το SUMMIT χρειάστηκε να επαναλάβει τη διαδρομή μερικές επιπλέον φορές

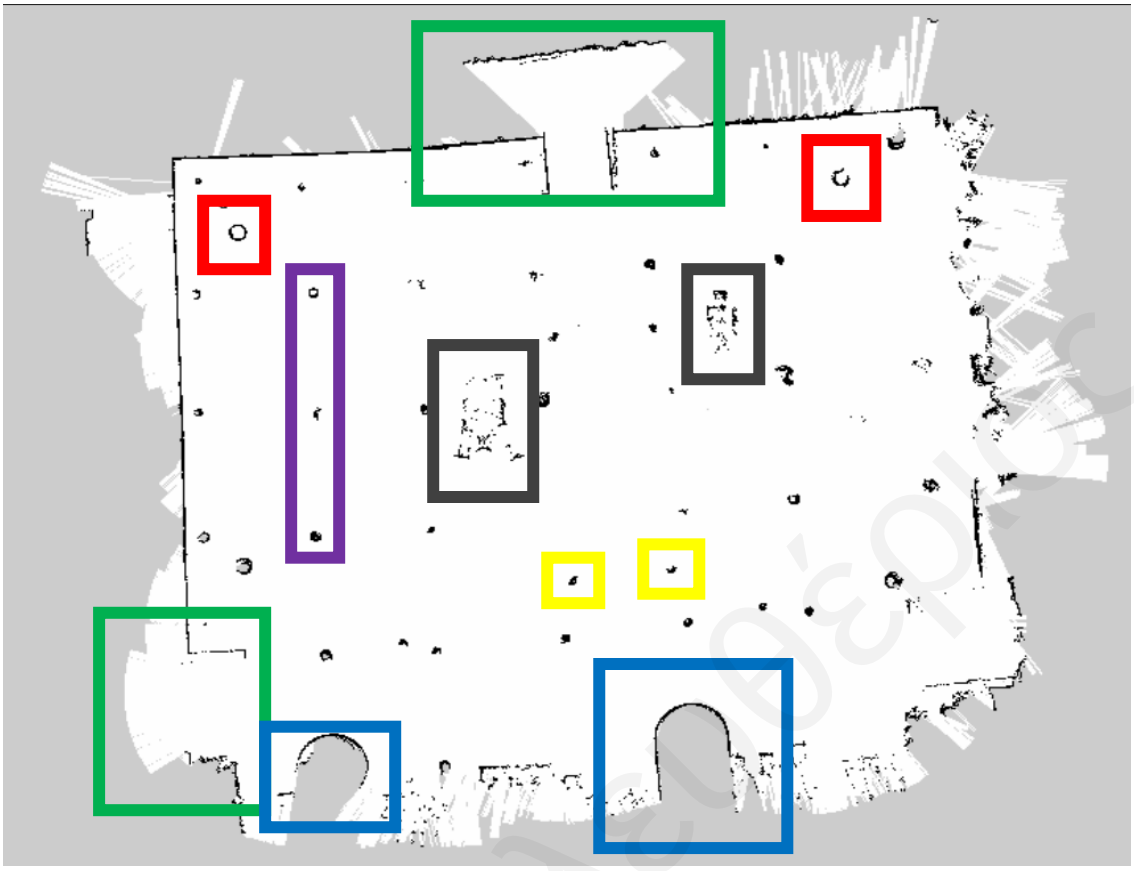
ώστε να προκύψει ένα ευδιάκριτο αποτέλεσμα. Πολύ βοηθητικό για τη δημιουργία του χάρτη ήταν το εξωτερικό περιτοίχισμα το οποίο καθορίζει τα όρια της αυλής και αποτυπώνεται με ακρίβεια στον χάρτη (βλ. Εικόνα 9-3).

Υπήρχαν βέβαια και σημεία όπου το ρομπότ χρειάστηκε να επαναλάβει τη χαρτογράφηση περισσότερες φορές προκειμένου ο χάρτης να είναι ομοιόμορφος χωρίς ατέλειες. Σημεία όπως οι γωνίες της περιοχής, τα δύο ανοίγματα (ράμπες) βορειοανατολικά και νοτιοανατολικά της περιοχής ενδιαφέροντος είναι μερικά από αυτά. Οι εν λόγω ιδιαιτερότητες αποτυπώνονται στην Εικόνα 9-4.



Εικόνα 9-3: Χάρτης εξωτερικού χώρου εστιατορίου του Πανεπιστημίου (Καλλιπόλεως).

Σε μια προσπάθεια περιγραφής της διαδικασίας κατασκευής του χάρτη κατά τη διάρκεια της περιήγησης μέσω του τηλεχειριστηρίου, στις Εικόνες 9-5 με 9-12 ξεδιπλώνεται η πορεία υλοποίησης εστιάζοντας σε σημεία τα οποία χρειάστηκαν σημαντικό αριθμό επαναλήψεων, ώστε να αποτυπωθούν με ακρίβεια. Αρχής γενομένης από την Εικόνα 9-3 που αποτυπώνει τον τελικό χάρτη της περιοχής. Παρατηρείται ότι τα περισσότερα εμπόδια είναι κυκλικής διατομής, εντούτοις σε δισδιάστατο χάρτη δεν



Εικόνα 9-4: Χάρτης εξωτερικού χώρου εστιατορίου, με σεσημασμένα τα εμπόδια.

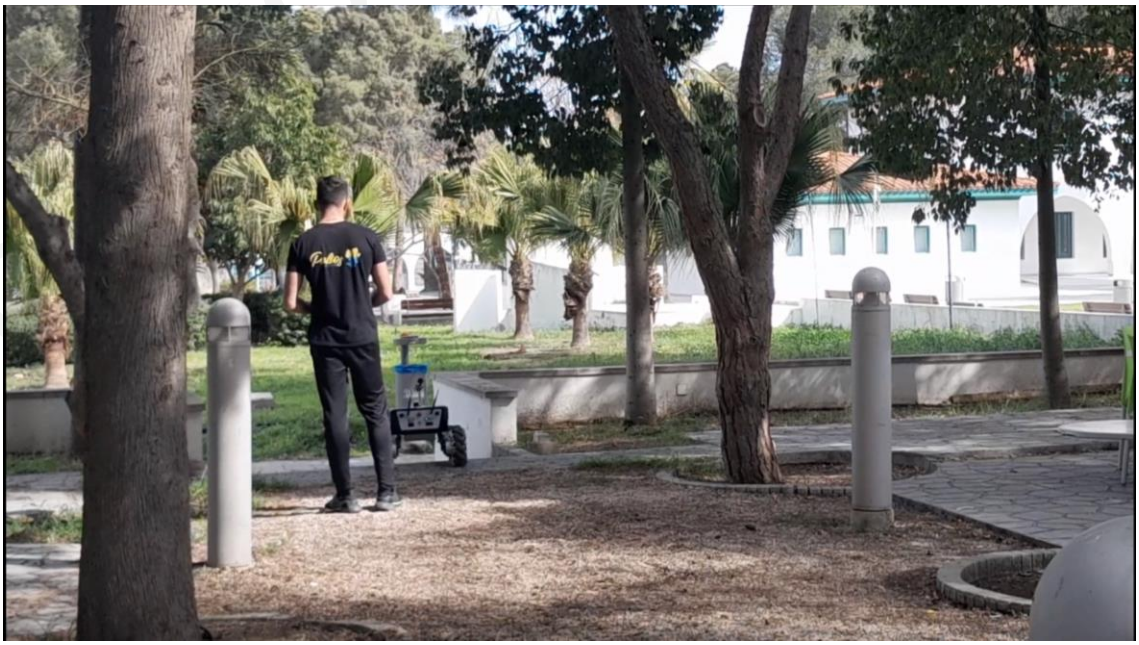
διακρίνεται η φύση του κάθε εμποδίου. Είναι όμως σημαντικό ο χειριστής να γνωρίζει σε τι αντιστοιχεί το κάθε τι εφόσον το κάθε εμπόδιο χρίζει διαφορετικής προσέγγισης.

Για τον λόγο αυτό στην Εικόνα 9-4 παρουσιάζεται ο χάρτης με σεσημασμένα τα εμπόδια είτε αυτά ανήκουν στην κατηγορία των σταθερών, είτε κινητών εμποδίων. Αρχίζοντας από την πρώτη κατηγορία, σε αυτήν εμπίπτουν τα δέντρα, από τα οποία απεικονίζεται η κυκλική διατομή των κορμών τους. Στην Εικόνα 9-4 αυτά μαρκάρονται με μοβ χρώμα και είναι κατά κύριο λόγο ομοιόμορφα κατανεμημένα στον χώρο. Αυτό διακρίνεται τόσο στο χάρτη, όσο και στην Εικόνα 9-2, με τις λεκάνες τους να καθορίζουν την απόσταση ασφαλείας που μπορεί το ρομπότ να πλησιάσει.

Μία άλλη ιδιαιτερότητα που αποτυπώνεται και μαρκάρεται με πράσινο χρώμα στην Εικόνα 9-4 είναι οι ράμπες οι οποίες αποτελούν τις δύο μοναδικές εισόδους στον χώρο πέραν της θύρας που οδηγεί στο εστιατόριο. Οι ράμπες ως γνωστό επιτρέπουν την διέλευση μεταξύ δύο περιοχών με διαφορετικό υψόμετρο, έτσι και η ακτινική σάρωση που αποτυπώνεται με λευκό χρώμα στην προέκταση των δύο ανοιγμάτων. Στην Εικόνα 9-5 απεικονίζεται η ράμπα νοτιοανατολικά του χώρου ενδιαφέροντος, ενώ στην Εικόνα 9-6 ο τρόπος προσέγγισης της περιοχής ώστε να προκύψουν τα καλύτερα δυνατά

Εικόνα 9-5: Άνοιγμα στα νοτιοανατολικά του χώρου ενδιαφέροντος.

αποτελέσματα στη χαρτογράφηση. Η διαδικασία επαναλαμβάνεται με παρόμοιο τρόπο και για το δεύτερο άνοιγμα.



Εικόνα 9-6: Μέθοδος προσέγγισης ανοίγματος.

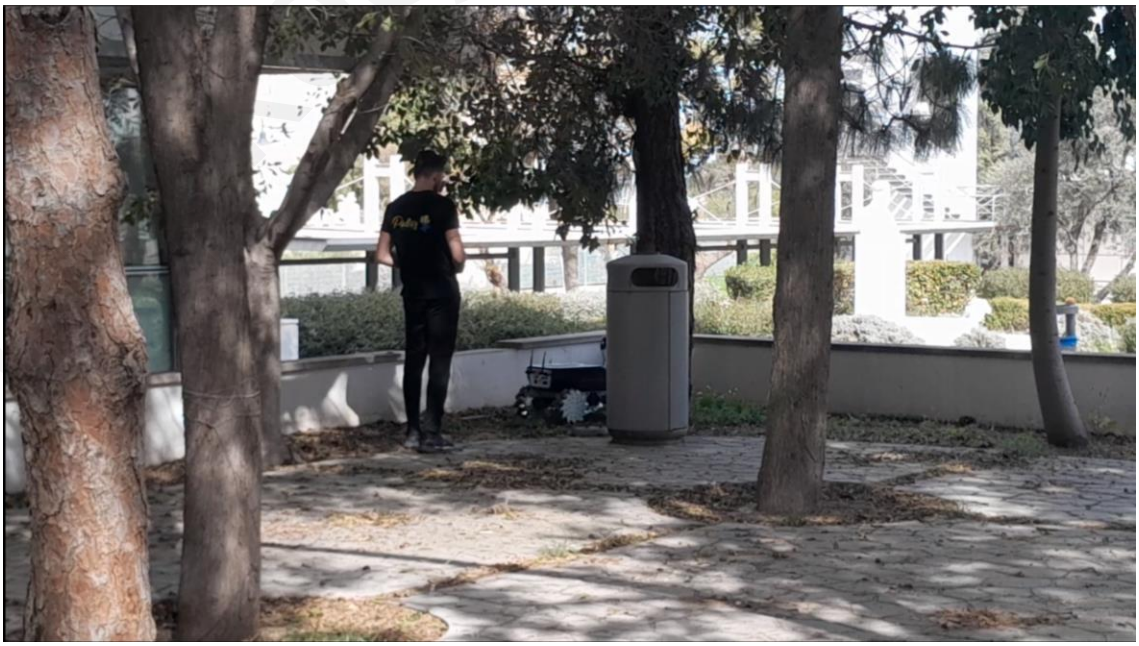
Πέρα από τα ανοίγματα και τα δέντρα, εντός της περιοχής υπάρχουν και περιτοιχίσματα λεκανών και θάμνων τα οποία μαρκάρονται με μπλε πλαίσιο. Το ύψος τους δεν ξεπερνά το μισό μέτρο και υπάρχουν στον χώρο με κάπως άτακτα κατανεμημένο τρόπο. Ακολούθως, η Εικόνα 9-7 παρουσιάζει το μακρόστενο περιτοίχισμα, τμήμα του οποίου μαρκάρεται με το μεγάλο πλαίσιο.

Εικόνα 9-7: Μακρόστενο οβάλ περιτοίχισμα.

Η Εικόνα 9-8 απεικονίζει το κυκλικό περιτοίχισμα που εξυπηρετεί λεκάνη θάμνου και μαρκάρεται με το μικρό μπλε πλαίσιο στον χάρτη. Σημειώνεται ότι στην ίδια Εικόνα διακρίνεται στο βάθος και το δεύτερο άνοιγμα βορειοανατολικά της περιοχής.

Στη συνέχεια η Εικόνα 9-9 απεικονίζει τον κάδο απορριμμάτων στην αριστερή γωνία του χάρτη. Οι κάδοι μαρκάρονται στην Εικόνα 9-4 με κόκκινο πλαίσιο. Στην Εικόνα 9-9 παρατηρείται και η λεπτομερής περιήγηση στην περιοχή πέριξ του κάδου προκειμένου να ανακτηθεί χάρτης χωρίς ατέλειες στην εν λόγω περιοχή.

Εικόνα 9-8: Κυκλικό περιτοίχισμα που εξυπηρετεί λεκάνη θάμνου.



Εικόνα 9-9: Προσέγγιση κάδου απορριμμάτων για χαρτογράφηση περιμετρικά.

Εικόνα 9-10: Υπόδειξη φωτιστικών που μαρκάρονται στον χάρτη.

Η αναφορά στα σταθερά εμπόδια ολοκληρώνεται με το μαρκάρισμα των φωτιστικών στον χάρτη με κίτρινα πλαίσια. Τα συγκεκριμένα δύο φωτιστικά, συμμετρικά των κορμών, παρουσιάζονται στην Εικόνα 9-10.

Αναφορικά με τα κινητά εμπόδια, αυτά τοποθετήθηκαν κυρίως για σκοπούς αναγνώρισης από τον χειριστή, σε περίπτωση που βρεθεί αντιμέτωπος με αυτά κατά την πλοήγηση. Μαρκάρονται στον χάρτη με γκρι πλαίσια, ενώ παρατηρείται ότι σε σύγκριση με τα υπόλοιπα εμπόδια, παρουσιάζουν αρκετά χαμηλότερη ευκρίνεια. Το μικρό πλαίσιο μαρκάρει τον εξοπλισμό που απεικονίζει η Εικόνα 9-11, ενώ το μεγάλο πλαίσιο περιέχει τον σταθμό ελέγχου του ρομποτικού συστήματος (βλ. Εικόνα 9-12).

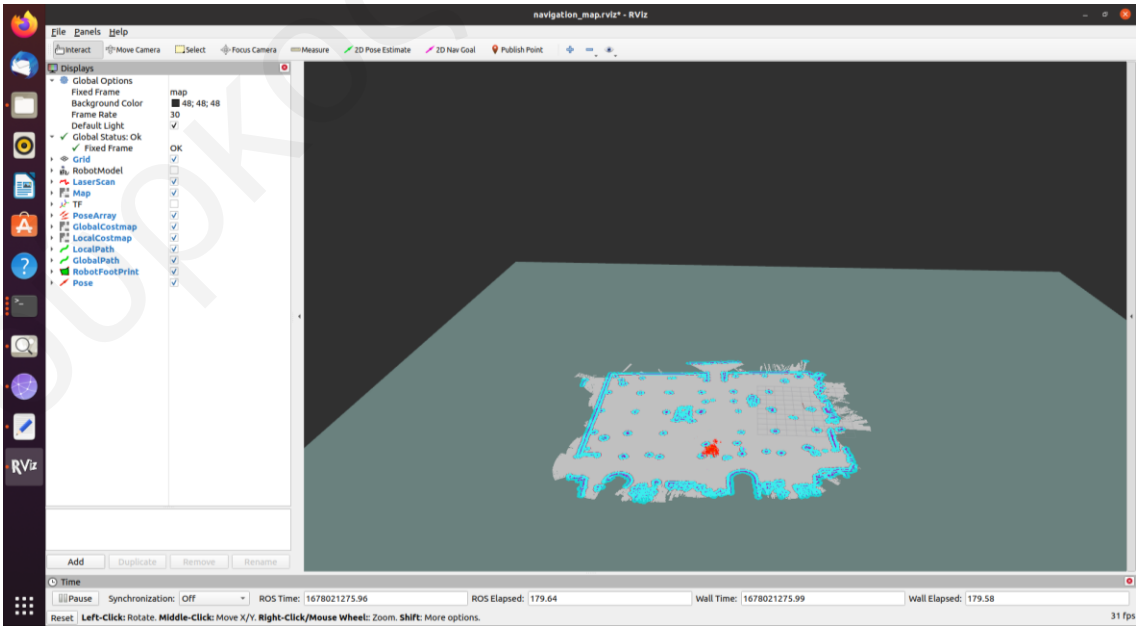
Εικόνα 9-11: Μετακινούμενος εξοπλισμός.

Εικόνα 9-12: Σταθμός ελέγχου.

9.5 Localization

Όταν ανακτηθεί ο χάρτης και ολοκληρωθεί η διαδικασία, χάνεται κατά κάποιο τρόπο η θέση και ο προσανατολισμός του ρομπότ στον χώρο. Με το που ξεκινά η διαδικασία για τη δημιουργία πορείας πλοήγησης τρέχοντας τον κώδικα start_navigation_with_map.launch (βλ. Παράρτημα Ι), το ρομπότ θεωρεί ότι βρίσκεται στην τελευταία θέση όπου αφέθηκε κατά τη διάρκεια της χαρτογράφησης, άσχετα με το αν έχει μετατοπιστεί στον χώρο. Προκειμένου να γίνει σωστός σχεδιασμός πορείας, το ρομπότ χρειάζεται να τοποθετηθεί στον χάρτη στην εκάστοτε φυσική του θέση. Σε αντίθετη περίπτωση, εάν δημιουργηθεί το μονοπάτι με το ρομπότ να βρίσκεται σε διαφορετική θέση με αυτή που πραγματικά υφίσταται, τότε θα ξεκινήσει τη διαδικασία από εκεί και θα ακολουθεί την πορεία που σχεδιάστηκε με σταθερή μετατόπιση. Αυτό εμπεριέχει τον κίνδυνο το ρομπότ να βρεθεί αντιμέτωπο με εμπόδια τα οποία δεν αναμένει με βάση τον χάρτη.

Προς αποφυγή του πιο πάνω προβλήματος γίνεται launch ο amcl αλγόριθμος, είτε απευθείας ή μέσω άλλου launch file, ο οποίος είναι υπεύθυνος για το localization του ρομπότ. Όπως φαίνεται και στην Εικόνα 9-13, το ρομπότ τοποθετείται στον χάρτη στο σημείο που βρίσκεται στην πραγματικότητα. Η θέση και ο προσανατολισμός του ρομπότ μπορεί επίσης να γίνει γνωστή στο σύστημα μέσω της επιλογής 2D Pose Estimate.

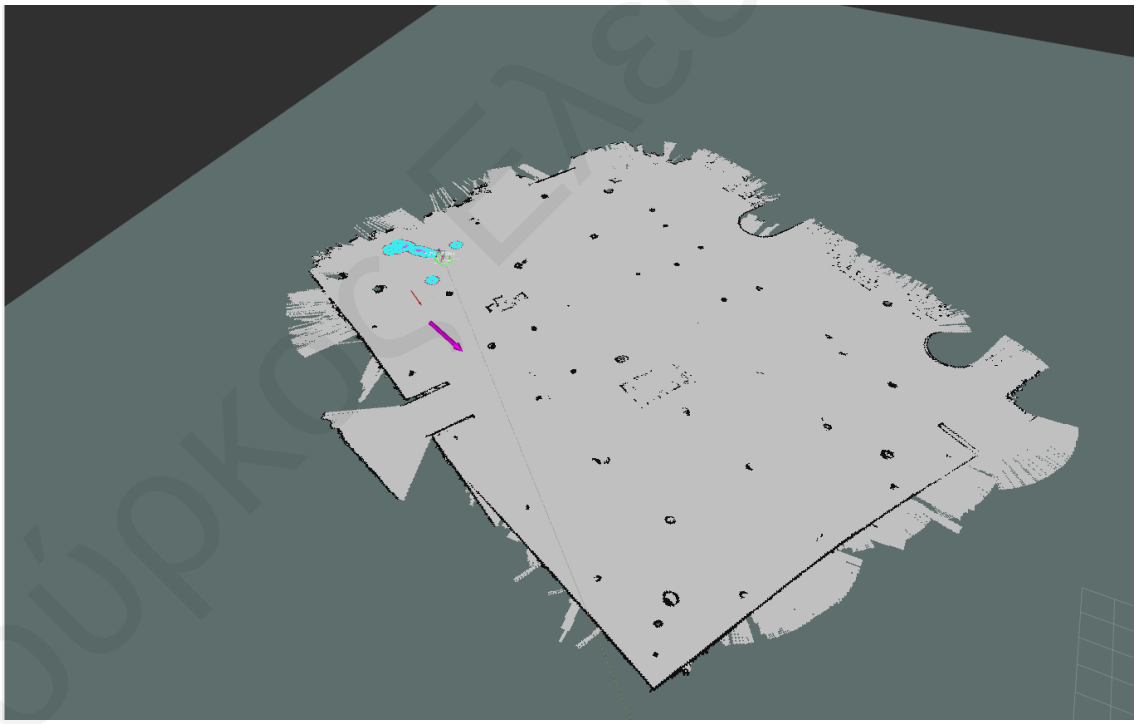


Εικόνα 9-13: Εισαγωγή χάρτη στο Gazebo και προσδιορισμός της θέσης και του προσανατολισμού του ρομπότ.

9.6 Σχεδιασμός πορείας πλοήγησης

Εφόσον ανακτηθεί ο χάρτης και επιβεβαιωθεί η ποιότητά του καταχωρείται στο λογισμικό προσομοίωσης Gazebo προκειμένου να αξιοποιηθεί για σκοπούς πλοήγησης. Κάνοντας launch το πακέτο follow waypoints (βλ. Παράρτημα Ι) παρέχεται η δυνατότητα καταχώρησης σημείων πορείας πλοήγησης. Μέσω της επιλογής 2D Nav Goal καταχωρούνται διαδοχικά τα σημεία με σκοπό τη δημιουργία της επιθυμητής πορείας του ρομπότ. Μόλις ο χειριστής ολοκληρώσει την πορεία, αποθηκεύει τα σημεία και τα προωθεί στο topic /start_journey, προκειμένου να προωθηθούν στον κόμβο move_base.

Κατά τη διάρκεια καταχώρησης των σημείων, αυτά θα αποτυπώνονται με βέλη στον χάρτη με τον τρόπο που φαίνεται στην Εικόνα 9-14. Η αρχή του βέλους καθορίζει την επιθυμητή θέση του ρομπότ, ενώ η φορά του βέλους προφανώς αποτυπώνει την επιθυμητή κατεύθυνση του ρομπότ στο συγκεκριμένο σημείο.



Εικόνα 9-14: Καταχώρηση σημείων πορείας.

Σημειώνεται ότι οι γαλάζιες φωτεινές ενδείξεις αντιστοιχούν με την ανίχνευση των αισθητήρων σε πραγματικό χρόνο. Τα εμπόδια που εντοπίζει το ρομπότ στον περίγυρό του τη συγκεκριμένη στιγμή. Καθώς αυτό κατευθύνεται από ένα σημείο στο επόμενο οι ενδείξεις διαφοροποιούνται αναλόγως, ωστόσο παραμένουν περιμετρικά του ρομπότ, σε σημεία δηλαδή που ανιχνεύονται από τον αισθητήρα laser.

9.7 Πλοήγηση

Αφότου ολοκληρωθούν όλες οι πιο πάνω διαδικασίες το ρομπότ βρίσκεται σε θέση να διεισδύσει στο χώρο ενδιαφέροντος στο προκαθορισμένο μονοπάτι. Αρκεί μόνο να γίνει launch το αρχείο start_follow_waypoints.launch. Υπάρχουν περιπτώσεις όμως που ενδέχεται να προκύψει απόκλιση από σημείο σε σημείο ή να διαπιστωθεί σφάλμα στον χάρτη, εν πάση περιπτώσει να χρειάζεται τροποποίηση η πορεία. Αυτό είτε μπορεί να γίνει με επανασχεδιασμό ολόκληρης της πορείας, είτε με τροποποίηση μεμονωμένων σημείων στην λίστα που έχει δημιουργηθεί με τα καταχωρημένα σημεία.

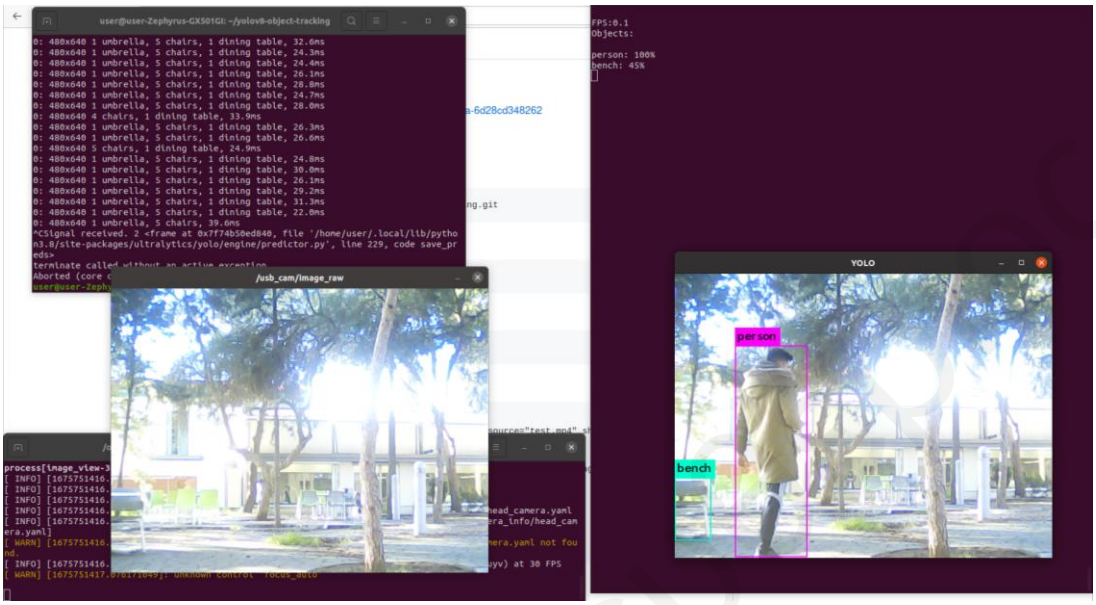
Αυτό όμως επιφυλάσσει την ανάγκη επίγνωσης της ακριβούς τοποθεσίας βάσει συντεταγμένων του νέου σημείου. Το να υπολογιστεί στο περίπου με το μάτι είναι μία λύση, ωστόσο όχι η καλύτερη δυνατή. Αντ' αυτού ο χρήστης μπορεί μέσω της εντολής Publish Point να προωθήσει τις συντεταγμένες οποιουδήποτε σημείου στον συγκεκριμένο κόμβο και ακολούθως μέσω του τερματικού να καλέσει τον κόμβο προκειμένου να αποκτήσει πρόσβαση για να δει τις συντεταγμένες.

Κατά την διάρκεια της πλοήγησης, οπτικό υλικό από την εκάστοτε θέση του ρομπότ προωθείται στον αλγόριθμο YOLO με σκοπό την ανίχνευση σε πραγματικό χρόνο.

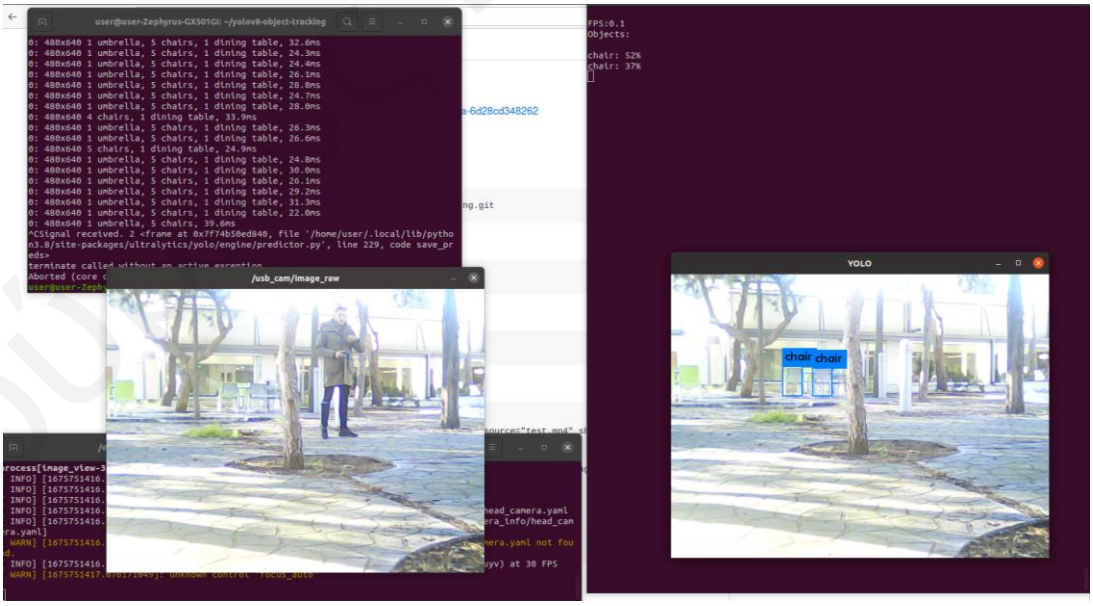
9.8 Αποτελέσματα

Για την ολοκλήρωση της διαδικασίας χρησιμοποιήθηκαν δύο εκδόσεις YOLO. Το YOLOv7 και το YOLOv8 με την διαφορά ότι το πρώτο εκτελούσε την ανίχνευση μέσω ROS Noetic, ενώ το δεύτερο όχι. Για την επίτευξη της σύνδεσης μεταξύ YOLOv7 και ROS έτρεξε σε πρώτη φάση ο κώδικας yolov7_webcam.launch για την εκκίνηση της διαδικασίας λήψης και στη συνέχεια το αρχείο yolov7.launch προκειμένου να ξεκινήσει η ανίχνευση σε πραγματικό χρόνο. Τα δύο αρχεία παρατίθενται στο Παράρτημα ΙΙ. Στη συνέχεια παρουσιάζονται ενδεικτικά αποτελέσματα των δύο περιπτώσεων που αναδεικνύουν τις δυνατότητες, χαρακτηριστικά και αδυναμίες των μεθόδων. Έπειτα ακολουθεί σχολιασμός και σύγκριση. Αξίζει να σημειωθεί ότι για σκοπούς σύγκρισης ολοκληρώθηκε και το μοντέλο YOLOv3 μέσω ROS Noetic, ωστόσο δόθηκε μεγαλύτερη έμφαση στις τελευταίες εκδόσεις του YOLO.

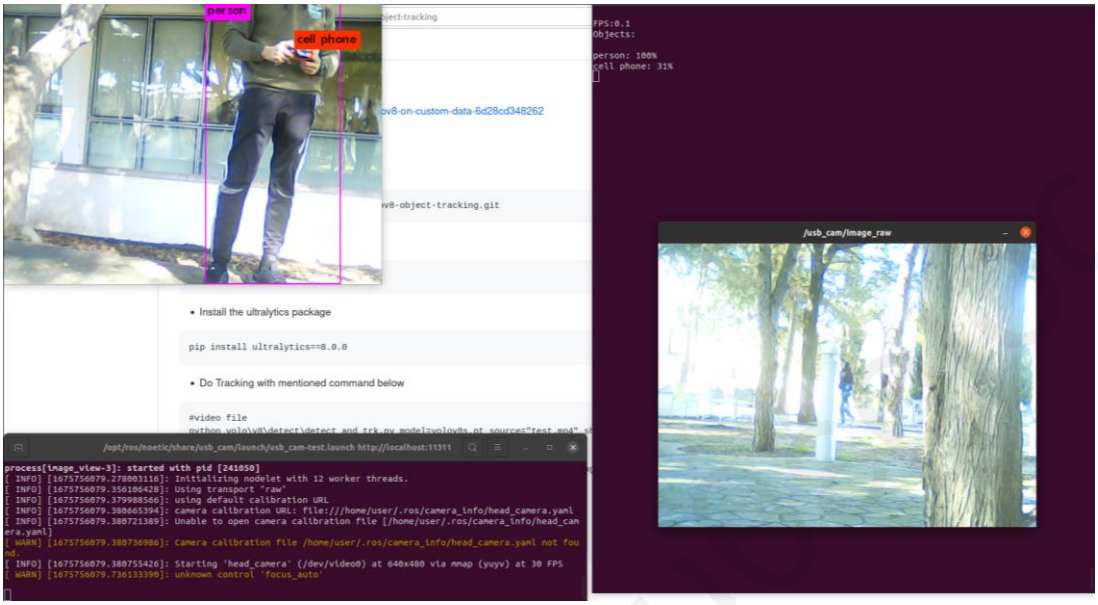
9.8.1Αποτελέσματα YOLOv3 – ROS Noetic



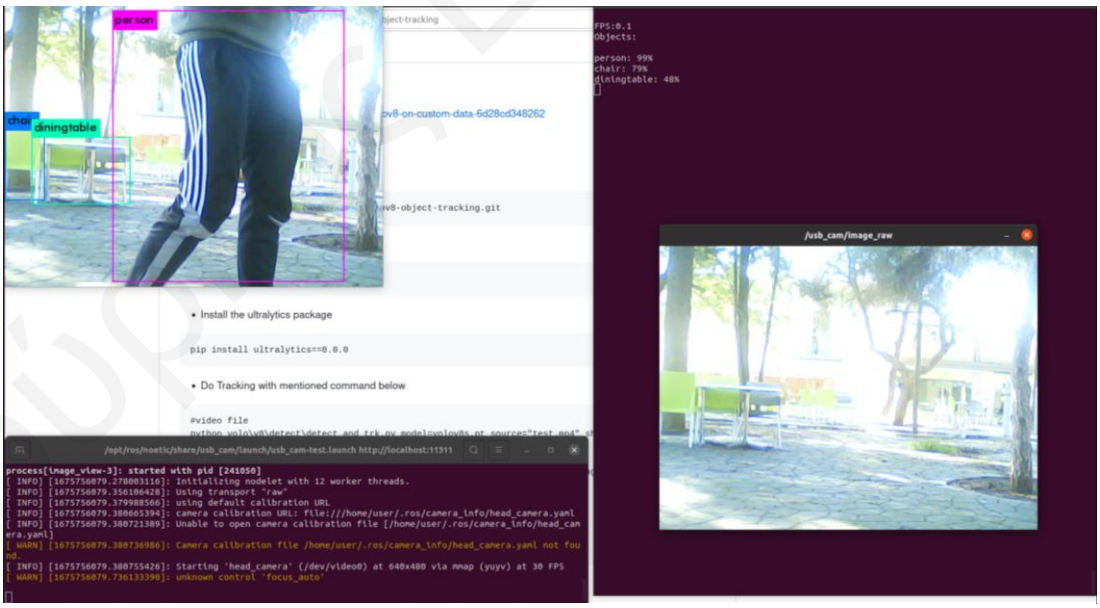
Εικόνα 9-15: Στιγμιότυπο οθόνης με καθυστέρηση και λανθασμένη ετικέτα.



Εικόνα 9-16: Στιγμιότυπο οθόνης με χρονική καθυστέρηση στην επεξεργασία.

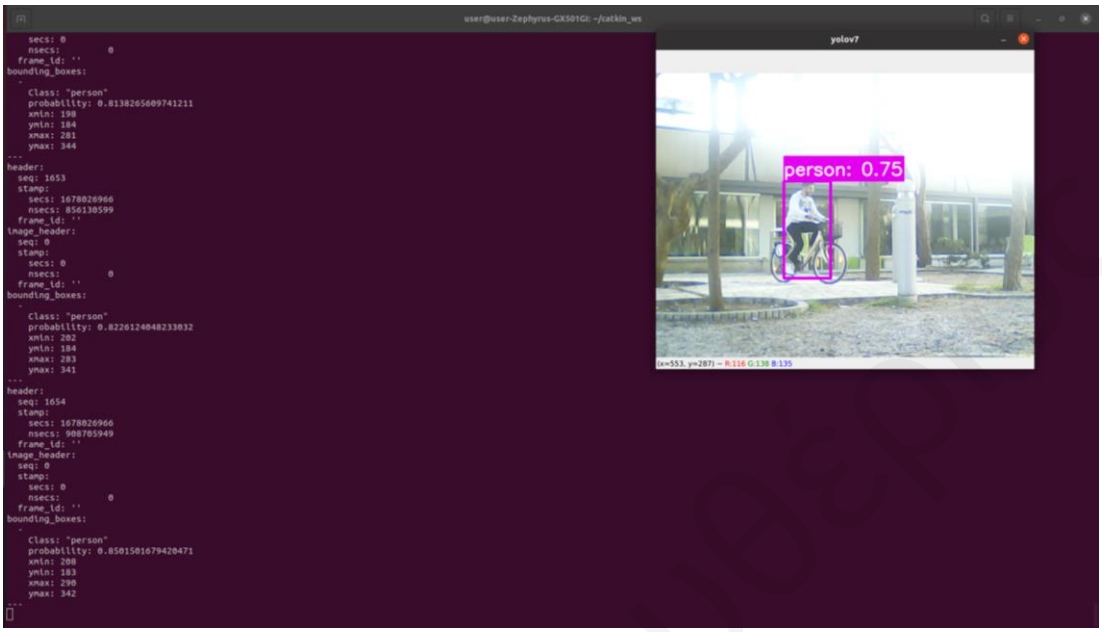


Εικόνα 9-17: Στιγμιότυπο οθόνης με χρονική καθυστέρηση και λανθασμένη ετικέτα.

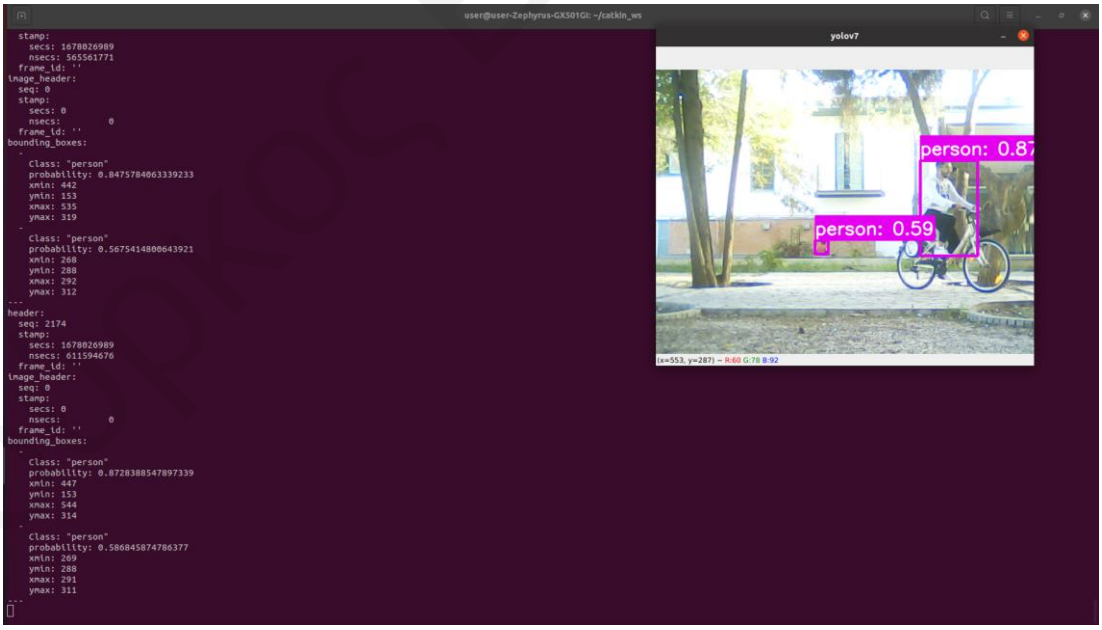


Εικόνα 9-18: Στιγμιότυπο οθόνης με καθυστέρηση αλλά σωστές ετικέτες.

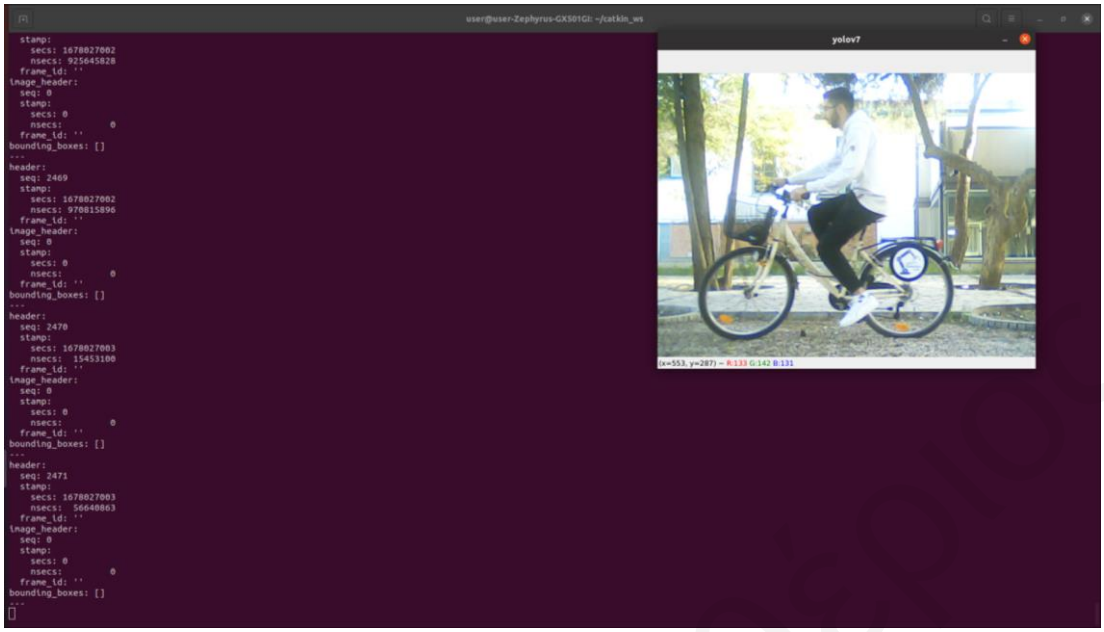
9.8.2Αποτελέσματα YOLOv7 – ROS Noetic



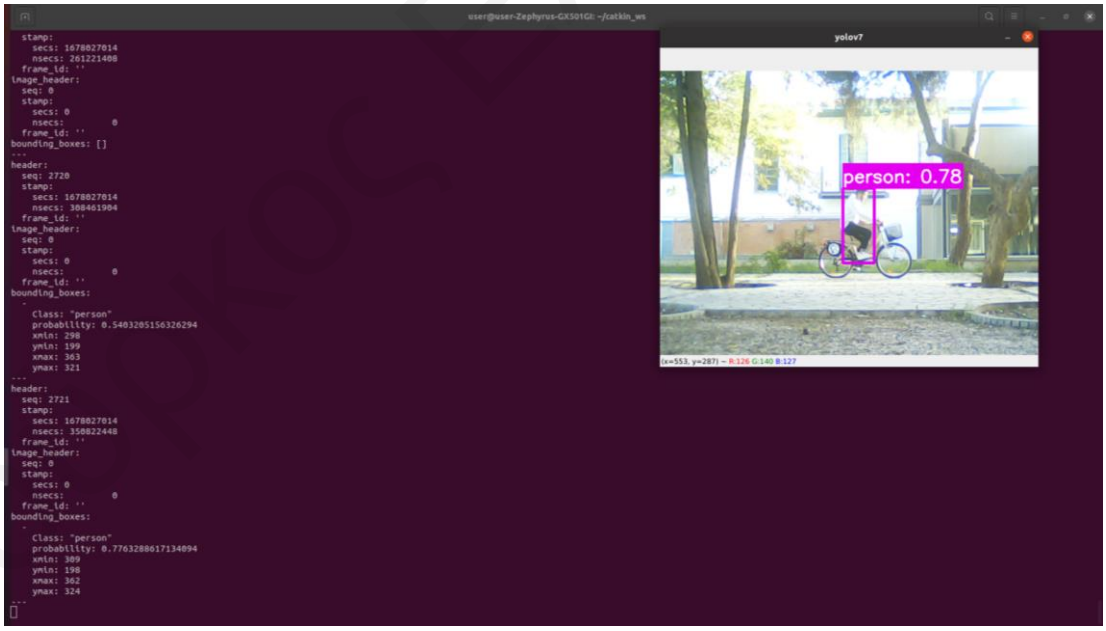
Εικόνα 9-19: Στιγμιότυπο οθόνης με καλή ακρίβεια.



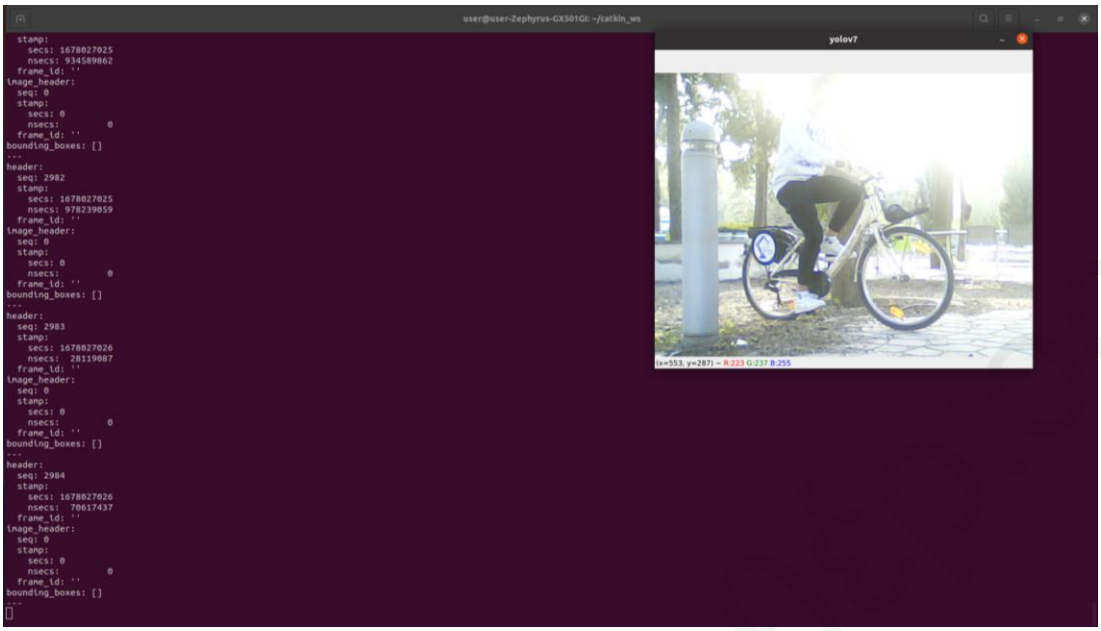
Εικόνα 9-20: Στιγμιότυπο οθόνης με αναγνώριση σε μακρινή απόσταση.



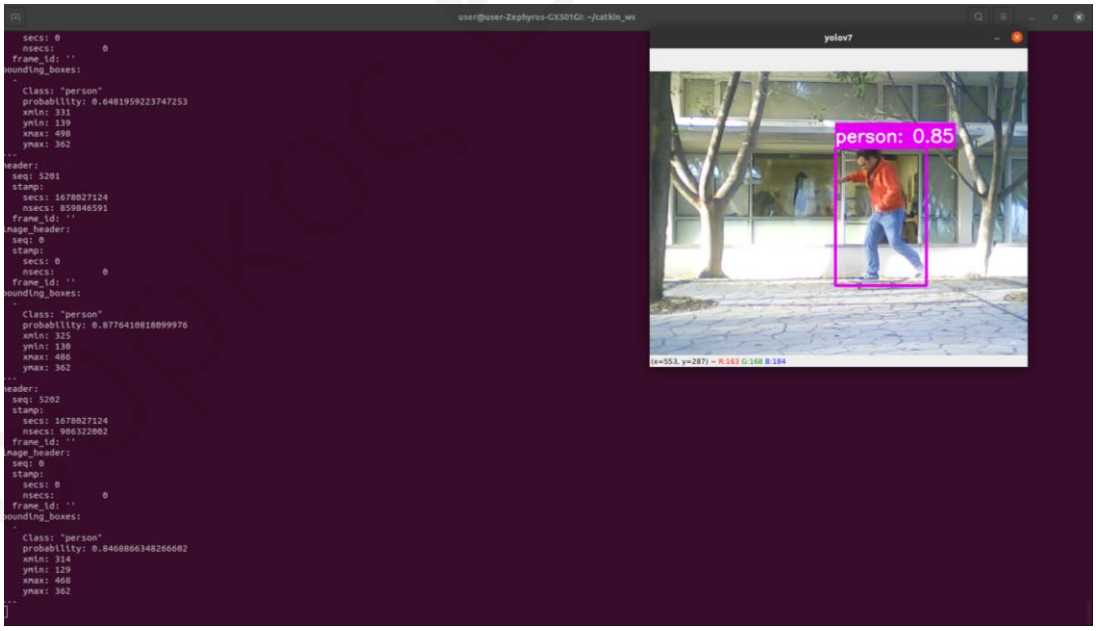
Εικόνα 9-21: Στιγμιότυπο οθόνης με αρνητικό bounding box.



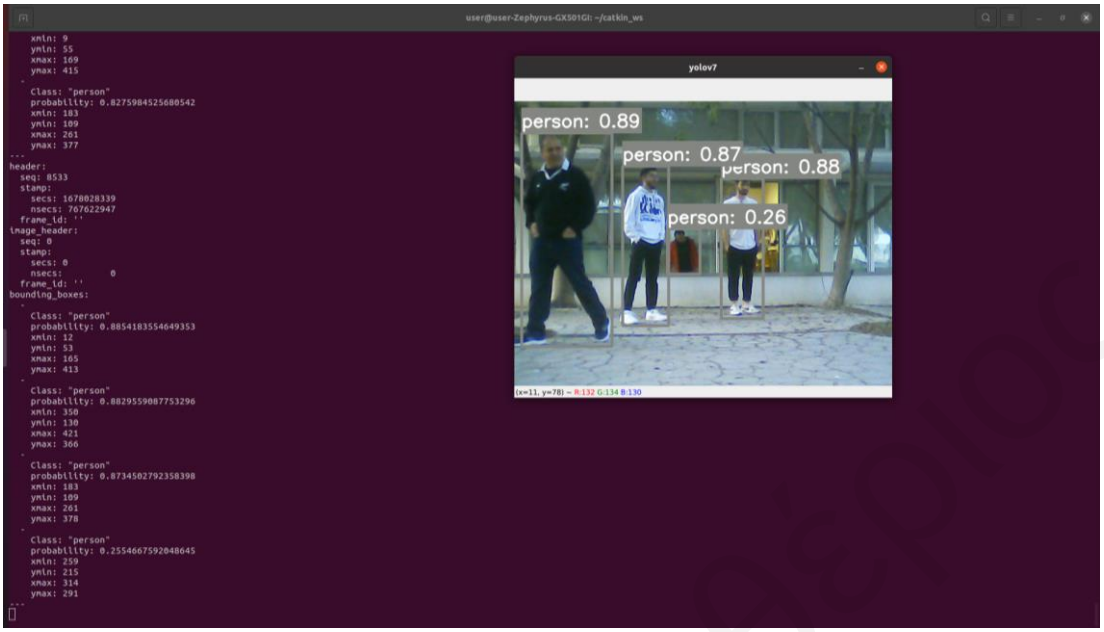
Εικόνα 9-22: Στιγμιότυπο οθόνης με καλή ακρίβεια.



Εικόνα 9-23: Στιγμιότυπο οθόνης με αρνητικό bounding box λόγω ακτινοβολίας.



Εικόνα 9-24: Στιγμιότυπο οθόνης σε γρήγορη κίνηση.



Εικόνα 9-25: Στιγμιότυπο οθόνης με ανθρώπους σε διάφορες αποστάσεις.

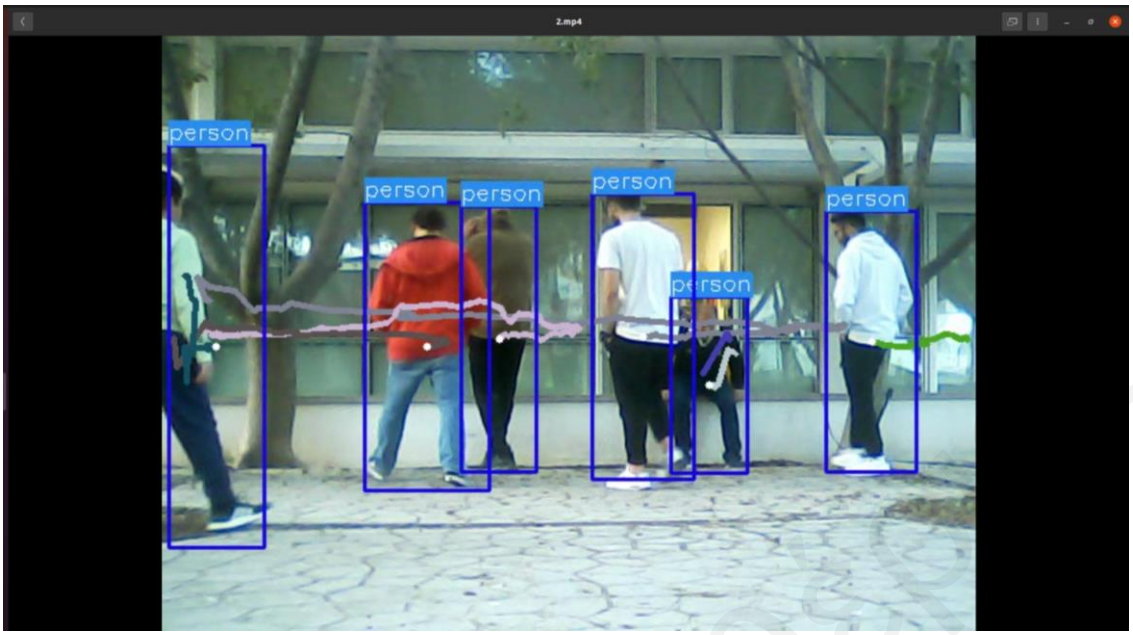
9.8.3Αποτελέσματα YOLOv8

Εικόνα 9-26: Στιγμιότυπο οθόνης με διαφορετικές κλάσεις.

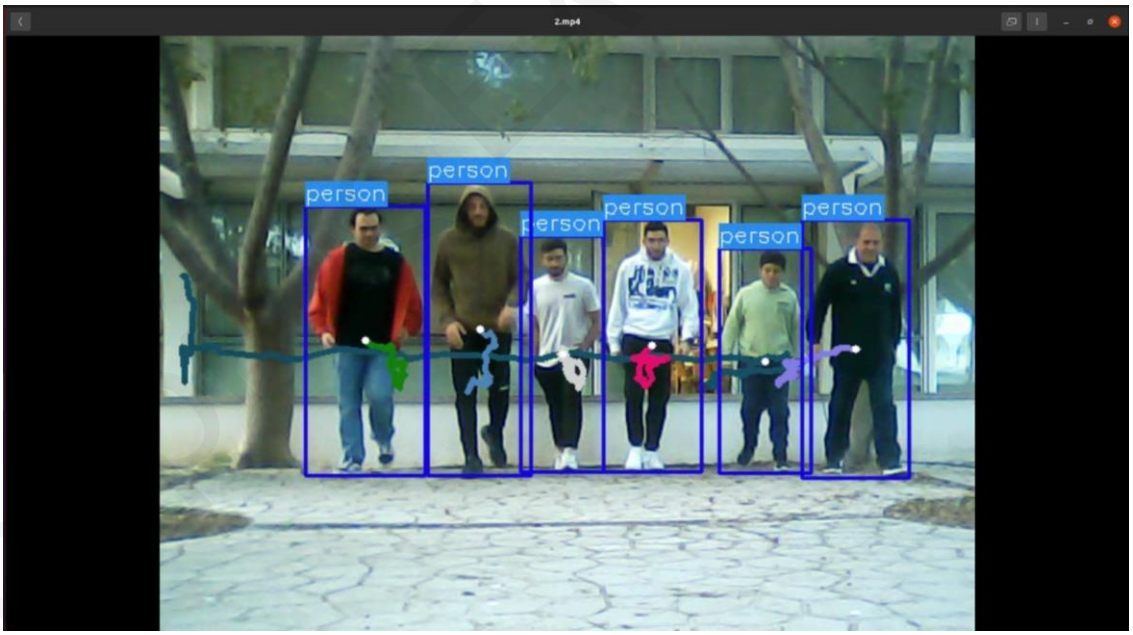
Εικόνα 9-27: Στιγμιότυπο οθόνης με πολλές κλάσεις.

Εικόνα 9-28: Στιγμιότυπο οθόνης με μερική απόκρυψη αντικειμένων σε γρήγορη ωστόσο κίνηση.

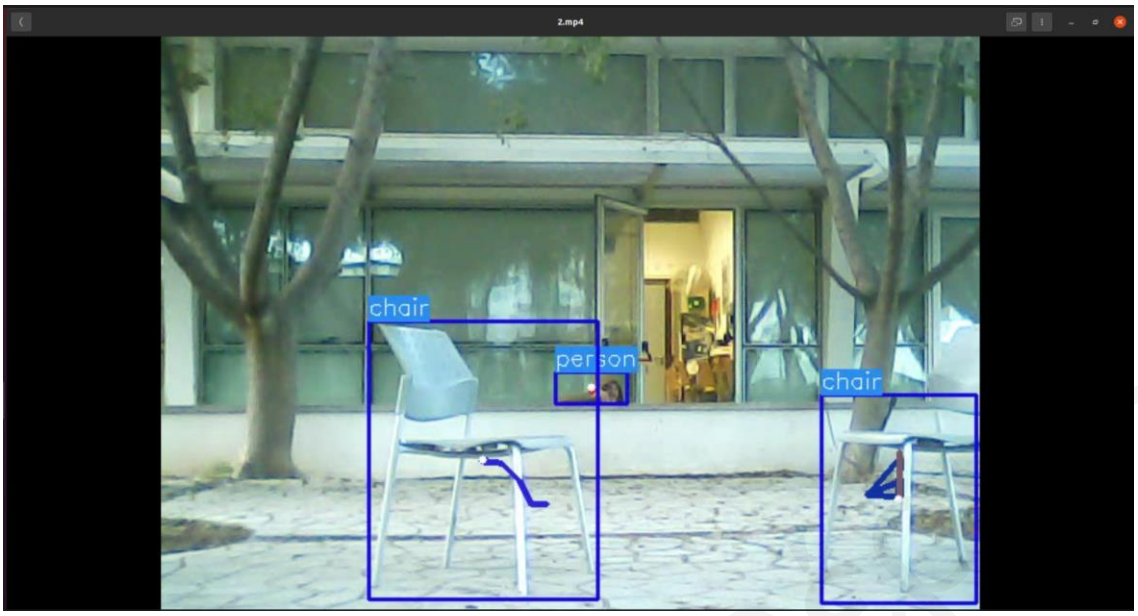
Εικόνα 9-29: Στιγμιότυπο οθόνης σε γρήγορη κίνηση.



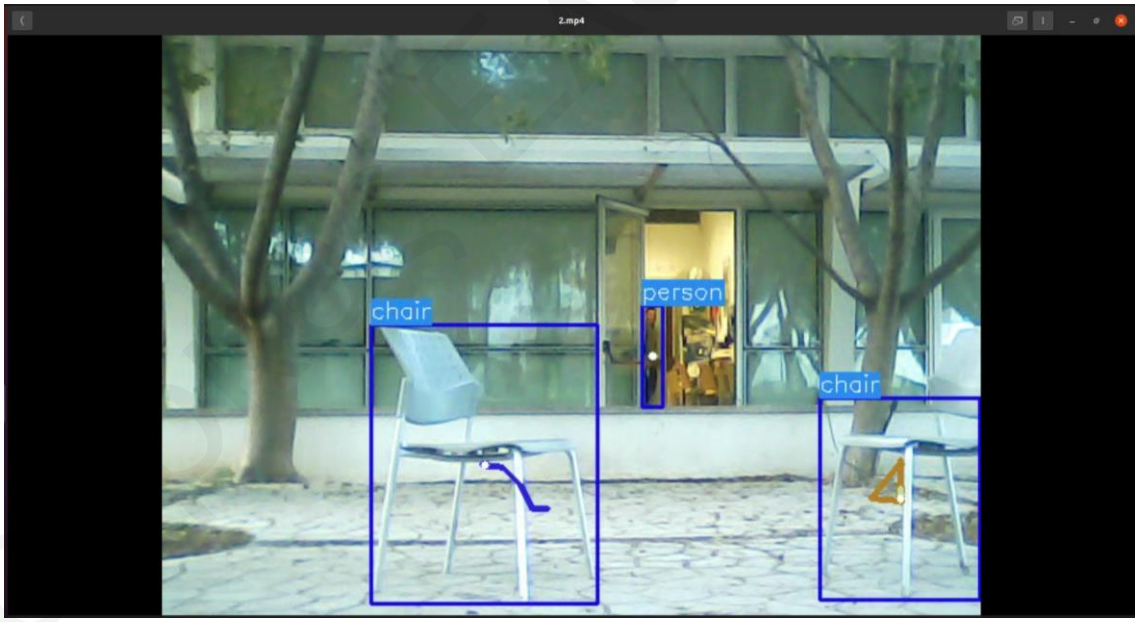
Εικόνα 9-30: Στιγμιότυπο οθόνης με διαφορετικά σχήματα αντικειμένων ίδιας κλάσης.



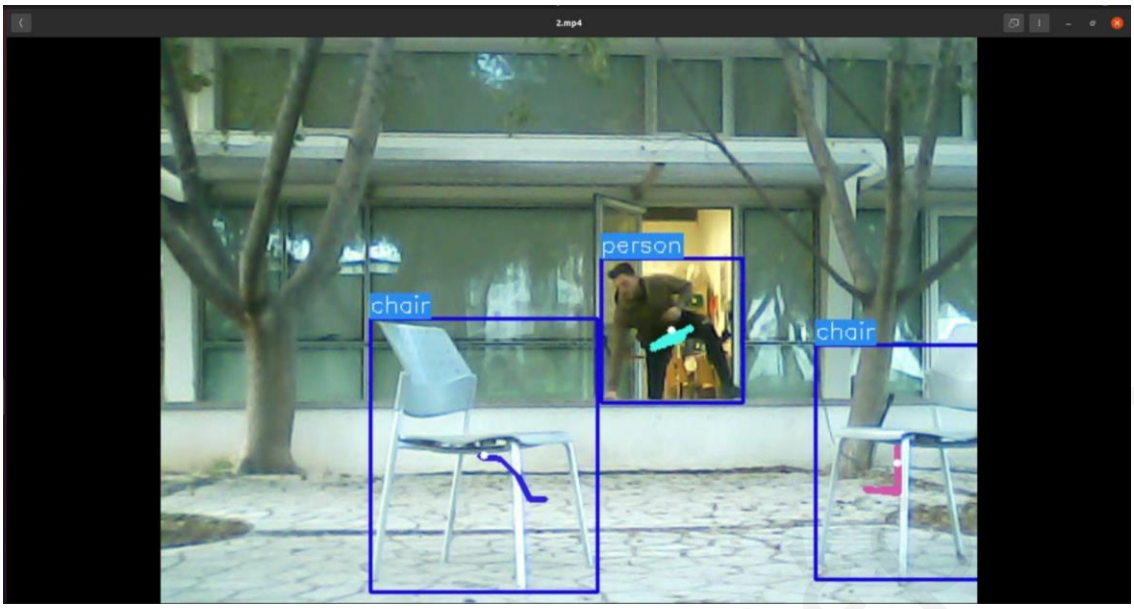
Εικόνα 9-31: Στιγμιότυπο οθόνης με ομοιόμορφα αντικείμενα ίδιας κλάσης.



Εικόνα 9-32: Στιγμιότυπο οθόνης με κρυμμένο άνθρωπο (πίσω από φράκτη).



Εικόνα 9-33: Στιγμιότυπο οθόνης με άνθρωπο σε μεγάλη απόσταση (εντός του κτηρίου).



Εικόνα 9-34: Στιγμιότυπο οθόνης σε γρήγορη κίνηση με διαφορετικό σχήμα αντικειμένου.

9.8.4Σχολιασμός αποτελεσμάτων

Στα αποτελέσματα που παρατέθηκαν, πάρθηκαν στιγμιότυπα οθόνης από διαφορετικές περιπτώσεις κατά τη διάρκεια ανίχνευσης σε πραγματικό χρόνο.

Αυτό που συμπεραίνεται από τα αποτελέσματα του YOLOv3 είναι ότι υπάρχει σημαντική χρονική καθυστέρηση της τάξης των 6 δευτερολέπτων. Στα στιγμιότυπα που παρατέθηκαν υπάρχουν δύο ανοιχτά παράθυρα, στην οθόνη, από λήψεις. Το ένα παράθυρο παρουσιάζει την εικόνα που λαμβάνεται από την κάμερα σε πραγματικό χρόνο. Το δεύτερο παράθυρο είναι η ταινία που προκύπτει έπειτα από την ανίχνευση. Παρόλο που η αναφορά για ανίχνευση αντικειμένου γίνεται σχεδόν αμέσως στο τερματικό, το βίντεο με τα bounding boxes καθυστερεί να προβληθεί. Το πόσο σημαντικό ζήτημα είναι αυτό εξαρτάται πάντα από την εφαρμογή.

Παράλληλα, το συγκεκριμένο μοντέλο φαίνεται να προσδίδει λανθασμένη ετικέτα σε μία περίπτωση. Η ένδειξη δεν πρόκειται για την κατηγορία του ανθρώπου που ενδιαφέρει άμεσα την εφαρμογή, ωστόσο αποτελεί ζήτημα για την αξιοπιστία του μοντέλου.

Όσον αφορά τα αποτελέσματα του YOLOv7, αυτά παρουσιάζουν σαφώς μικρότερη καθυστέρηση της τάξης των 1 – 2 δευτερολέπτων. Το συγκεκριμένο χρονικό διάστημα δεν αποτελεί τόσο μεγάλο ζήτημα, εντούτοις είναι προτιμότερο να μην υπάρχει καθόλου. Ένα άλλο ζήτημα που φαίνεται να παρουσιάζει το συγκεκριμένο

μοντέλο είναι τα αρνητικά bounding boxes. Σε δύο περιπτώσεις, στις Εικόνες 9-21 και 9-23 δεν ανιχνεύθηκε ο άνθρωπος. Στη δεύτερη περίπτωση σαφώς επηρεάζει η φωτεινότητα στην εικόνα και στις δύο όμως το γεγονός ότι δεν ανιχνεύτηκε άνθρωπος ούτε καν με μικρή ακρίβεια, προβληματίζει.

Στον αντίποδα τα αποτελέσματα του YOLOv8 είναι σαφώς πιο αισιόδοξα. Με χρονική καθυστέρηση μικρότερη του ενός δευτερολέπτου, δεν φαίνεται να έχει κάνει καμία λάθος ετικέτα, την ίδια στιγμή που δεν παρουσιάζει κανένα αρνητικό bounding box. Ακόμη και σε δυσμενής συνθήκες ανίχνευσης όπως η μεγάλη απόσταση, η μεγάλη ταχύτητα και η απόκρυψη του αντικειμένου σε μεγάλο βαθμό όπως στην Εικόνα 9-33, η απόδοση του μοντέλου παρουσιάζει εξαιρετικά αποτελέσματα.

Κεφάλαιο 10

10 ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ – ΧΡΗΣΗ ΘΕΡΜΙΚΗΣ ΚΑΜΕΡΑΣ ΕΝΣΩΜΑΤΩΜΕΝΗ ΣΤΟ ΡΟΜΠΟΤ

Το παρόν κεφάλαιο περιγράφει δοκιμασία και αποτελέσματα που αφορούν την χρήση θερμικής κάμερας ενσωματωμένη σε ιπτάμενο ρομπότ και αντιστοιχεί στην Επιλογή 3 που αναφέρθηκε στο Κεφάλαιο 7.

10.1 Θερμική Κάμερα

Η Θερμική κάμερα που χρησιμοποιήθηκε βρίσκεται εγκατεστημένη στο drone και με το πάτημα ενός κουμπιού τίθεται σε λειτουργία. Με τη λειτουργία της θερμικής κάμερας επεκτείνονται αυτόματα οι δυνατότητες της εφαρμογής, ωστόσο χρειάζεται προσοχή ώστε να γίνεται χρήση εκεί όπου χρειάζεται. Είναι προτιμότερο δηλαδή εάν μία εφαρμογή μπορεί να υλοποιηθεί με μία συμβατική κάμερα, τότε περισσεύει η χρήση θερμικής.

Οι λόγοι προτίμησης κανονικής κάμερας έναντι της θερμικής είναι αρκετοί και εξαπλώνονται σε όλο το φάσμα της εφαρμογής. Για παράδειγμα χρειάζονται νέοι συντελεστές βαρύτητας. Με μία σύντομη ματιά στο διαδίκτυο οι επιλογές για pre-trained weights με σύνολο δεδομένων από θερμική κάμερα είναι εξαιρετικά περιορισμένες. Χρειάζεται λοιπόν να συνταχθεί νέο σύνολο δεδομένων που να αποτελείται αποκλειστικά από εικόνες που έχουν παρθεί από θερμική κάμερα. Αυτό θα έχει ως αποτέλεσμα να αυξηθεί το κόστος εφόσον το drone θα κληθεί να εκτελέσει αλλεπάλληλες πτήσεις προκειμένου να αποκτήσει τις εικόνες του συνόλου δεδομένων, με τρόπο τέτοιο ώστε να υπάρχουν φωτογραφίες, ανθρώπου τουλάχιστο, από διαφορετικές λήψεις με διαφορετικούς σχηματισμούς του αντικειμένου ενδιαφέροντος.

Τη συλλογή εικόνων θα ακολουθήσει η επισήμανση με ετικέτες, διαδικασία που αυξάνει σε πολύ μεγάλο βαθμό τον απαιτούμενο χρόνο για εξαγωγή συντελεστών βαρύτητας. Για ένα σύνολο δεδομένων διαφορετικής φύσης απ’ ότι τα υπόλοιπα. Η λεπτομερής ρύθμιση των παραμέτρων ενδέχεται να διαρκέσει μεγάλο χρονικό

διάστημα, εάν αναλογιστεί κανείς ότι χρειάζεται καθ' ένα από τα runs να ολοκληρώνεται προκειμένου να προκύπτουν τα όποια αποτελέσματα.

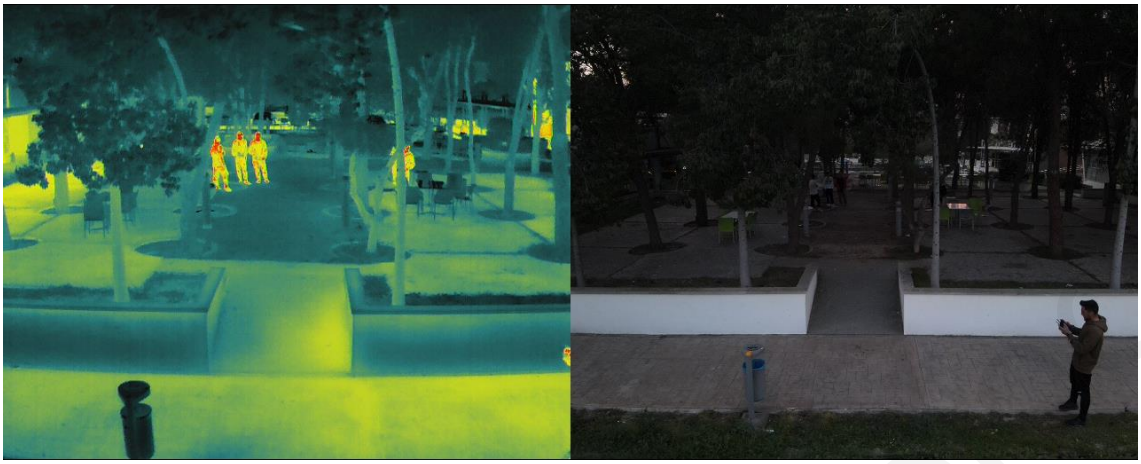
Όσο δύσκολη όμως κι αν είναι η συγκεκριμένη εφαρμογή, υπάρχουν περιπτώσεις που είναι η μοναδική πιθανή λύση. Για να φτάσουν τα πράγματα όμως μέχρι εκεί, το μόνο σίγουρο είναι ότι πρόκειται για αρκετά απαιτητική εφαρμογή, άρα εάν θα γίνει ανίχνευση με τον συγκεκριμένο τρόπο χρειάζεται να γίνει με όσο το δυνατό μεγαλύτερη ακρίβεια. Έτσι και στη συγκεκριμένη εφαρμογή στους χώρους ενδιαφέροντος που περιεγράφηκαν στα προηγούμενα δύο κεφάλαια, απαιτείται η ανίχνευση σε εικόνες από θερμική κάμερα, εφόσον είναι ο μόνος τρόπος να διακριθεί άνθρωπος κατά τις βραδινές ώρες.

10.2 Δημιουργία Συντελεστών Βαρύτητας

Σε πρώτη φάση επιχειρήθηκε η χρήση συντελεστών βαρύτητας από το υφιστάμενο σύνολο δεδομένων. Τα αποτελέσματα όμως ήταν απογοητευτικά με ένα αντιπροσωπευτικό δείγμα να παρουσιάζεται στη συνέχεια. Η δεύτερη σκέψη ήταν να εκπαιδευτεί το μοντέλο σε σύνολα εικόνων που βρίσκονται διαθέσιμα στο διαδίκτυο, κάτι που έγινε με επιτυχία συνδυάζοντας δύο ξεχωριστά σύνολα δεδομένων. Το πρώτο αφορά εικόνες λήψεων drone από μεγάλο ύψος [24], ενώ το δεύτερο από εικόνες λήψεων θερμικής κάμερας [62].

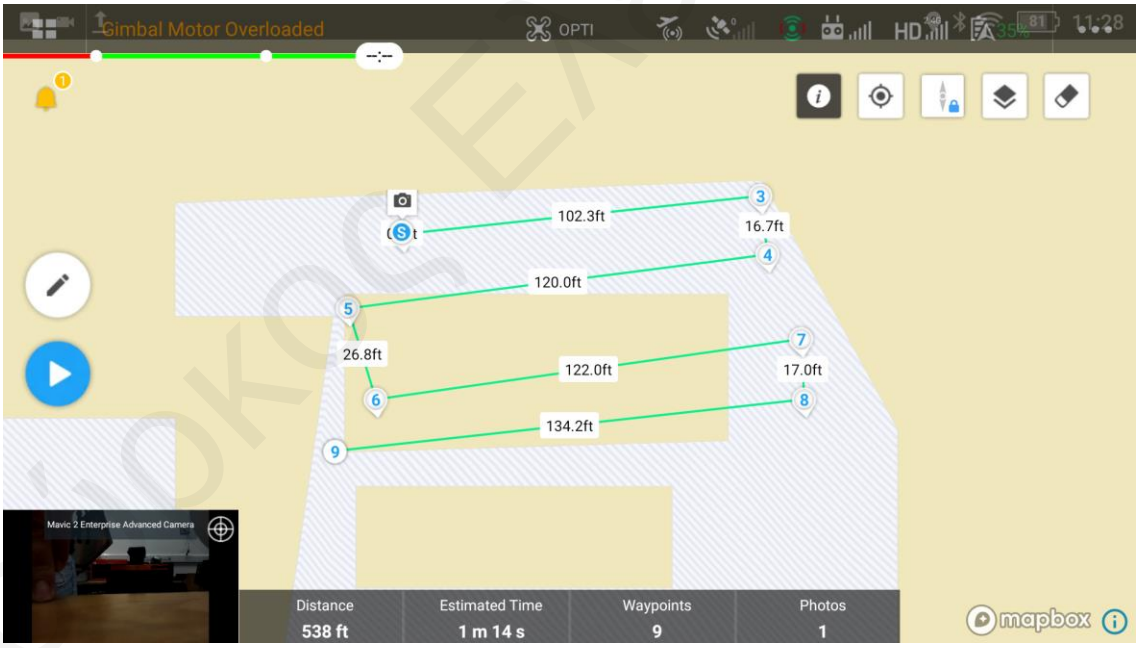
10.3 Σχεδιασμός πορείας πλοήγησης

Το γεγονός ότι η θερμική κάμερα λαμβάνει μικρότερης έκτασης λήψη δεν επιτρέπει την αξιοποίηση της πορείας που διαγράφεται στην Εικόνα 8-4. Για πλήρη κατανόηση του πιο πάνω ζητήματος, είναι βοηθητική ή Εικόνα 10-1 που αντιστοιχεί σε ταυτόχρονη λήψη από τις δύο κάμερες που βρίσκονται ενσωματωμένες στο ρομπότ. Στα αριστερά της εικόνας, είναι η φωτογραφία από τη θερμική κάμερα, ενώ στα δεξιά από την κανονική την ίδια ακριβώς χρονική στιγμή. Όπως φαίνεται, στη φωτογραφία από την κανονική κάμερα καλύπτεται ένα ευρύτερο μέρος της περιοχής, αντίθετα στη φωτογραφία από τη θερμική κάμερα καλύπτεται ένα πιο περιορισμένο μέρος. Το βασικό πρόβλημα που προκύπτει από τις δύο λήψεις είναι ότι στην πρώτη δεν καλύπτεται μέρος της περιοχής περιμετρικά της εικόνας με αποτέλεσμα να μην φαίνεται ο άνθρωπος στα δεξιά.



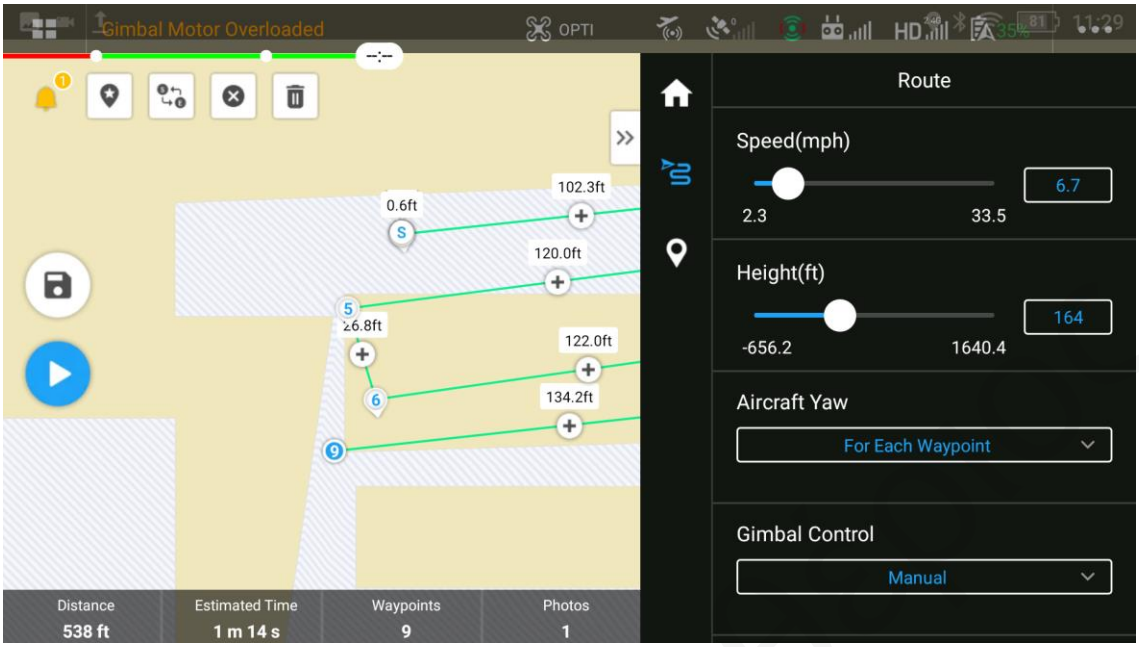
Εικόνα 10-1: Ταυτόχρονη λήψη από συμβατική και θερμική κάμερα.

Προκειμένου να μην υπάρχουν τυφλά σημεία κατά την πλοήγηση, δημιουργήθηκε εκ νέου η πορεία με κοντινότερες αποστάσεις μεταξύ των εγκάρσιων διαδρομών. Η νέα πορεία πλοήγησης παρουσιάζεται στην Εικόνα 10-2.



Εικόνα 10-2: Δημιουργία νέας πορείας πλοήγησης κοντινότερων αποστάσεων.

Ακολούθως παρουσιάζεται στην Εικόνα 10-3 το παράθυρο που επιτρέπει την επεξεργασία των παραμέτρων που αφορούν την πτήση. Λόγω του ότι η θερμική κάμερα αποτυπώνει τις εικόνες με μικρότερη ακρίβεια χρειάστηκε να ελαττωθεί η ταχύτητα πλοήγησης προκειμένου να αποφευχθεί επιπλέον μείωση της ποιότητας της εικόνας.

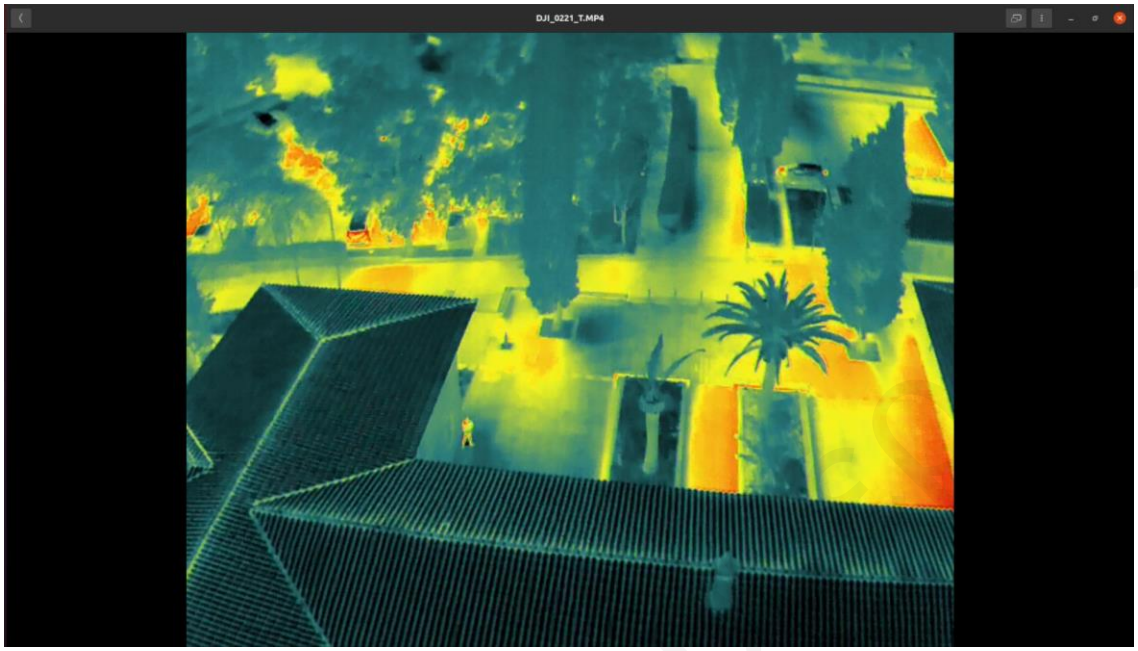


Εικόνα 10-3: Καθορισμός παραμέτρων πτήσης.

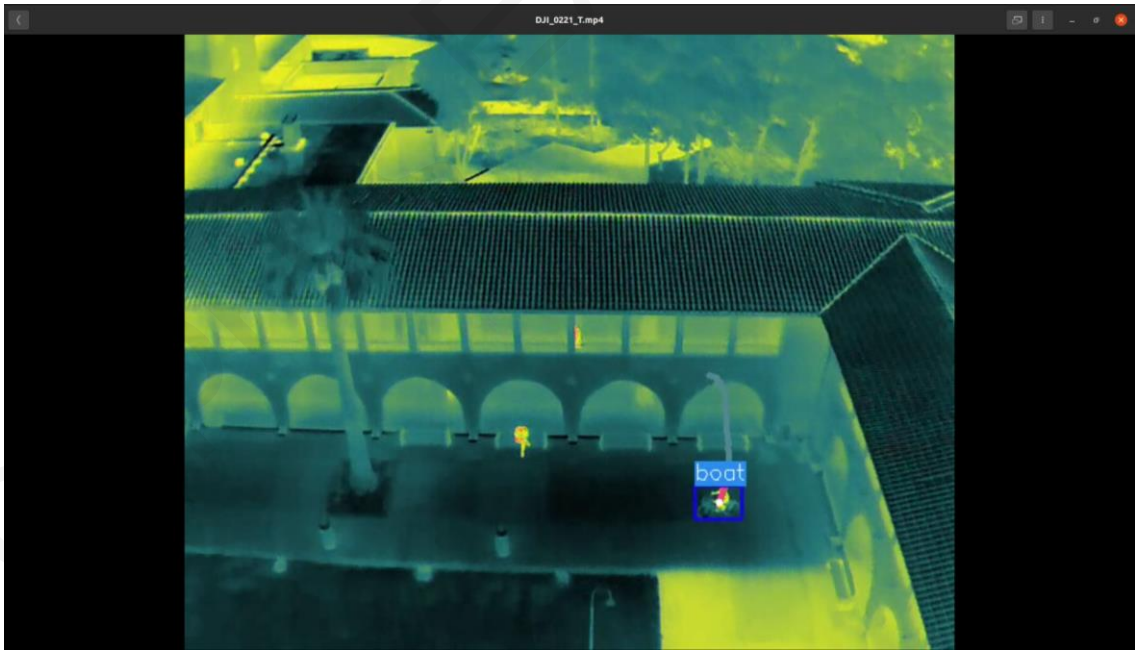
10.4 Αποτελέσματα

Οι δύο περιπτώσεις που ακολουθούν βασίζονται στους δύο τρόπους προσέγγισης της εφαρμογής με βασική διαφορά τους συντελεστές βαρύτητας. Με σειρά καταγραφής λήψης παρουσιάζονται πρώτα τα αποτελέσματα από ανίχνευση με weights συνόλου δεδομένων που περιείχε κανονικές εικόνες. Μετέπειτα παρουσιάζονται αποτελέσματα ανίχνευσης με weights από εικόνες θερμικής κάμερας που αναδεικνύουν τις δυνατότητες, τα χαρακτηριστικά και τους περιορισμούς της μεθοδολογίας. Η αναφορά στην εφαρμογή ολοκληρώνεται με σύγκριση και σχολιασμό των αποτελεσμάτων.

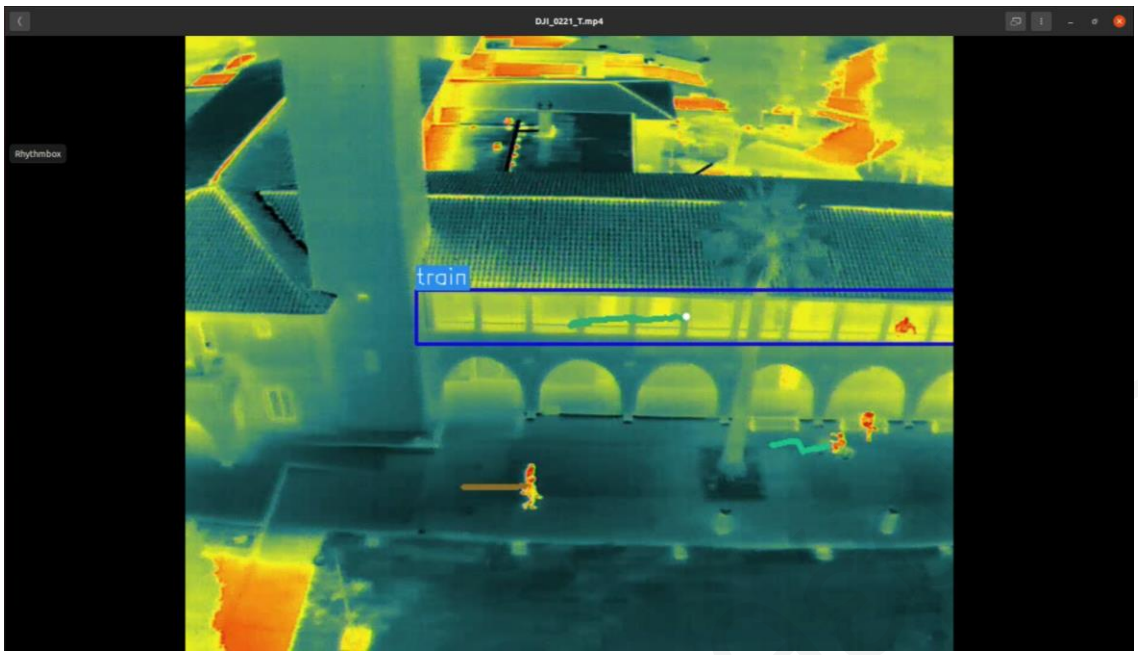
10.4.1 Αποτελέσματα από weights κανονικών εικόνων



Εικόνα 10-4: Στιγμιότυπο οθόνης θερμική κάμερας με αρνητικό bounding box.

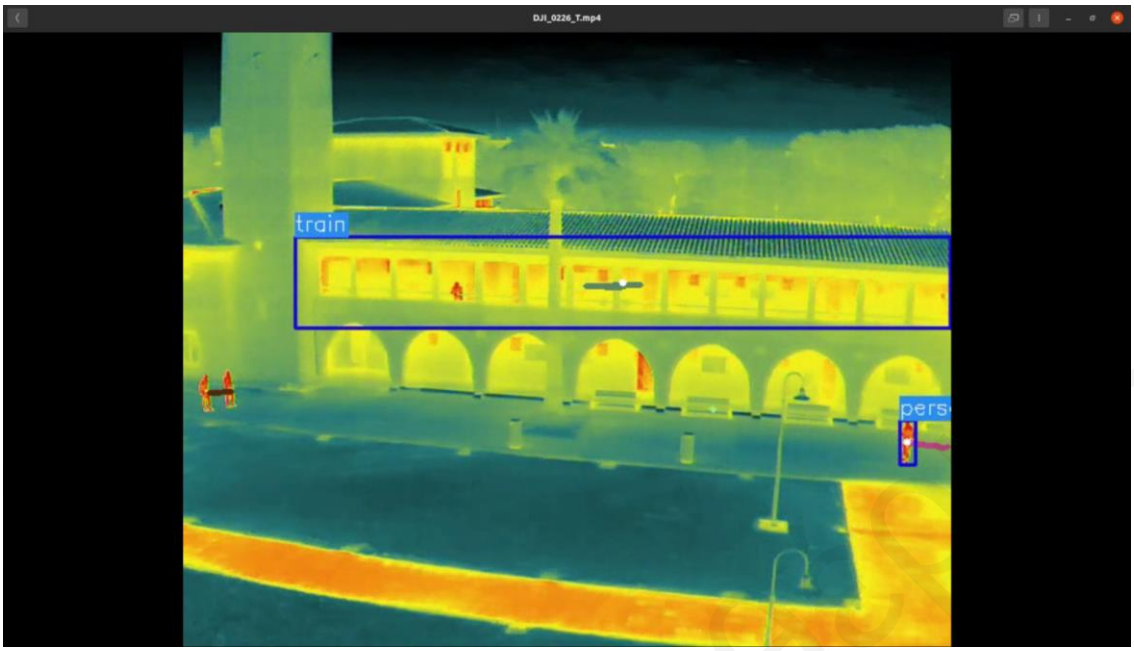


Εικόνα 10-5: Στιγμιότυπο οθόνης θερμικής κάμερας με αρνητικά bounding boxes και λανθασμένη ετικέτα.

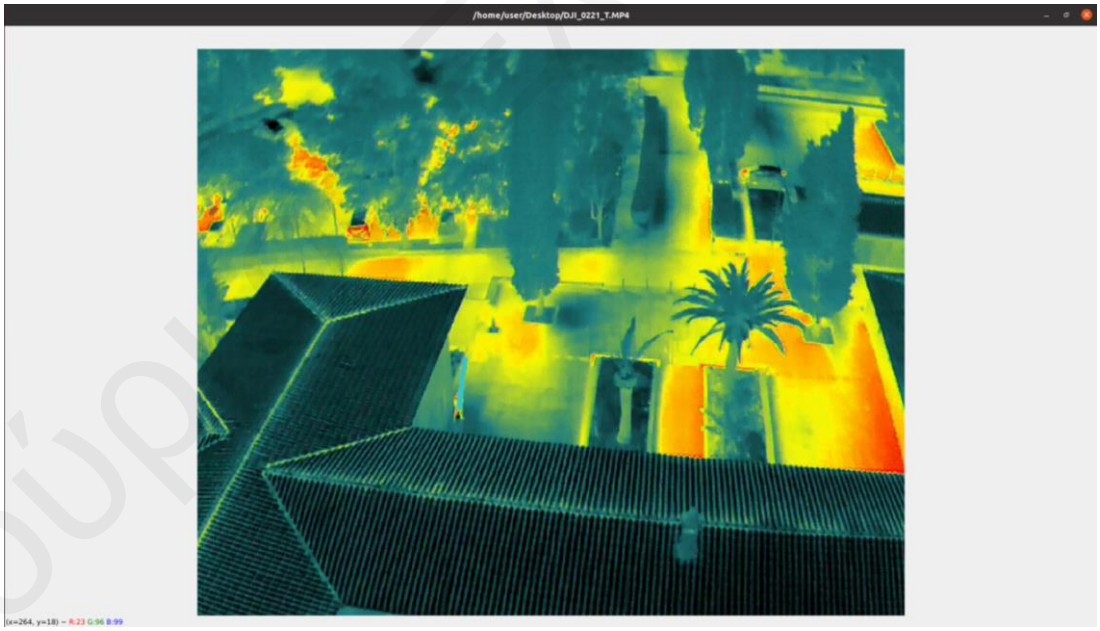


Εικόνα 10-6: Στιγμιότυπο οθόνης θερμικής κάμερας με πολλά αρνητικά bounding boxes και λανθασμένη ετικέτα.

Εικόνα 10-7: Στιγμιότυπο οθόνης θερμικής κάμερας με σωστή ετικέτα και δύο αρνητικά bounding boxes.

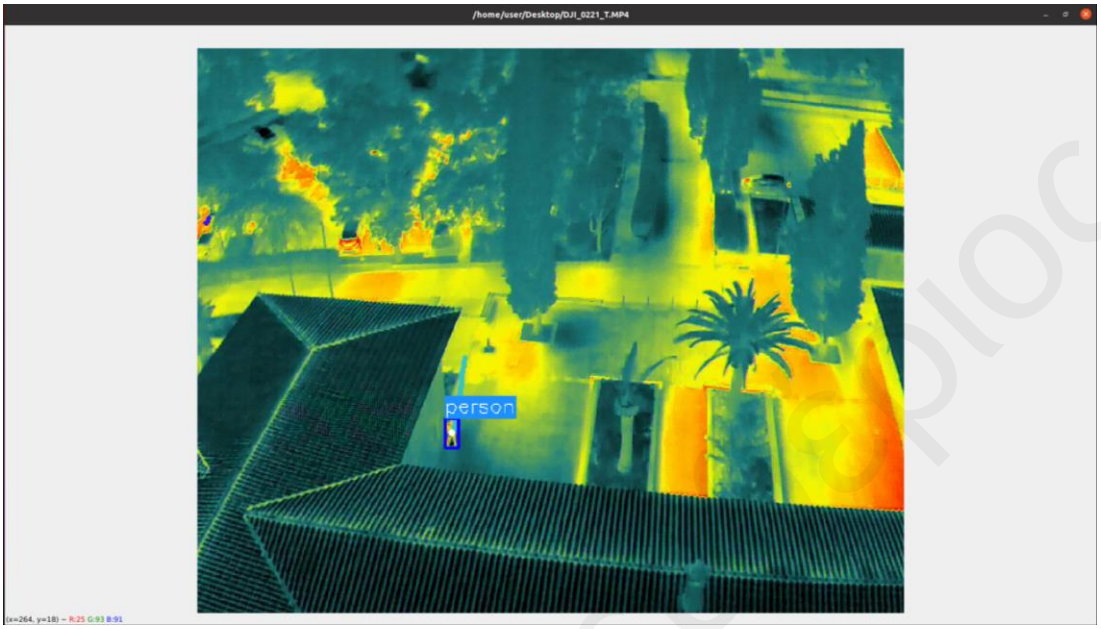


Εικόνα 10-8: Στιγμιότυπο οθόνης θερμικής κάμερας με ορθά και λανθασμένα αποτελέσματα.

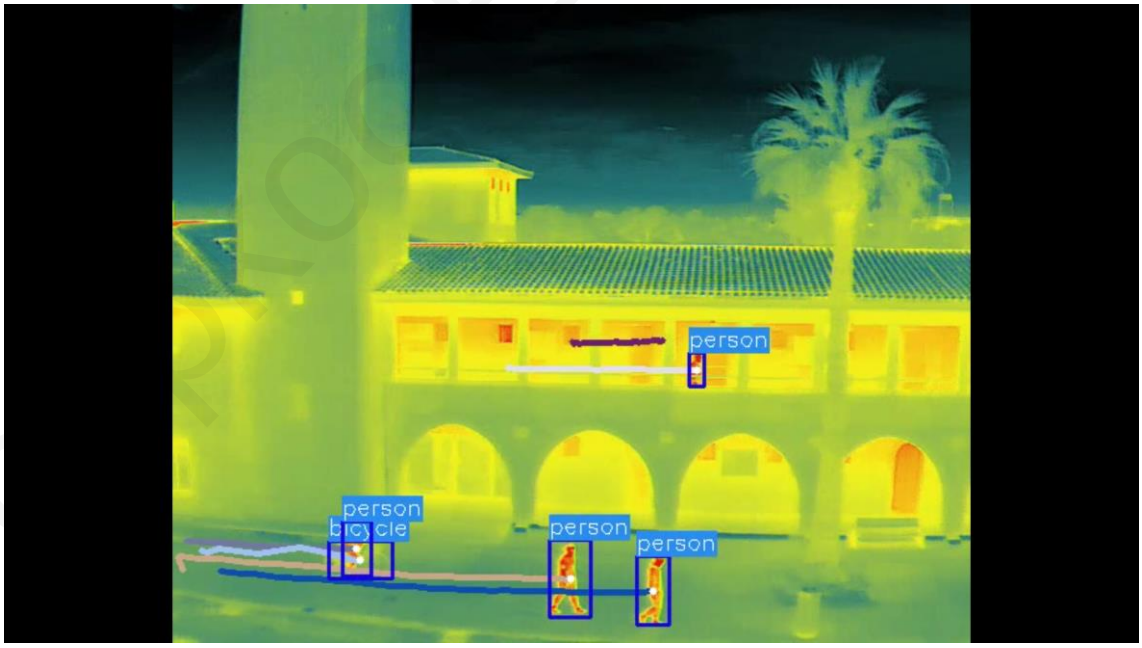


Εικόνα 10-9: Στιγμιότυπο οθόνης θερμικής κάμερας με μη σταθερό bounding box.

10.4.2 Αποτελέσματα από weights μετά την συνένωση των δύο συνόλων δεδομένων

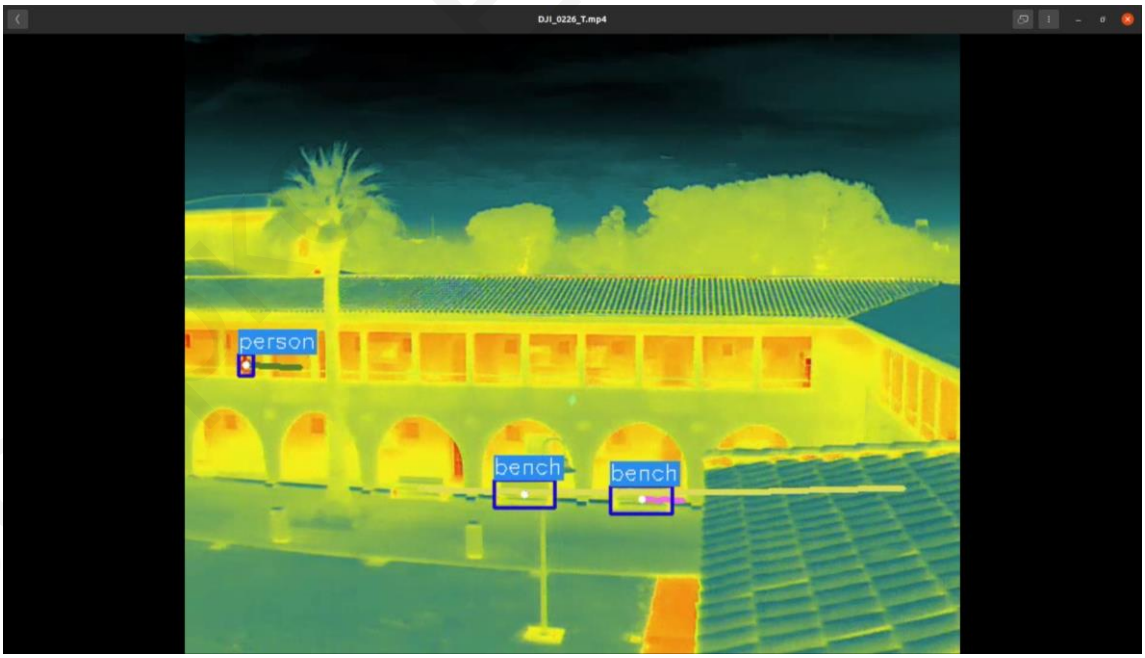


Εικόνα 10-10: Στιγμιότυπο οθόνης θερμικής κάμερας με σταθερό bounding box.

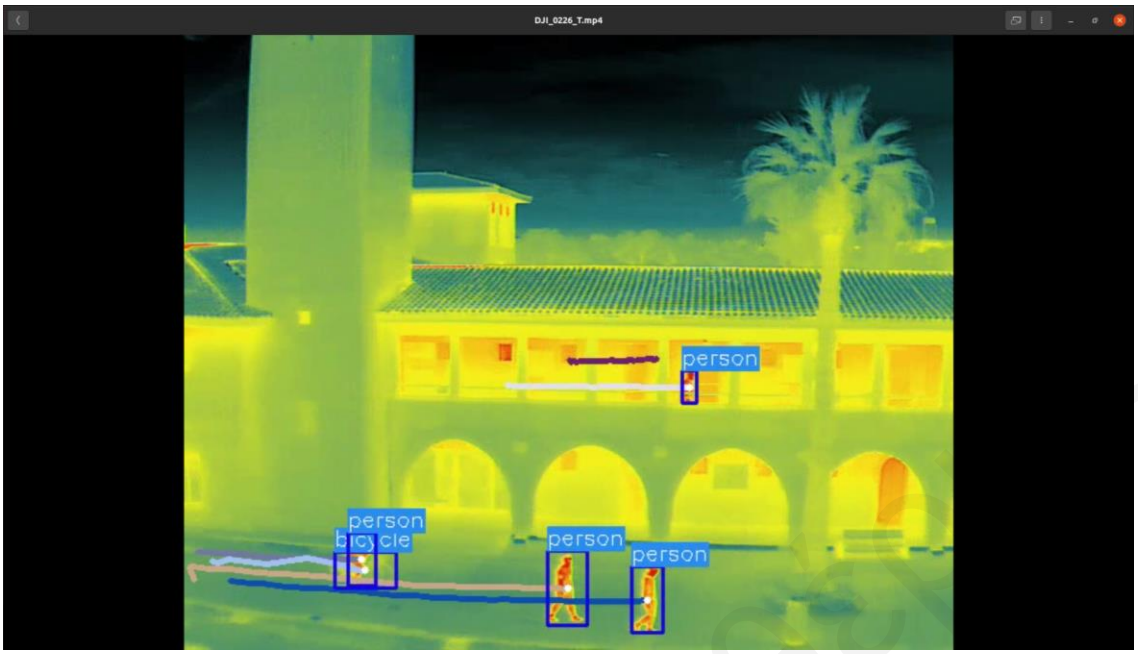


Εικόνα 10-11: Στιγμιότυπο οθόνης θερμικής κάμερας με όλα τα bounding boxes και labels σωστά.

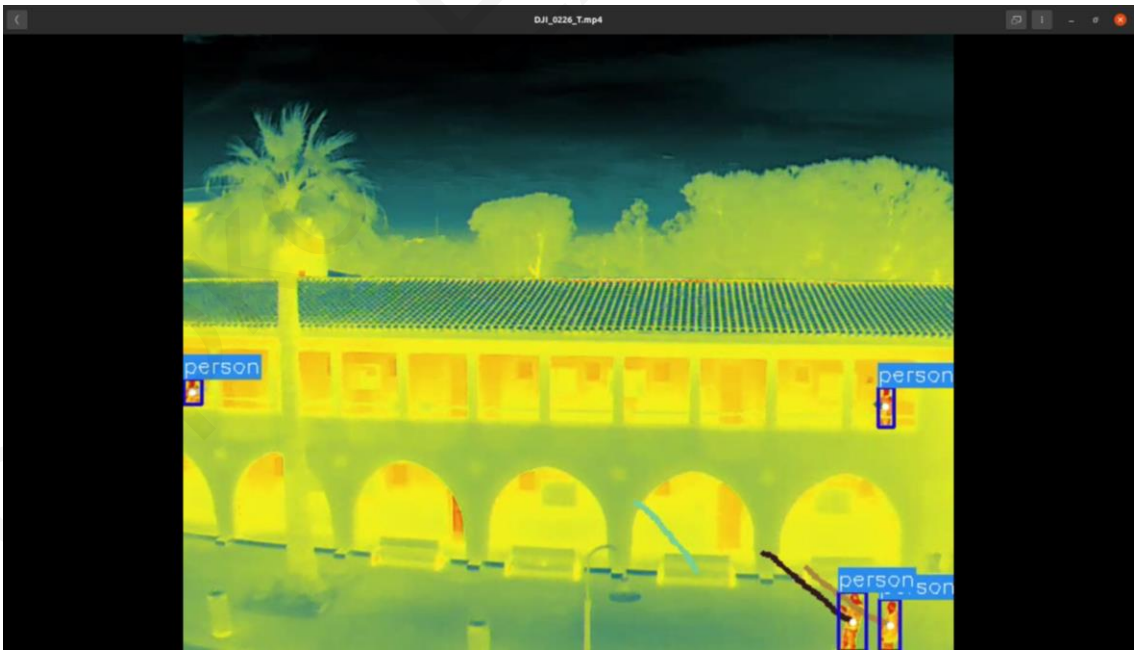
Εικόνα 10-12: Στιγμιότυπο οθόνης θερμικής κάμερας με ανίχνευση σε μακρινή απόσταση.



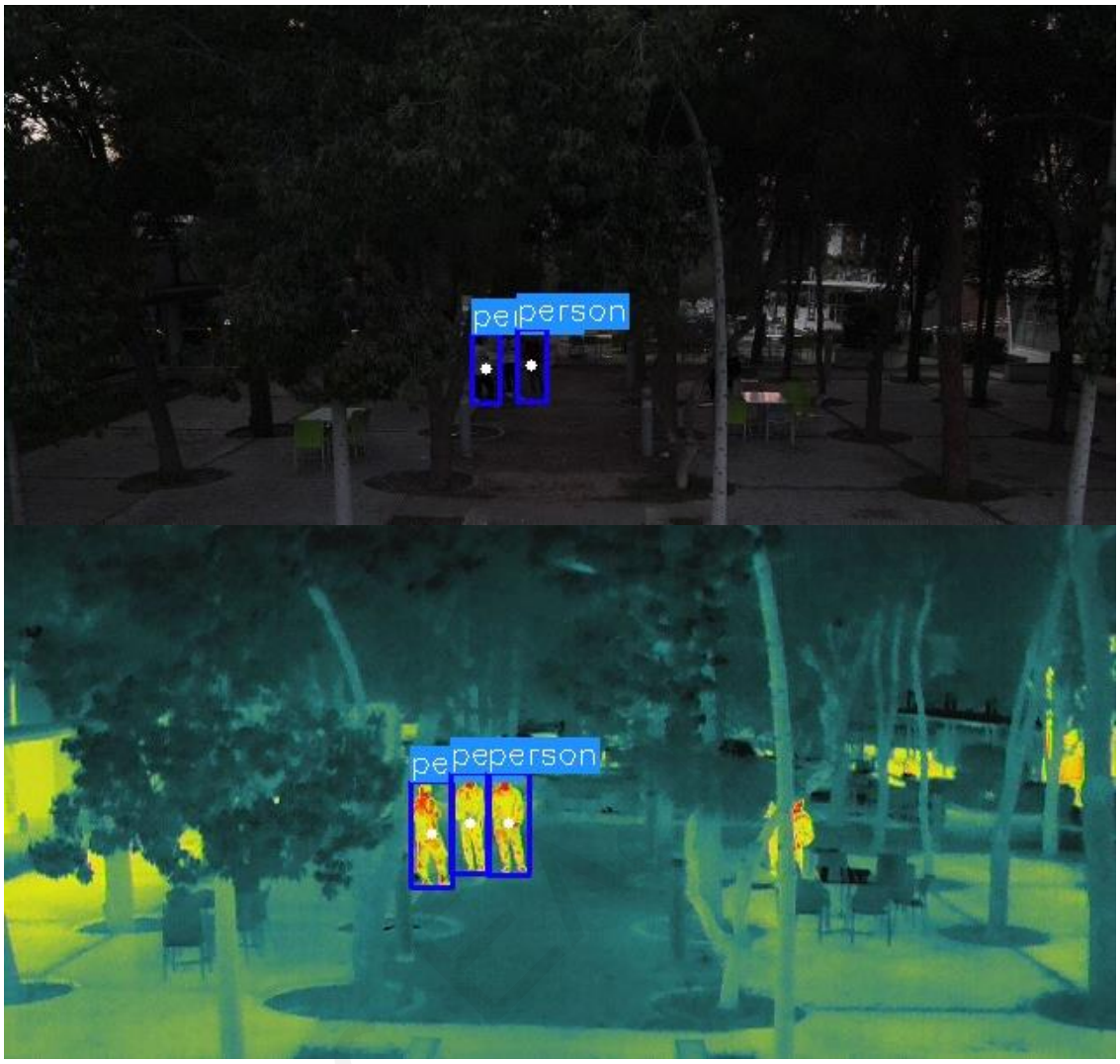
Εικόνα 10-13: Στιγμιότυπο οθόνης θερμικής κάμερας με ανίχνευση σε δύσκολη τοποθεσία.



Εικόνα 10-14: Στιγμιότυπο οθόνης θερμικής κάμερας με ανίχνευση διαφορετικών κλάσεων.



Εικόνα 10-15: Στιγμιότυπο οθόνης θερμικής κάμερας με ανίχνευση σε διαφορετικής δυσκολίας τοποθεσίες.



Εικόνα 10-16: Στιγμιότυπο ταυτόχρονης λήψης από τις δύο κάμερες με weights=yolov8s.pt.

10.4.3 Σχολιασμός αποτελεσμάτων

Η διαφορά στα αποτελέσματα με την αλλαγή των weights είναι εμφανής. Με τα weights που δημιουργήθηκαν από το σύνολο δεδομένων κανονικών λήψεων το μοντέλο δεν αποδίδει καθόλου καλά. Παρουσιάζει σημαντικό αριθμό αρνητικών bounding boxes, καθώς επίσης και αρκετές λάθος ετικέτες. Κατά τη διάρκεια εντοπισμού ανθρώπων σε πραγματικό χρόνο, ακόμη και τα σωστά bounding boxes που εντόπιζε δεν τα διατηρούσε για χρονικό διάστημα πέραν των 2 – 3 δευτερολέπτων όπως φαίνεται και στην Εικόνα 10-9 στιγμιότυπο το οποίο πάρθηκε μόλις μετά την απώλεια του bounding box κάτι που επιβεβαιώνει και η γραμμή πορείας η οποία εξακολουθεί να υφίσταται. Σε παρόμοια λήψη, όπως στην Εικόνα 10-4 το μοντέλο παρουσίαζε αρνητικό bounding box καθ’ όλη την διάρκεια της λήψης.

Στην αντίπερα όχθη, με την εκπαίδευση του μοντέλου σε σύνολο δεδομένων εικόνων και από θερμική κάμερα, τα αποτελέσματα είναι αρκετά πιο αισιόδοξα. Το μοντέλο παρουσιάζει ελάχιστα αρνητικά bounding boxes τα οποία ανακτά εκ νέου την επόμενη στιγμή. Παράλληλα, καθ' όλη τη διάρκεια του εντοπισμού σε πραγματικό χρόνο δεν παρουσίασε καμία λανθασμένη ένδειξη στις κλάσεις. Εντοπίζει με μεγάλη επιτυχία ακόμη και τους ανθρώπους που βρίσκονται στον πρώτο όροφο από μακρινές λήψεις και δύσκολες οπτικές γωνίες. Αξίζει να σημειωθεί ότι το μοντέλο παρουσιάζει υψηλά ποσοστά αξιοπιστίας τόσο κατά τη διάρκεια της ημέρας, όσο και κατά την διάρκεια της νύχτας.

Τέλος, παράλειψη θα αποτελούσε να μην γίνει μία σύγκριση στις Εικόνες 10-16. Γίνεται ξεκάθαρο ότι οι δύο φωτογραφίες πάρθηκαν την ίδια στιγμή από το ίδιο σημείο. Επίσης, για τον εντοπισμό χρησιμοποιήθηκαν weights τα οποία δεν δημιουργήθηκαν από σύνολα δεδομένων με φωτογραφίες από θερμική κάμερα. Ωστόσο, στην εικόνα από τη θερμική κάμερα, η ανίχνευση των τριών ανθρώπων γίνεται με απόλυτη επιτυχία. Από την άλλη, η περιορισμένη ορατότητα λόγω καιρικών συνθηκών εμπόδισε το μοντέλο να εντοπίσει όλους τους ανθρώπους, παρά μόνο δύο από αυτούς. Το συγκεκριμένο σημείο της πειραματικής διαδικασίας επιβεβαιώνει την ανάγκη για αξιοποίηση της συγκεκριμένης εφαρμογής.

Κεφάλαιο 11

11 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα Διατριβή επικεντρώνεται στο θέμα της ασφάλειας εγκαταστάσεων και αφορά δύο τεχνολογικές περιοχές. Τον έλεγχο και λειτουργία κινητών ρομπότ, καθώς επίσης και την ρομποτική όραση. Όσον αφορά την πρώτη περιοχή χρησιμοποιήθηκαν δύο ρομπότ. Το τροχοφόρο SUMMIT XL και το drone MAVIC 2 τα οποία χρησιμοποιήθηκαν σε διάφορες εφαρμογές ανάλογα με τις δυνατότητες και τα χαρακτηριστικά του καθενός. Σε σχέση με την ρομποτική όραση έγινε χρήση συμβατικής κάμερας καθώς επίσης θερμικής κάμερας.

Σχετικά με την ρομποτική όραση, χρησιμοποιήθηκε ο αλγόριθμος ανίχνευσης αντικειμένων YOLO (You Only Look Once), προκειμένου να επιτευχθεί η ανίχνευση ανθρώπων τόσο σε κτηριακές εγκαταστάσεις, όσο και σε ανοικτές δομές. Μέσω της επικοινωνίας ROS Noetic και YOLOv7 κατέστη δυνατό να επιτευχθεί ανίχνευση σε πραγματικό χρόνο και μεταφορά δεδομένων σε τοπικό δίκτυο, όσον αφορά τις εφαρμογές που εμπλέκεται το SUMMIT XL. Για το MAVIC 2 η ανίχνευση σε πραγματικό χρόνο γίνεται με το YOLOv8.

Αναφορικά με τις εφαρμογές, επιλέχθηκαν για φρούρηση οι παλαιές εγκαταστάσεις του Πανεπιστημίου Κύπρου στην οδό Καλλιπόλεως. Υπήρξε εστίαση σε δύο περιοχές, μία ανοιχτή δομή μεγάλης έκτασης που περιβάλλεται από κτηριακές υποδομές σε ισόγειο και πρώτο όροφο και μία μικρότερη δεντρόφυτη δομή με επιπλέον φυσικά εμπόδια.

Στην κάθε περίπτωση χρησιμοποιήθηκε το κατάλληλο ρομπότ σε συνδυασμό με την κατάλληλη έκδοση του αλγορίθμου προκειμένου να ικανοποιηθούν οι απαιτήσεις της κάθε εφαρμογής. Και στις δύο περιπτώσεις κατέστη δυνατό ο εντοπισμός ανθρώπων να γίνεται με μεγάλη ακρίβεια σε πραγματικό χρόνο. Παράλληλα επιτεύχθηκε η ανάκτηση των δεδομένων στον σταθμό ελέγχου με δυνατότητα μεταφοράς σε πραγματικό χρόνο είτε μέσω τοπικού δικτύου (local network) ή μέσω διευρυμένου δικτύου (global network).

Με την επιτυχή ολοκλήρωση των πιο πάνω εφαρμογών ανοίγονται νέοι ορίζοντες στον τομέα του ελέγχου των ρομπότ σε συνδυασμό με ρομποτική όραση.

Όπως αποδείχθηκε το YOLOv8 παρουσιάζει καλύτερες αποδόσεις συγκριτικά με το YOLOv7. Η επικοινωνία λοιπόν του ROS σε συνδυασμό με το YOLOv8 για ανίχνευση αντικειμένων σε πραγματικό χρόνο αποτελεί μία νέα μεγάλη πρόκληση.

Την ίδια στιγμή πρόκληση αποτελεί και η ολοκλήρωση των υφιστάμενων μοντέλων σε διαφορετικές εφαρμογές. Εφαρμογές που παρουσιάζουν αυξημένο ενδιαφέρον το τελευταίο διάστημα είναι για παράδειγμα η χρήση νοσηλευτικών ρομπότ για υποστήριξη του νοσηλευτικού προσωπικού σε νοσοκομειακές εγκαταστάσεις καθώς επίσης ένταξη της ανίχνευσης ανθρώπου σε στρατιωτικές επιχειρήσεις.

Τέλος, η αξιοποίηση του ρομποτικού βραχίονα που μπορεί να εγκατασταθεί στο τροχοφόρο ρομπότ θα προσέδιδε ένα τεράστιο πακέτο δεξιοτήτων κατατάσσοντας το ρομποτικό σύστημα σε άλλες διαστάσεις. Τέτοιας φύσης εργαλείο σε συνδυασμό με την ρομποτική όραση αναζητείται σε εφαρμογές υψηλής επικινδυνότητας όπως είναι για παράδειγμα ο εντοπισμός, προσέγγιση και εξουδετέρωση εκρηκτικών μηχανισμών. Επίσης, η χρήση του συστήματος κινητού βραχίονα προσδίδει την δυνατότητα υποστήριξης ηλικιωμένων ανθρώπων ή ανθρώπων με αναπηρίες στο σπίτι ανοίγοντας ένα νέο κύκλο εφαρμογών που πρόκειται να διερευνηθούν στο εργαστήριό μας.

Βιβλιογραφία

[1] AliExpress (2022). Wifi умный робот автомобильный комплект для Arduino/Hd камеры Ds робот умный Обучающий робот комплект | Электронные компоненты и принадлежности | АлиЭкспресс. [online] aliexpress.ru. Available at: https://aliexpress.ru/item/32654734117.html?gatewayAdapt=glo2rus&sku_id=59808628049.

[2] Butler, S. (2022). *Home Security Robots Are Already Here to Protect You*. [online] How-To Geek. Available at: https://www.howtogeek.com/786216/home-security-robots-are-already-here-to-protect-you/.

[3] Fossati, A., Juergen Gall, Helmut Grabner, Xiaofeng Ren and Konolige, K. (2013). *Consumer Depth Cameras for Computer Vision Research Topics and Applications*. London Springer.

[4] Santora, M. (2018). *5 Challenges in Creating Autonomous Navigation Systems*. [online] The Robot Report. Available at: https://www.therobotreport.com/autonomous-navigation-design-challenges/.

[5] Simon, G. and László Sujbert (2021). Recent Advances in Indoor Localization Systems and Technologies. MDPI.

[6] AliExpress (2022). Wifi умный робот автомобильный комплект для Arduino/Hd камеры Ds робот умный Обучающий робот комплект | Электронные компоненты и принадлежности | АлиЭкспресс. [online] aliexpress.ru. Available at: https://aliexpress.ru/item/32654734117.html?gatewayAdapt=glo2rus&sku_id=59808628049.

[7] Bell, V. (2018). *Robocop-style security guards used in a LA shopping mall*. [online] Mail Online. Available at: https://www.dailymail.co.uk/sciencetech/article-6508367/Robocop-style-security-guards-used-LA-shopping-mall.html.

[8] Butler, S. (2022). *Home Security Robots Are Already Here to Protect You*. [online] How-To Geek. Available at: https://www.howtogeek.com/786216/home-security-robots-are-already-here-to-protect-you/.

[9] Daily Mail (2014). *Robot security guards now patrolling Microsoft's Silicon Valley campus*. [online] CBC. Available at: https://www.cbc.ca/news/canada/calgary/robot-security-guards-now-patrolling-microsoft-s-silicon-valley-campus-1.2847329.

[10] Fossati, A., Juergen Gall, Helmut Grabner, Xiaofeng Ren and Konolige, K. (2013). *Consumer Depth Cameras for Computer Vision Research Topics and Applications*. London Springer.

[11] iStock (2021). *Diamond Museum Stock Photos, Pictures & Royalty-Free Images - iStock*. [online] www.istockphoto.com. Available at: https://www.istockphoto.com/photos/diamond-museum.

[12] Open eVision (2022). *Image Coordinate Systems*. [online] documentation.euresys.com. Available at: https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/03_Using/1_Starting_Up/3_Manipulating/Image_Coordinate_Systems.htm?TocPath=Using%20Open%20eVision%7CManipulating%20Pixels%20Containers%20and%20Files%7C_____5.

[13] Quick, D. (2010). *Battlefield Extraction-Assist Robot to ferry wounded to safety*. [online] New Atlas. Available at: https://newatlas.com/battlefield-extraction-assist-robot/17059/.

[14] Robosecurity (2020). *Robosecurity Is Here: How Robots Can Save Money and Lives*. [online] Security Sales & Integration. Available at: https://www.securitysales.com/emerging-tech/robotics-tech/robosecurity-save-money-lives/.

[15] Santora, M. (2018). *5 Challenges in Creating Autonomous Navigation Systems*. [online] The Robot Report. Available at: https://www.therobotreport.com/autonomous-navigation-design-challenges/.

[16] Simon, G. and László Sujbert (2021). Recent Advances in Indoor Localization Systems and Technologies. MDPI.

[17] US Government (2019). *Snapshot: S&T Test Methods Used to Evaluate Rescue Robots | Homeland Security*. [online] www.dhs.gov. Available at: https://www.dhs.gov/science-and-technology/news/2018/08/07/snapshot-st-test-methods-used-evaluate-rescue-robots.

[18] Rosebrock, Adrian. *Deep Learning for Computer Vision with Python Vol. 3, ImageNet Bundle*. Philadelphia, Pa] Adrian Rosebrock, Pyimagesearch.com, 2019.

[19] Pu, Yuanyuan, et al. "Machine Learning Methods for Rockburst Prediction-State-of-The-Art Review." *International Journal of Mining Science and*

Technology, vol. 29, no. 4, July 2019, pp. 565–570, https://doi.org/10.1016/j.ijmst.2019.06.009.

[20] Rohrer, Brandon. "How to Train Your Robot." *E2eml.school*, 2019, e2eml.school/convert_rgb_to_grayscale.html.

[21] Image Analysis Software Tools, Open eVision. "Image Coordinate Systems." *Documentation.euresys.com*, Dec. 2022, documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/03_Using/1_Starting_Up/3_Manipulating/Image_Coordinate_Systems.htm?TocPath=Using%20Open%20eVision%7CManipulating%20Pixels%20Containers%20and%20Files%7C_____5.

[22] MathWorks. "Image Coordinate Systems - MATLAB & Simulink - MathWorks Switzerland." *Ch.mathworks.com*, 2023, ch.mathworks.com/help/images/image-coordinate-systems.html.

[23] Baheti, Pragati . "Train, Validation, and Test Set: How to Split Your Machine Learning Data." *V7labs.com*, 2021, www.v7labs.com/blog/train-validation-test-set.

[24] Roboflow. "Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes." *Roboflow.ai*, 2021, roboflow.com/.

[25] Robotnik. "XL-GEN - Robotnik®." *Robotnik*, 2019, robotnik.eu/products/mobile-manipulators/xl-gen/.

[26] Generation, Robots. "Ned2: 6-Axis Robot Arm by Niryo." *Génération Robots*, 2020, www.generationrobots.com/en/403937-ned2-6-axis-robot-arm.html.

[27] Businesswire. "Knightscope Deploys New Autonomous Security Robot in Southern California." *Www.businesswire.com*, 16 Mar. 2022, www.businesswire.com/news/home/20220316005436/en/Knightscope-Deploys-New-Autonomous-Security-Robot-in-Southern-California.

[28] Google, Maps. "Google Maps." *Google Maps*, 2023, www.google.com/maps/@35.1599859.

[29] Ltd, Bossgoo Co. "Κίνα Αξεσουάρ Drone Flir Tau2 Υπέρυθρη Θερμική κάμερα Gimbal, υψηλής ποιότητας Αξεσουάρ Drone Flir Tau2 Υπέρυθρη Θερμική κάμερα Gimbal στο Bossgoo.com." *Gr.bossgoo.com*, gr.bossgoo.com/product-detail/drone-accessories-flir-tau2-infrared-thermal-53371063.html.

[30] "How to Label Your Drone." *Drones.gov.cy*, 2023, drones.gov.cy/wp-content/uploads/2021/06/UAS_how_to_label_your-drone-scaled.jpg.

[31] "Αισθητήρες λέιζερ - επισκόπηση τεχνολογίας." *Ifm.com*, 2023, www.ifm.com/gr/el/shared/technologien/magic-cube/%CE%B1%CE%B9%CF%83%CE%B8%CE%B7%CF%84%CE%AE%C

F%81%CE%B5%CF%82-
%CE%BB%CE%AD%CE%B9%CE%B6%CE%B5%CF%81.

[32] "Buzzer 22mm for Acoustic Signal Alarm for Control Panel 220 VAC in Red." *Cablematic.com*, 2021, cablematic.com/en/products/buzzer-22mm-for-acoustic-signal-alarm-for-control-panel-220-vac-in-red-TX083/.

[33] humans.txt. "Acoustic Indicator Complete Device Siemens SIRIUS ACT 3SU1200-6LB10-1AA0." *Www.automation24.Biz*, 2022, www.automation24.biz/acoustic-indicator-complete-device-siemens-sirius-act-3su1200-6lb10-1aa0.

[34] "Sloan - Indicator Lights." *Www.sloan.ch*, 2022, www.sloan.ch/en/Products/Indicator-Lights/?oid=52&lang=en.

[35] "Interface Ultrasonic Sensor with Arduino." *Robocraze*, 2023, robocraze.com/blogs/post/arduino-interfacing-with-ultrasonic-sensor.

[36] Olinski, Michał, et al. "Human Motion Characterization Using Wireless Inertial Sensors Adaptive Gas Turbine Drive View Project Legs Exoskeleton Robot\ Football Robot View Project Human Motion Characterization Using Wireless Inertial Sensors." *Human Motion Characterization Using Wireless Inertial Sensors*, 2017, https://doi.org/10.1007/978-3-319-45450-4_40.

[37] Fox, Dieter. "Adaptive Monte Carlo Localization." *Robotics Knowledgebase*, 3 Feb. 2020, roboticsknowledgebase.com/wiki/state-estimation/adaptive-monte-carlo-localization/.

[38] Fox Dieter et al. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. Jan. 1999.

[39] Vasanth Vidyakar (2019). *Surveillance Robot*. [online] Skyfilabs.com. Available at: https://www.skyfilabs.com/project-ideas/surveillance-robot.

[40] Hertz, L. (2022). *Human Presence Detection System | Coronavirus Projects*. [online] LeewayHertz - Software Development Company. Available at: https://www.leewayhertz.com/human-presence-detection-system/.

[41] Wäldchen, J. and Mäder, P. (2017). Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review. *Archives of Computational Methods in Engineering*, 25(2), pp.507–543. doi:https://doi.org/10.1007/s11831-016-9206-z.

[42] Sheridan T.B., "Human-Robot Interacion: Status and Challenges" *Human Factors*, 2016, vol. 58, no4, pp. 525-532.

[43] Sebastian Hjorth, Dimitrios Chrysostomou, "Human–robot collaboration in industrial environments: A literature review on non-destructive disassembly", *Robotics and Computer-Integrated Manufacturing*, 2022, vol. 73, 102208, ISSN 0736-5845

[44] Goodrich M.A. and Schultz A.C., "Human-Robot Interaction: A Survey," *Foundations and Trends in Human-Computer Interaction*, 2017, vol. 1 no.3, pp. 203-275.

[45] Alzubaidi, L., Zhang, J., Humaidi, A.J. *et al.* "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." *J Big Data*, 2021, **8**, 53

[46] Redmon J. et. al., "You only Look Once: Unified, Real-Time Object Detection", *arXiv*: 1506.02640v5, 2016.

[47] Jocher, G. (2023b). *YOLOv5 Documentation*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/#yolov5.

[48] ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations (2020)

[49] Sharma A., "Training the YOLOv5 Object Detector on a Custom Dataset," *PyImageSearch*, D. Chakraborty, *et al.*, eds., 2022.

[50] Cartucho (2022). *OpenLabeling: open-source image and video labeler*. [online] GitHub. Available at: https://github.com/Cartucho/OpenLabeling.

[51] Jocher, G. (2023). *Ultralytics | Revolutionizing the World of Vision AI*. [online] Ultralytics. Available at: https://ultralytics.com/yolov8.

[52] Jocher, G., Chaurasia, A. and Qiu, J. (2023). *YOLO by Ultralytics*. [online] GitHub. Available at: https://github.com/ultralytics/ultralytics.

[53] Kukil and Rath, S. (2022). *YOLOv7 Paper Explanation: Object Detection and YOLOv7 Pose*. [online] LearnOpenCV. Available at: https://learnopencv.com/yolov7-object-detection-paper-explanation-and-inference/.

[54] Solawetz, J. and Jan 11, F. (2023). *What is YOLOv8? The Ultimate Guide.* [online] Roboflow Blog. Available at: https://blog.roboflow.com/whats-new-in-yolov8/.

[55] Wong, K.-Y. (2022). *Official YOLOv7*. [online] GitHub. Available at: https://github.com/WongKinYiu/yolov7.

[56] Ultralytics (2023). *YOLOv5*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/models/yolov5/#supported-modes.

[57] Jocher, G. and Chaurasia, A. (2020). *YOLOv5 Documentation*. [online] docs.ultralytics.com. Available at: https://docs.ultralytics.com/.

[58] Ewecker, L. (2023). *ROS package for official YOLOv7*. [online] GitHub. Available at: https://github.com/lukazso/yolov7-ros.

[59] Παπαθανασίου, Μ. (2008). *Max Frisch-Αποφθέγματα*. [online] Γνωμικολογικόν. Available at: https://www.gnomikologikon.gr/authquotes.php?auth=866.

[60] Fossati, A., Juergen Gall, Helmut Grabner, Xiaofeng Ren and Konolige, K. (2013). *Consumer Depth Cameras for Computer Vision Research Topics and Applications*. London Springer.

[61] Wilber (2017). *GIMP - GIMP 2.8.22 Released*. [online] www.gimp.org. Available at: https://www.gimp.org/news/2017/05/11/gimp-2-8-22-released/.

[62] Nelson, J. (2020). *Thermal Dogs and People Object Detection Dataset*. [online] Roboflow. Available at: https://public.roboflow.com/object-detection/thermal-dogs-and-people?ref=roboflow-blog.

[63] Vourkos, E., Toulkeridou, E., Kourris, A., Ros, R.J., Christoforou, E., Ramdani, N. and Panayides, A. (2023). Safe Robot Navigation in Indoor Healthcare Workspaces. In: *The 20th International Conference on Computer Analysis of Images and Patterns*. Springer.

Παραρτήματα

Παράρτημα Ι – Λογισμικό Ελέγχου του Ρομπότ

#start_mapping.launch

```
<launch>

  <!--- Run gmapping -->
  <include file="$(find my_summit_xl_tools)/launch/gmapping.launch" />

  <!--- Run Move Base -->
  <include file="$(find my_summit_xl_tools)/launch/move_base_map.launch" />

</launch>
```

#start_navigation_with_map.launch

```
<launch>

    <!-- Run the map server -->
    <arg name="map_file" default="$(find my_summit_xl_tools)/maps/mymap.yaml"/>
    <node name="map_server" pkg="map_server" type="map_server" args="$(arg
map_file)" />

    <!--- Run AMCL -->
    <include file="$(find my_summit_xl_tools)/launch/amcl.launch" />

    <!--- Run Move Base -->
    <include file="$(find my_summit_xl_tools)/launch/move_base_map.launch" />

</launch>
```

#follow_waypoints.launch

```
<launch>
  <env name="ROSCONSOLE_FORMAT"
value="[${severity}][${thread}][${node}/${function}:${line}]: ${message}"/>

 <arg name="addpose_topic" default="/initialpose"/>
 <arg name="posearray_topic" default="/waypoints"/>

  <node pkg="follow_waypoints" type="follow_waypoints" name="follow_waypoints"
output="screen" clear_params="true">
    <param name="goal_frame_id" value="map"/>
    <param name="addpose_topic" value="$(arg addpose_topic)"/>
    <param name="posearray_topic" value="$(arg posearray_topic)"/>
  </node>
</launch>
```

Παράρτημα ΙΙ – Λογισμικό Ανίχνευσης σε πραγματικό χρόνο

#yolov7.webcam.launch

```
<launch>
   <arg name="video_device"          default="/dev/video2"/>
   <arg name="pixel_format"          default="yuyv"/>
   <node pkg="usb_cam" name="usb_cam" type="usb_cam_node" >
      <param name="video_device"          value="$(arg video_device)" />
      <param name="pixel_format"          value="$(arg pixel_format)" />
      <remap from="/usb_cam/image_raw" to="/image_raw" />
   </node>
   <arg name="conf_thresh"           default="0.5"/>
   <arg name="weights"               default="/home/user/yolov7/best.pt"/>
   <include file="$(find  yolov7_ros)/launch/yolov7.launch">
      <arg name="conf_thresh"           value="$(arg conf_thresh)"/>
      <arg name="weights"               value="$(arg weights)"/>
   </include>
</launch>
```

#yolov7.launch

```
<launch>
   <!-- Detection configuration -->
   <arg name="weights"             default="/home/user/yolov7/best.pt"/>
   <arg name="conf_thresh"         default="0.25"/>
   <arg name="iou_thresh"          default="0.45"/>
   <arg name="device"              default="cuda"/>
   <arg name="img_size"            default="640"/>

   <!-- Visualize using OpenCV window -->
   <arg name="visualize"           default="true"/>

   <!-- ROS topics -->
   <arg name="input_img_topic"      default="/image_raw"/>
   <arg name="output_img_topic"     default="/yolov7/image_raw"/>
   <arg name="output_topic"         default="/yolov7/detections"/>

   <node pkg="yolov7_ros" name="detect" type="detect.py" output="screen"
ns="yolov7">
      <param name="weights"             value="$(arg weights)"/>
      <param name="conf_thresh"         value="$(arg conf_thresh)"/>
      <param name="iou_thresh"          value="$(arg iou_thresh)" />
      <param name="device"              value="$(arg device)" />
      <param name="input_img_topic"     value="$(arg input_img_topic)"/>
      <param name="output_img_topic"    value="$(arg output_img_topic)"/>
      <param name="output_topic"        value="$(arg output_topic)"/>
      <param name="visualize"           value="$(arg visualize)"/>
   </node>
</launch>
```

Παράρτημα ΙΙΙ – Λογισμικό Συστήματος όρασης

#YOLOv5

#train.py

```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
"""
Train a YOLOv5 model on a custom dataset.
Models and datasets download automatically from the latest YOLOv5 release.

Usage - Single-GPU training:
    $ python train.py --data coco128.yaml --weights yolov5s.pt --img 640  # from
pretrained (recommended)
    $ python train.py --data coco128.yaml --weights '' --cfg yolov5s.yaml --img 640  #
from scratch

Usage - Multi-GPU DDP training:
    $ python -m torch.distributed.run --nproc_per_node 4 --master_port 1 train.py --data
coco128.yaml --weights yolov5s.pt --img 640 --device 0,1,2,3

Models:     https://github.com/ultralytics/yolov5/tree/master/models
Datasets:   https://github.com/ultralytics/yolov5/tree/master/data
Tutorial:   https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data
"""

import argparse
import math
import os
import random
import sys
import time
from copy import deepcopy
from datetime import datetime
from pathlib import Path

import numpy as np
import torch
import torch.distributed as dist
import torch.nn as nn
import yaml
from torch.optim import lr_scheduler
from tqdm import tqdm

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative
```

```
import val as validate  # for end-of-epoch mAP
from models.experimental import attempt_load
from models.yolo import Model
from utils.autoanchor import check_anchors
from utils.autobatch import check_train_batch_size
from utils.callbacks import Callbacks
from utils.dataloaders import create_dataloader
from utils.downloads import attempt_download, is_url
from utils.general import (LOGGER, check_amp, check_dataset, check_file,
check_git_status, check_img_size,
                           check_requirements, check_suffix, check_yaml, colorstr,
get_latest_run, increment_path,
                           init_seeds, intersect_dicts, labels_to_class_weights,
labels_to_image_weights, methods,
                           one_cycle, print_args, print_mutation, strip_optimizer, yaml_save)
from utils.loggers import Loggers
from utils.loggers.comet.comet_utils import check_comet_resume
from utils.loss import ComputeLoss
from utils.metrics import fitness
from utils.plots import plot_evolve
from utils.torch_utils import (EarlyStopping, ModelEMA, de_parallel, select_device,
smart_DDP, smart_optimizer,
                               smart_resume, torch_distributed_zero_first)

LOCAL_RANK = int(os.getenv('LOCAL_RANK', -1))  #
https://pytorch.org/docs/stable/elastic/run.html
RANK = int(os.getenv('RANK', -1))
WORLD_SIZE = int(os.getenv('WORLD_SIZE', 1))


def train(hyp, opt, device, callbacks):  # hyp is path/to/hyp.yaml or hyp dictionary
    save_dir, epochs, batch_size, weights, single_cls, evolve, data, cfg, resume, noval,
nosave, workers, freeze = \
        Path(opt.save_dir), opt.epochs, opt.batch_size, opt.weights, opt.single_cls,
opt.evolve, opt.data, opt.cfg, \
        opt.resume, opt.noval, opt.nosave, opt.workers, opt.freeze
    callbacks.run('on_pretrain_routine_start')

    # Directories
    w = save_dir / 'weights'  # weights dir
    (w.parent if evolve else w).mkdir(parents=True, exist_ok=True)  # make dir
    last, best = w / 'last.pt', w / 'best.pt'

    # Hyperparameters
    if isinstance(hyp, str):
        with open(hyp, errors='ignore') as f:
            hyp = yaml.safe_load(f)  # load hyps dict
    LOGGER.info(colorstr('hyperparameters: ') + ', '.join(f'{k}={v}' for k, v in
hyp.items()))
    opt.hyp = hyp.copy()  # for saving hyps to checkpoints
```

```
    # Save run settings
    if not evolve:
        yaml_save(save_dir / 'hyp.yaml', hyp)
        yaml_save(save_dir / 'opt.yaml', vars(opt))

    # Loggers
    data_dict = None
    if RANK in {-1, 0}:
        loggers = Loggers(save_dir, weights, opt, hyp, LOGGER)  # loggers instance

        # Register actions
        for k in methods(loggers):
            callbacks.register_action(k, callback=getattr(loggers, k))

        # Process custom dataset artifact link
        data_dict = loggers.remote_dataset
        if resume:  # If resuming runs from remote artifact
            weights, epochs, hyp, batch_size = opt.weights, opt.epochs, opt.hyp,
opt.batch_size

    # Config
    plots = not evolve and not opt.noplots  # create plots
    cuda = device.type != 'cpu'
    init_seeds(opt.seed + 1 + RANK, deterministic=True)
    with torch_distributed_zero_first(LOCAL_RANK):
        data_dict = data_dict or check_dataset(data)  # check if None
    train_path, val_path = data_dict['train'], data_dict['val']
    nc = 1 if single_cls else int(data_dict['nc'])  # number of classes
    names = {0: 'item'} if single_cls and len(data_dict['names']) != 1 else
data_dict['names']  # class names
    is_coco = isinstance(val_path, str) and val_path.endswith('coco/val2017.txt')  #
COCO dataset

    # Model
    check_suffix(weights, '.pt')  # check weights
    pretrained = weights.endswith('.pt')
    if pretrained:
        with torch_distributed_zero_first(LOCAL_RANK):
            weights = attempt_download(weights)  # download if not found locally
        ckpt = torch.load(weights, map_location='cpu')  # load checkpoint to CPU to avoid
CUDA memory leak
        model = Model(cfg or ckpt['model'].yaml, ch=3, nc=nc,
anchors=hyp.get('anchors')).to(device)  # create
        exclude = ['anchor'] if (cfg or hyp.get('anchors')) and not resume else []  # exclude
keys
        csd = ckpt['model'].float().state_dict()  # checkpoint state_dict as FP32
        csd = intersect_dicts(csd, model.state_dict(), exclude=exclude)  # intersect
        model.load_state_dict(csd, strict=False)  # load
        LOGGER.info(f'Transferred {len(csd)}/{len(model.state_dict())} items from
{weights}')  # report
    else:
```

```
        model = Model(cfg, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
    amp = check_amp(model)  # check AMP

    # Freeze
    freeze = [f'model.{x}.' for x in (freeze if len(freeze) > 1 else range(freeze[0]))]  #
layers to freeze
    for k, v in model.named_parameters():
        v.requires_grad = True  # train all layers
        # v.register_hook(lambda x: torch.nan_to_num(x))  # NaN to 0 (commented for
erratic training results)
        if any(x in k for x in freeze):
            LOGGER.info(f'freezing {k}')
            v.requires_grad = False

    # Image size
    gs = max(int(model.stride.max()), 32)  # grid size (max stride)
    imgsz = check_img_size(opt.imgsz, gs, floor=gs * 2)  # verify imgsz is gs-multiple

    # Batch size
    if RANK == -1 and batch_size == -1:  # single-GPU only, estimate best batch size
        batch_size = check_train_batch_size(model, imgsz, amp)
        loggers.on_params_update({"batch_size": batch_size})

    # Optimizer
    nbs = 64  # nominal batch size
    accumulate = max(round(nbs / batch_size), 1)  # accumulate loss before optimizing
    hyp['weight_decay'] *= batch_size * accumulate / nbs  # scale weight_decay
    optimizer = smart_optimizer(model, opt.optimizer, hyp['lr0'], hyp['momentum'],
hyp['weight_decay'])

    # Scheduler
    if opt.cos_lr:
        lf = one_cycle(1, hyp['lrf'], epochs)  # cosine 1->hyp['lrf']
    else:
        lf = lambda x: (1 - x / epochs) * (1.0 - hyp['lrf']) + hyp['lrf']  # linear
    scheduler = lr_scheduler.LambdaLR(optimizer, lr_lambda=lf)  #
plot_lr_scheduler(optimizer, scheduler, epochs)

    # EMA
    ema = ModelEMA(model) if RANK in {-1, 0} else None

    # Resume
    best_fitness, start_epoch = 0.0, 0
    if pretrained:
        if resume:
            best_fitness, start_epoch, epochs = smart_resume(ckpt, optimizer, ema, weights,
epochs, resume)
        del ckpt, csd

    # DP mode
    if cuda and RANK == -1 and torch.cuda.device_count() > 1:
```

```
        LOGGER.warning('WARNING ⚠ DP not recommended, use
torch.distributed.run for best DDP Multi-GPU results.\n'
                       'See Multi-GPU Tutorial at
https://github.com/ultralytics/yolov5/issues/475 to get started.')
        model = torch.nn.DataParallel(model)

    # SyncBatchNorm
    if opt.sync_bn and cuda and RANK != -1:
        model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model).to(device)
        LOGGER.info('Using SyncBatchNorm()')

    # Trainloader
    train_loader, dataset = create_dataloader(train_path,
                                              imgsz,
                                              batch_size // WORLD_SIZE,
                                              gs,
                                              single_cls,
                                              hyp=hyp,
                                              augment=True,
                                              cache=None if opt.cache == 'val' else opt.cache,
                                              rect=opt.rect,
                                              rank=LOCAL_RANK,
                                              workers=workers,
                                              image_weights=opt.image_weights,
                                              quad=opt.quad,
                                              prefix=colorstr('train: '),
                                              shuffle=True)
    labels = np.concatenate(dataset.labels, 0)
    mlc = int(labels[:, 0].max())  # max label class
    assert mlc < nc, f'Label class {mlc} exceeds nc={nc} in {data}. Possible class labels
are 0-{nc - 1}'

    # Process 0
    if RANK in {-1, 0}:
        val_loader = create_dataloader(val_path,
                                       imgsz,
                                       batch_size // WORLD_SIZE * 2,
                                       gs,
                                       single_cls,
                                       hyp=hyp,
                                       cache=None if noval else opt.cache,
                                       rect=True,
                                       rank=-1,
                                       workers=workers * 2,
                                       pad=0.5,
                                       prefix=colorstr('val: '))[0]

        if not resume:
            if not opt.noautoanchor:
                check_anchors(dataset, model=model, thr=hyp['anchor_t'], imgsz=imgsz)  #
run AutoAnchor
```

```
            model.half().float()  # pre-reduce anchor precision

        callbacks.run('on_pretrain_routine_end', labels, names)

    # DDP mode
    if cuda and RANK != -1:
        model = smart_DDP(model)

    # Model attributes
    nl = de_parallel(model).model[-1].nl  # number of detection layers (to scale hyps)
    hyp['box'] *= 3 / nl  # scale to layers
    hyp['cls'] *= nc / 80 * 3 / nl  # scale to classes and layers
    hyp['obj'] *= (imgsz / 640) ** 2 * 3 / nl  # scale to image size and layers
    hyp['label_smoothing'] = opt.label_smoothing
    model.nc = nc  # attach number of classes to model
    model.hyp = hyp  # attach hyperparameters to model
    model.class_weights = labels_to_class_weights(dataset.labels, nc).to(device) * nc  #
attach class weights
    model.names = names

    # Start training
    t0 = time.time()
    nb = len(train_loader)  # number of batches
    nw = max(round(hyp['warmup_epochs'] * nb), 100)  # number of warmup iterations,
max(3 epochs, 100 iterations)
    # nw = min(nw, (epochs - start_epoch) / 2 * nb)  # limit warmup to < 1/2 of training
    last_opt_step = -1
    maps = np.zeros(nc)  # mAP per class
    results = (0, 0, 0, 0, 0, 0, 0)  # P, R, mAP@.5, mAP@.5-.95, val_loss(box, obj, cls)
    scheduler.last_epoch = start_epoch - 1  # do not move
    scaler = torch.cuda.amp.GradScaler(enabled=amp)
    stopper, stop = EarlyStopping(patience=opt.patience), False
    compute_loss = ComputeLoss(model)  # init loss class
    callbacks.run('on_train_start')
    LOGGER.info(f'Image sizes {imgsz} train, {imgsz} val\n'
                f'Using {train_loader.num_workers * WORLD_SIZE} dataloader workers\n'
                f"Logging results to {colorstr('bold', save_dir)}\n"
                f'Starting training for {epochs} epochs...')
    for epoch in range(start_epoch, epochs):  # epoch ------------------------------------------
----------------------
        callbacks.run('on_train_epoch_start')
        model.train()

        # Update image weights (optional, single-GPU only)
        if opt.image_weights:
            cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc  # class weights
            iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw)  #
image weights
            dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n)  #
rand weighted idx
```

```
    # Update mosaic border (optional)
    # b = int(random.uniform(0.25 * imgsz, 0.75 * imgsz + gs) // gs * gs)
    # dataset.mosaic_border = [b - imgsz, -b]  # height, width borders

    mloss = torch.zeros(3, device=device)  # mean losses
    if RANK != -1:
        train_loader.sampler.set_epoch(epoch)
    pbar = enumerate(train_loader)
    LOGGER.info(('\n' + '%11s' * 7) % ('Epoch', 'GPU_mem', 'box_loss', 'obj_loss',
'cls_loss', 'Instances', 'Size'))
    if RANK in {-1, 0}:
        pbar = tqdm(pbar, total=nb, bar_format='{l_bar}{bar:10}{r_bar}{bar:-10b}')  #
progress bar
    optimizer.zero_grad()
    for i, (imgs, targets, paths, _) in pbar:  # batch ------------------------------------------
----------------
        callbacks.run('on_train_batch_start')
        ni = i + nb * epoch  # number integrated batches (since train start)
        imgs = imgs.to(device, non_blocking=True).float() / 255  # uint8 to float32, 0-
255 to 0.0-1.0

        # Warmup
        if ni <= nw:
            xi = [0, nw]  # x interp
            # compute_loss.gr = np.interp(ni, xi, [0.0, 1.0])  # iou loss ratio (obj_loss =
1.0 or iou)
            accumulate = max(1, np.interp(ni, xi, [1, nbs / batch_size]).round())
            for j, x in enumerate(optimizer.param_groups):
                # bias lr falls from 0.1 to lr0, all other lrs rise from 0.0 to lr0
                x['lr'] = np.interp(ni, xi, [hyp['warmup_bias_lr'] if j == 0 else 0.0,
x['initial_lr'] * lf(epoch)])
                if 'momentum' in x:
                    x['momentum'] = np.interp(ni, xi, [hyp['warmup_momentum'],
hyp['momentum']])

        # Multi-scale
        if opt.multi_scale:
            sz = random.randrange(imgsz * 0.5, imgsz * 1.5 + gs) // gs * gs  # size
            sf = sz / max(imgs.shape[2:])  # scale factor
            if sf != 1:
                ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]]  # new shape
(stretched to gs-multiple)
                imgs = nn.functional.interpolate(imgs, size=ns, mode='bilinear',
align_corners=False)

        # Forward
        with torch.cuda.amp.autocast(amp):
            pred = model(imgs)  # forward
            loss, loss_items = compute_loss(pred, targets.to(device))  # loss scaled by
batch_size
            if RANK != -1:
```

```
                loss *= WORLD_SIZE  # gradient averaged between devices in DDP mode
            if opt.quad:
                loss *= 4.

        # Backward
        scaler.scale(loss).backward()

        # Optimize - https://pytorch.org/docs/master/notes/amp_examples.html
        if ni - last_opt_step >= accumulate:
            scaler.unscale_(optimizer)  # unscale gradients
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=10.0)  # clip
gradients
            scaler.step(optimizer)  # optimizer.step
            scaler.update()
            optimizer.zero_grad()
            if ema:
                ema.update(model)
            last_opt_step = ni

        # Log
        if RANK in {-1, 0}:
            mloss = (mloss * i + loss_items) / (i + 1)  # update mean losses
            mem = f'{torch.cuda.memory_reserved() / 1E9 if torch.cuda.is_available()
else 0:.3g}G'  # (GB)
            pbar.set_description(('%11s' * 2 + '%11.4g' * 5) %
                                 (f'{epoch}/{epochs - 1}', mem, *mloss, targets.shape[0],
imgs.shape[-1]))
            callbacks.run('on_train_batch_end', model, ni, imgs, targets, paths,
list(mloss))
            if callbacks.stop_training:
                return
        # end batch ------------------------------------------------------------------------
-------------

    # Scheduler
    lr = [x['lr'] for x in optimizer.param_groups]  # for loggers
    scheduler.step()

    if RANK in {-1, 0}:
        # mAP
        callbacks.run('on_train_epoch_end', epoch=epoch)
        ema.update_attr(model, include=['yaml', 'nc', 'hyp', 'names', 'stride',
'class_weights'])
        final_epoch = (epoch + 1 == epochs) or stopper.possible_stop
        if not noval or final_epoch:  # Calculate mAP
            results, maps, _ = validate.run(data_dict,
                                            batch_size=batch_size // WORLD_SIZE * 2,
                                            imgsz=imgsz,
                                            half=amp,
                                            model=ema.ema,
                                            single_cls=single_cls,
```

```
                                dataloader=val_loader,
                                save_dir=save_dir,
                                plots=False,
                                callbacks=callbacks,
                                compute_loss=compute_loss)

            # Update best mAP
            fi = fitness(np.array(results).reshape(1, -1))  # weighted combination of [P, R,
mAP@.5, mAP@.5-.95]
            stop = stopper(epoch=epoch, fitness=fi)  # early stop check
            if fi > best_fitness:
                best_fitness = fi
            log_vals = list(mloss) + list(results) + lr
            callbacks.run('on_fit_epoch_end', log_vals, epoch, best_fitness, fi)

            # Save model
            if (not nosave) or (final_epoch and not evolve):  # if save
                ckpt = {
                    'epoch': epoch,
                    'best_fitness': best_fitness,
                    'model': deepcopy(de_parallel(model)).half(),
                    'ema': deepcopy(ema.ema).half(),
                    'updates': ema.updates,
                    'optimizer': optimizer.state_dict(),
                    'opt': vars(opt),
                    'date': datetime.now().isoformat()}

                # Save last, best and delete
                torch.save(ckpt, last)
                if best_fitness == fi:
                    torch.save(ckpt, best)
                if opt.save_period > 0 and epoch % opt.save_period == 0:
                    torch.save(ckpt, w / f'epoch{epoch}.pt')
                del ckpt
                callbacks.run('on_model_save', last, epoch, final_epoch, best_fitness, fi)

        # EarlyStopping
        if RANK != -1:  # if DDP training
            broadcast_list = [stop if RANK == 0 else None]
            dist.broadcast_object_list(broadcast_list, 0)  # broadcast 'stop' to all ranks
            if RANK != 0:
                stop = broadcast_list[0]
        if stop:
            break  # must break all DDP ranks

        # end epoch ----------------------------------------------------------------------------
---------------
    # end training -------------------------------------------------------------------------------
---------------
    if RANK in {-1, 0}:
```

```
        LOGGER.info(f'\n{epoch - start_epoch + 1} epochs completed in {(time.time() -
t0) / 3600:.3f} hours.')
        for f in last, best:
            if f.exists():
                strip_optimizer(f)  # strip optimizers
                if f is best:
                    LOGGER.info(f'\nValidating {f}...')
                    results, _, _ = validate.run(
                        data_dict,
                        batch_size=batch_size // WORLD_SIZE * 2,
                        imgsz=imgsz,
                        model=attempt_load(f, device).half(),
                        iou_thres=0.65 if is_coco else 0.60,  # best pycocotools at iou 0.65
                        single_cls=single_cls,
                        dataloader=val_loader,
                        save_dir=save_dir,
                        save_json=is_coco,
                        verbose=True,
                        plots=plots,
                        callbacks=callbacks,
                        compute_loss=compute_loss)  # val best model with plots
                    if is_coco:
                        callbacks.run('on_fit_epoch_end', list(mloss) + list(results) + lr, epoch,
best_fitness, fi)

        callbacks.run('on_train_end', last, best, epoch, results)

    torch.cuda.empty_cache()
    return results


def parse_opt(known=False):
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial
weights path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml',
help='dataset.yaml path')
    parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-
low.yaml', help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=100, help='total training epochs')
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all
GPUs, -1 for autobatch')
    parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train,
val image size (pixels)')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume
most recent training')
    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument('--noval', action='store_true', help='only validate final epoch')
```

```
    parser.add_argument('--noautoanchor', action='store_true', help='disable
AutoAnchor')
    parser.add_argument('--noplots', action='store_true', help='save no plot files')
    parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve
hyperparameters for x generations')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache', type=str, nargs='?', const='ram', help='image --cache
ram/disk')
    parser.add_argument('--image-weights', action='store_true', help='use weighted image
selection for training')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/-
50%%')
    parser.add_argument('--single-cls', action='store_true', help='train multi-class data as
single-class')
    parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'],
default='SGD', help='optimizer')
    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only
available in DDP mode')
    parser.add_argument('--workers', type=int, default=8, help='max dataloader workers
(per RANK in DDP mode)')
    parser.add_argument('--project', default=ROOT / 'runs/train', help='save to
project/name')
    parser.add_argument('--name', default='exp', help='save to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')
    parser.add_argument('--quad', action='store_true', help='quad dataloader')
    parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label
smoothing epsilon')
    parser.add_argument('--patience', type=int, default=100, help='EarlyStopping
patience (epochs without improvement)')
    parser.add_argument('--freeze', nargs='+', type=int, default=[0], help='Freeze layers:
backbone=10, first3=0 1 2')
    parser.add_argument('--save-period', type=int, default=-1, help='Save checkpoint
every x epochs (disabled if < 1)')
    parser.add_argument('--seed', type=int, default=0, help='Global training seed')
    parser.add_argument('--local_rank', type=int, default=-1, help='Automatic DDP
Multi-GPU argument, do not modify')

    # Logger arguments
    parser.add_argument('--entity', default=None, help='Entity')
    parser.add_argument('--upload_dataset', nargs='?', const=True, default=False,
help='Upload data, "val" option')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box
image logging interval')
    parser.add_argument('--artifact_alias', type=str, default='latest', help='Version of
dataset artifact to use')

    return parser.parse_known_args()[0] if known else parser.parse_args()
```

```
def main(opt, callbacks=Callbacks()):
    # Checks
    if RANK in {-1, 0}:
        print_args(vars(opt))
        check_git_status()
        check_requirements()

    # Resume (from specified or most recent last.pt)
    if opt.resume and not check_comet_resume(opt) and not opt.evolve:
        last = Path(check_file(opt.resume) if isinstance(opt.resume, str) else
get_latest_run())
        opt_yaml = last.parent.parent / 'opt.yaml'  # train options yaml
        opt_data = opt.data  # original dataset
        if opt_yaml.is_file():
            with open(opt_yaml, errors='ignore') as f:
                d = yaml.safe_load(f)
        else:
            d = torch.load(last, map_location='cpu')['opt']
        opt = argparse.Namespace(**d)  # replace
        opt.cfg, opt.weights, opt.resume = '', str(last), True  # reinstate
        if is_url(opt_data):
            opt.data = check_file(opt_data)  # avoid HUB resume auth timeout
    else:
        opt.data, opt.cfg, opt.hyp, opt.weights, opt.project = \
            check_file(opt.data), check_yaml(opt.cfg), check_yaml(opt.hyp),
str(opt.weights), str(opt.project)  # checks
        assert len(opt.cfg) or len(opt.weights), 'either --cfg or --weights must be specified'
        if opt.evolve:
            if opt.project == str(ROOT / 'runs/train'):  # if default project name, rename to
runs/evolve
                opt.project = str(ROOT / 'runs/evolve')
            opt.exist_ok, opt.resume = opt.resume, False  # pass resume to exist_ok and
disable resume
        if opt.name == 'cfg':
            opt.name = Path(opt.cfg).stem  # use model.yaml as name
        opt.save_dir = str(increment_path(Path(opt.project) / opt.name,
exist_ok=opt.exist_ok))

    # DDP mode
    device = select_device(opt.device, batch_size=opt.batch_size)
    if LOCAL_RANK != -1:
        msg = 'is not compatible with YOLOv5 Multi-GPU DDP training'
        assert not opt.image_weights, f'--image-weights {msg}'
        assert not opt.evolve, f'--evolve {msg}'
        assert opt.batch_size != -1, f'AutoBatch with --batch-size -1 {msg}, please pass a
valid --batch-size'
        assert opt.batch_size % WORLD_SIZE == 0, f'--batch-size {opt.batch_size} must
be multiple of WORLD_SIZE'
        assert torch.cuda.device_count() > LOCAL_RANK, 'insufficient CUDA devices
for DDP command'
```

```
        torch.cuda.set_device(LOCAL_RANK)
        device = torch.device('cuda', LOCAL_RANK)
        dist.init_process_group(backend="nccl" if dist.is_nccl_available() else "gloo")

    # Train
    if not opt.evolve:
        train(opt.hyp, opt, device, callbacks)

    # Evolve hyperparameters (optional)
    else:
        # Hyperparameter evolution metadata (mutation scale 0-1, lower_limit,
upper_limit)
        meta = {
            'lr0': (1, 1e-5, 1e-1),  # initial learning rate (SGD=1E-2, Adam=1E-3)
            'lrf': (1, 0.01, 1.0),  # final OneCycleLR learning rate (lr0 * lrf)
            'momentum': (0.3, 0.6, 0.98),  # SGD momentum/Adam beta1
            'weight_decay': (1, 0.0, 0.001),  # optimizer weight decay
            'warmup_epochs': (1, 0.0, 5.0),  # warmup epochs (fractions ok)
            'warmup_momentum': (1, 0.0, 0.95),  # warmup initial momentum
            'warmup_bias_lr': (1, 0.0, 0.2),  # warmup initial bias lr
            'box': (1, 0.02, 0.2),  # box loss gain
            'cls': (1, 0.2, 4.0),  # cls loss gain
            'cls_pw': (1, 0.5, 2.0),  # cls BCELoss positive_weight
            'obj': (1, 0.2, 4.0),  # obj loss gain (scale with pixels)
            'obj_pw': (1, 0.5, 2.0),  # obj BCELoss positive_weight
            'iou_t': (0, 0.1, 0.7),  # IoU training threshold
            'anchor_t': (1, 2.0, 8.0),  # anchor-multiple threshold
            'anchors': (2, 2.0, 10.0),  # anchors per output grid (0 to ignore)
            'fl_gamma': (0, 0.0, 2.0),  # focal loss gamma (efficientDet default gamma=1.5)
            'hsv_h': (1, 0.0, 0.1),  # image HSV-Hue augmentation (fraction)
            'hsv_s': (1, 0.0, 0.9),  # image HSV-Saturation augmentation (fraction)
            'hsv_v': (1, 0.0, 0.9),  # image HSV-Value augmentation (fraction)
            'degrees': (1, 0.0, 45.0),  # image rotation (+/- deg)
            'translate': (1, 0.0, 0.9),  # image translation (+/- fraction)
            'scale': (1, 0.0, 0.9),  # image scale (+/- gain)
            'shear': (1, 0.0, 10.0),  # image shear (+/- deg)
            'perspective': (0, 0.0, 0.001),  # image perspective (+/- fraction), range 0-0.001
            'flipud': (1, 0.0, 1.0),  # image flip up-down (probability)
            'fliplr': (0, 0.0, 1.0),  # image flip left-right (probability)
            'mosaic': (1, 0.0, 1.0),  # image mixup (probability)
            'mixup': (1, 0.0, 1.0),  # image mixup (probability)
            'copy_paste': (1, 0.0, 1.0)}  # segment copy-paste (probability)

        with open(opt.hyp, errors='ignore') as f:
            hyp = yaml.safe_load(f)  # load hyps dict
            if 'anchors' not in hyp:  # anchors commented in hyp.yaml
                hyp['anchors'] = 3
        if opt.noautoanchor:
            del hyp['anchors'], meta['anchors']
        opt.noval, opt.nosave, save_dir = True, True, Path(opt.save_dir)  # only val/save
final epoch
```

```
    # ei = [isinstance(x, (int, float)) for x in hyp.values()]  # evolvable indices
    evolve_yaml, evolve_csv = save_dir / 'hyp_evolve.yaml', save_dir / 'evolve.csv'
    if opt.bucket:
        os.system(f'gsutil cp gs://{opt.bucket}/evolve.csv {evolve_csv}')  # download
evolve.csv if exists

    for _ in range(opt.evolve):  # generations to evolve
        if evolve_csv.exists():  # if evolve.csv exists: select best hyps and mutate
            # Select parent(s)
            parent = 'single'  # parent selection method: 'single' or 'weighted'
            x = np.loadtxt(evolve_csv, ndmin=2, delimiter=',', skiprows=1)
            n = min(5, len(x))  # number of previous results to consider
            x = x[np.argsort(-fitness(x))][:n]  # top n mutations
            w = fitness(x) - fitness(x).min() + 1E-6  # weights (sum > 0)
            if parent == 'single' or len(x) == 1:
                # x = x[random.randint(0, n - 1)]  # random selection
                x = x[random.choices(range(n), weights=w)[0]]  # weighted selection
            elif parent == 'weighted':
                x = (x * w.reshape(n, 1)).sum(0) / w.sum()  # weighted combination

            # Mutate
            mp, s = 0.8, 0.2  # mutation probability, sigma
            npr = np.random
            npr.seed(int(time.time()))
            g = np.array([meta[k][0] for k in hyp.keys()])  # gains 0-1
            ng = len(meta)
            v = np.ones(ng)
            while all(v == 1):  # mutate until a change occurs (prevent duplicates)
                v = (g * (npr.random(ng) < mp) * npr.randn(ng) * npr.random() * s +
1).clip(0.3, 3.0)
            for i, k in enumerate(hyp.keys()):  # plt.hist(v.ravel(), 300)
                hyp[k] = float(x[i + 7] * v[i])  # mutate

        # Constrain to limits
        for k, v in meta.items():
            hyp[k] = max(hyp[k], v[1])  # lower limit
            hyp[k] = min(hyp[k], v[2])  # upper limit
            hyp[k] = round(hyp[k], 5)  # significant digits

        # Train mutation
        results = train(hyp.copy(), opt, device, callbacks)
        callbacks = Callbacks()
        # Write mutation results
        keys = ('metrics/precision', 'metrics/recall', 'metrics/mAP_0.5',
'metrics/mAP_0.5:0.95', 'val/box_loss',
                'val/obj_loss', 'val/cls_loss')
        print_mutation(keys, results, hyp.copy(), save_dir, opt.bucket)

    # Plot results
    plot_evolve(evolve_csv)
    LOGGER.info(f'Hyperparameter evolution finished {opt.evolve} generations\n'
```

```
            f"Results saved to {colorstr('bold', save_dir)}\n"
            f'Usage example: $ python train.py --hyp {evolve_yaml}')


def run(**kwargs):
    # Usage: import train; train.run(data='coco128.yaml', imgsz=320,
weights='yolov5m.pt')
    opt = parse_opt(True)
    for k, v in kwargs.items():
        setattr(opt, k, v)
    main(opt)
    return opt


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

```
#YOLOv5
#detect.py


# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
"""
Run YOLOv5 detection inference on images, videos, directories, globs, YouTube,
webcam, streams, etc.

Usage - sources:
    $ python detect.py --weights yolov5s.pt --source 0                               # webcam
                                                 img.jpg                         # image
                                                 vid.mp4                          # video
                                                 path/                           # directory
                                                 'path/*.jpg'                    # glob
                                                 'https://youtu.be/Zgi9g1ksQHc'  # YouTube
                                                 'rtsp://example.com/media.mp4'  # RTSP, RTMP,
HTTP stream

Usage - formats:
    $ python detect.py --weights yolov5s.pt                 # PyTorch
                                 yolov5s.torchscript        # TorchScript
                                 yolov5s.onnx               # ONNX Runtime or OpenCV DNN with --
dnn
                                 yolov5s_openvino_model     # OpenVINO
                                 yolov5s.engine             # TensorRT
                                 yolov5s.mlmodel            # CoreML (macOS-only)
                                 yolov5s_saved_model        # TensorFlow SavedModel
                                 yolov5s.pb                 # TensorFlow GraphDef
                                 yolov5s.tflite             # TensorFlow Lite
                                 yolov5s_edgetpu.tflite     # TensorFlow Edge TPU
                                 yolov5s_paddle_model       # PaddlePaddle
"""

import argparse
import os
import platform
import sys
from pathlib import Path

import torch

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend
from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages,
LoadScreenshots, LoadStreams
```

```
from utils.general import (LOGGER, Profile, check_file, check_img_size,
check_imshow, check_requirements, colorstr, cv2,
                           increment_path, non_max_suppression, print_args, scale_boxes,
strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, smart_inference_mode


@smart_inference_mode()
def run(
        weights=ROOT / 'yolov5s.pt',  # model path or triton URL
        source=ROOT / 'data/images',  # file/dir/URL/glob/screen/0(webcam)
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        vid_stride=1,  # video frame-rate stride
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
    screenshot = source.lower().startswith('screen')
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  #
make dir
```

```
    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data,
fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Dataloader
    bs = 1  # batch_size
    if webcam:
        view_img = check_imshow(warn=True)
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
        bs = len(dataset)
    elif screenshot:
        dataset = LoadScreenshots(source, img_size=imgsz, stride=stride, auto=pt)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt,
vid_stride=vid_stride)
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt or model.triton else bs, 3, *imgsz))  # warmup
    seen, windows, dt = 0, [], (Profile(), Profile(), Profile())
    for path, im, im0s, vid_cap, s in dataset:
        with dt[0]:
            im = torch.from_numpy(im).to(model.device)
            im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
            im /= 255  # 0 - 255 to 0.0 - 1.0
            if len(im.shape) == 3:
                im = im[None]  # expand for batch dim

        # Inference
        with dt[1]:
            visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize
else False
            pred = model(im, augment=augment, visualize=visualize)

        # NMS
        with dt[2]:
            pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)

        # Second-stage classifier (optional)
        # pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
```

```
            p, im0, frame = path[i], im0s[i].copy(), dataset.count
            s += f'{i}: '
        else:
            p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

        p = Path(p)  # to Path
        save_path = str(save_dir / p.name)  # im.jpg
        txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else
f'_{frame}')  # im.txt
        s += '%gx%g ' % im.shape[2:]  # print string
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
        imc = im0.copy() if save_crop else im0  # for save_crop
        annotator = Annotator(im0, line_width=line_thickness, example=str(names))
        if len(det):
            # Rescale boxes from img_size to im0 size
            det[:, :4] = scale_boxes(im.shape[2:], det[:, :4], im0.shape).round()

            # Print results
            for c in det[:, 5].unique():
                n = (det[:, 5] == c).sum()  # detections per class
                s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

            # Write results
            for *xyxy, conf, cls in reversed(det):
                if save_txt:  # Write to file
                    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                    line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                    with open(f'{txt_path}.txt', 'a') as f:
                        f.write(('%g ' * len(line)).rstrip() % line + '\n')

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]}
{conf:.2f}')
                    annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] /
f'{p.stem}.jpg', BGR=True)

        # Stream results
        im0 = annotator.result()
        if view_img:
            if platform.system() == 'Linux' and p not in windows:
                windows.append(p)
                cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond
```

```
            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4
suffix on results videos
                        vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        LOGGER.info(f"{s}{" if len(det) else '(no detections), '}{dt[1].dt * 1E3:.1f}ms")

    # Print results
    t = tuple(x.t / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per
image at shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}"
if save_txt else "
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt',
help='model path or triton URL')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images',
help='file/dir/URL/glob/screen/0(webcam)')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml',
help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640],
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU
threshold')
```

```
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections
per image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --
save-txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction
boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0,
or --classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box
thickness (pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide
labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision
inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX
inference')
    parser.add_argument('--vid-stride', type=int, default=1, help='video frame-rate stride')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))


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

```
#YOLOv7
#train.py


import argparse
import logging
import math
import os
import random
import time
from copy import deepcopy
from pathlib import Path
from threading import Thread

import numpy as np
import torch.distributed as dist
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.optim.lr_scheduler as lr_scheduler
import torch.utils.data
import yaml
from torch.cuda import amp
from torch.nn.parallel import DistributedDataParallel as DDP
from torch.utils.tensorboard import SummaryWriter
from tqdm import tqdm

import test  # import test.py to get mAP after each epoch
from models.experimental import attempt_load
from models.yolo import Model
from utils.autoanchor import check_anchors
from utils.datasets import create_dataloader
from utils.general import labels_to_class_weights, increment_path,
labels_to_image_weights, init_seeds, \
    fitness, strip_optimizer, get_latest_run, check_dataset, check_file, check_git_status,
check_img_size, \
    check_requirements, print_mutation, set_logging, one_cycle, colorstr
from utils.google_utils import attempt_download
from utils.loss import ComputeLoss, ComputeLossOTA
from utils.plots import plot_images, plot_labels, plot_results, plot_evolution
from utils.torch_utils import ModelEMA, select_device, intersect_dicts,
torch_distributed_zero_first, is_parallel
from utils.wandb_logging.wandb_utils import WandbLogger, check_wandb_resume

logger = logging.getLogger(__name__)


def train(hyp, opt, device, tb_writer=None):
    logger.info(colorstr('hyperparameters: ') + ', '.join(f'{k}={v}' for k, v in hyp.items()))
    save_dir, epochs, batch_size, total_batch_size, weights, rank = \
```

```
        Path(opt.save_dir), opt.epochs, opt.batch_size, opt.total_batch_size, opt.weights,
opt.global_rank

    # Directories
    wdir = save_dir / 'weights'
    wdir.mkdir(parents=True, exist_ok=True)  # make dir
    last = wdir / 'last.pt'
    best = wdir / 'best.pt'
    results_file = save_dir / 'results.txt'

    # Save run settings
    with open(save_dir / 'hyp.yaml', 'w') as f:
        yaml.dump(hyp, f, sort_keys=False)
    with open(save_dir / 'opt.yaml', 'w') as f:
        yaml.dump(vars(opt), f, sort_keys=False)

    # Configure
    plots = not opt.evolve  # create plots
    cuda = device.type != 'cpu'
    init_seeds(2 + rank)
    with open(opt.data) as f:
        data_dict = yaml.load(f, Loader=yaml.SafeLoader)  # data dict
    is_coco = opt.data.endswith('coco.yaml')

    # Logging- Doing this before checking the dataset. Might update data_dict
    loggers = {'wandb': None}  # loggers dict
    if rank in [-1, 0]:
        opt.hyp = hyp  # add hyperparameters
        run_id = torch.load(weights).get('wandb_id') if weights.endswith('.pt') and
os.path.isfile(weights) else None
        wandb_logger = WandbLogger(opt, Path(opt.save_dir).stem, run_id, data_dict)
        loggers['wandb'] = wandb_logger.wandb
        data_dict = wandb_logger.data_dict
        if wandb_logger.wandb:
            weights, epochs, hyp = opt.weights, opt.epochs, opt.hyp  # WandbLogger might
update weights, epochs if resuming

    nc = 1 if opt.single_cls else int(data_dict['nc'])  # number of classes
    names = ['item'] if opt.single_cls and len(data_dict['names']) != 1 else
data_dict['names']  # class names
    assert len(names) == nc, '%g names found for nc=%g dataset in %s' % (len(names),
nc, opt.data)  # check

    # Model
    pretrained = weights.endswith('.pt')
    if pretrained:
        with torch_distributed_zero_first(rank):
            attempt_download(weights)  # download if not found locally
        ckpt = torch.load(weights, map_location=device)  # load checkpoint
        model = Model(opt.cfg or ckpt['model'].yaml, ch=3, nc=nc,
anchors=hyp.get('anchors')).to(device)  # create
```

```
        exclude = ['anchor'] if (opt.cfg or hyp.get('anchors')) and not opt.resume else []  # exclude keys
        state_dict = ckpt['model'].float().state_dict()  # to FP32
        state_dict = intersect_dicts(state_dict, model.state_dict(), exclude=exclude)  # intersect
        model.load_state_dict(state_dict, strict=False)  # load
        logger.info('Transferred %g/%g items from %s' % (len(state_dict), len(model.state_dict()), weights))  # report
    else:
        model = Model(opt.cfg, ch=3, nc=nc, anchors=hyp.get('anchors')).to(device)  # create
    with torch_distributed_zero_first(rank):
        check_dataset(data_dict)  # check
    train_path = data_dict['train']
    test_path = data_dict['val']

    # Freeze
    freeze = []  # parameter names to freeze (full or partial)
    for k, v in model.named_parameters():
        v.requires_grad = True  # train all layers
        if any(x in k for x in freeze):
            print('freezing %s' % k)
            v.requires_grad = False

    # Optimizer
    nbs = 64  # nominal batch size
    accumulate = max(round(nbs / total_batch_size), 1)  # accumulate loss before optimizing
    hyp['weight_decay'] *= total_batch_size * accumulate / nbs  # scale weight_decay
    logger.info(f"Scaled weight_decay = {hyp['weight_decay']}")

    pg0, pg1, pg2 = [], [], []  # optimizer parameter groups
    for k, v in model.named_modules():
        if hasattr(v, 'bias') and isinstance(v.bias, nn.Parameter):
            pg2.append(v.bias)  # biases
        if isinstance(v, nn.BatchNorm2d):
            pg0.append(v.weight)  # no decay
        elif hasattr(v, 'weight') and isinstance(v.weight, nn.Parameter):
            pg1.append(v.weight)  # apply decay
        if hasattr(v, 'im'):
            if hasattr(v.im, 'implicit'):
                pg0.append(v.im.implicit)
            else:
                for iv in v.im:
                    pg0.append(iv.implicit)
        if hasattr(v, 'imc'):
            if hasattr(v.imc, 'implicit'):
                pg0.append(v.imc.implicit)
            else:
                for iv in v.imc:
                    pg0.append(iv.implicit)
```

```
        if hasattr(v, 'imb'):
            if hasattr(v.imb, 'implicit'):
                pg0.append(v.imb.implicit)
            else:
                for iv in v.imb:
                    pg0.append(iv.implicit)
        if hasattr(v, 'imo'):
            if hasattr(v.imo, 'implicit'):
                pg0.append(v.imo.implicit)
            else:
                for iv in v.imo:
                    pg0.append(iv.implicit)
        if hasattr(v, 'ia'):
            if hasattr(v.ia, 'implicit'):
                pg0.append(v.ia.implicit)
            else:
                for iv in v.ia:
                    pg0.append(iv.implicit)
        if hasattr(v, 'attn'):
            if hasattr(v.attn, 'logit_scale'):
                pg0.append(v.attn.logit_scale)
            if hasattr(v.attn, 'q_bias'):
                pg0.append(v.attn.q_bias)
            if hasattr(v.attn, 'v_bias'):
                pg0.append(v.attn.v_bias)
            if hasattr(v.attn, 'relative_position_bias_table'):
                pg0.append(v.attn.relative_position_bias_table)
        if hasattr(v, 'rbr_dense'):
            if hasattr(v.rbr_dense, 'weight_rbr_origin'):
                pg0.append(v.rbr_dense.weight_rbr_origin)
            if hasattr(v.rbr_dense, 'weight_rbr_avg_conv'):
                pg0.append(v.rbr_dense.weight_rbr_avg_conv)
            if hasattr(v.rbr_dense, 'weight_rbr_pfir_conv'):
                pg0.append(v.rbr_dense.weight_rbr_pfir_conv)
            if hasattr(v.rbr_dense, 'weight_rbr_1x1_kxk_idconv1'):
                pg0.append(v.rbr_dense.weight_rbr_1x1_kxk_idconv1)
            if hasattr(v.rbr_dense, 'weight_rbr_1x1_kxk_conv2'):
                pg0.append(v.rbr_dense.weight_rbr_1x1_kxk_conv2)
            if hasattr(v.rbr_dense, 'weight_rbr_gconv_dw'):
                pg0.append(v.rbr_dense.weight_rbr_gconv_dw)
            if hasattr(v.rbr_dense, 'weight_rbr_gconv_pw'):
                pg0.append(v.rbr_dense.weight_rbr_gconv_pw)
            if hasattr(v.rbr_dense, 'vector'):
                pg0.append(v.rbr_dense.vector)

    if opt.adam:
        optimizer = optim.Adam(pg0, lr=hyp['lr0'], betas=(hyp['momentum'], 0.999))  #
adjust beta1 to momentum
    else:
        optimizer = optim.SGD(pg0, lr=hyp['lr0'], momentum=hyp['momentum'],
nesterov=True)
```

```
    optimizer.add_param_group({'params': pg1, 'weight_decay': hyp['weight_decay']})  # 
add pg1 with weight_decay
    optimizer.add_param_group({'params': pg2})  # add pg2 (biases)
    logger.info('Optimizer groups: %g .bias, %g conv.weight, %g other' % (len(pg2), 
len(pg1), len(pg0)))
    del pg0, pg1, pg2

    # Scheduler https://arxiv.org/pdf/1812.01187.pdf
    # 
https://pytorch.org/docs/stable/_modules/torch/optim/lr_scheduler.html#OneCycleLR
    if opt.linear_lr:
        lf = lambda x: (1 - x / (epochs - 1)) * (1.0 - hyp['lrf']) + hyp['lrf']  # linear
    else:
        lf = one_cycle(1, hyp['lrf'], epochs)  # cosine 1->hyp['lrf']
    scheduler = lr_scheduler.LambdaLR(optimizer, lr_lambda=lf)
    # plot_lr_scheduler(optimizer, scheduler, epochs)

    # EMA
    ema = ModelEMA(model) if rank in [-1, 0] else None

    # Resume
    start_epoch, best_fitness = 0, 0.0
    if pretrained:
        # Optimizer
        if ckpt['optimizer'] is not None:
            optimizer.load_state_dict(ckpt['optimizer'])
            best_fitness = ckpt['best_fitness']

        # EMA
        if ema and ckpt.get('ema'):
            ema.ema.load_state_dict(ckpt['ema'].float().state_dict())
            ema.updates = ckpt['updates']

        # Results
        if ckpt.get('training_results') is not None:
            results_file.write_text(ckpt['training_results'])  # write results.txt

        # Epochs
        start_epoch = ckpt['epoch'] + 1
        if opt.resume:
            assert start_epoch > 0, '%s training to %g epochs is finished, nothing to resume.' 
% (weights, epochs)
        if epochs < start_epoch:
            logger.info('%s has been trained for %g epochs. Fine-tuning for %g additional 
epochs.' %
                        (weights, ckpt['epoch'], epochs))
            epochs += ckpt['epoch']  # finetune additional epochs

        del ckpt, state_dict
```

```
    # Image sizes
    gs = max(int(model.stride.max()), 32)  # grid size (max stride)
    nl = model.model[-1].nl  # number of detection layers (used for scaling hyp['obj'])
    imgsz, imgsz_test = [check_img_size(x, gs) for x in opt.img_size]  # verify imgsz are
gs-multiples

    # DP mode
    if cuda and rank == -1 and torch.cuda.device_count() > 1:
        model = torch.nn.DataParallel(model)

    # SyncBatchNorm
    if opt.sync_bn and cuda and rank != -1:
        model = torch.nn.SyncBatchNorm.convert_sync_batchnorm(model).to(device)
        logger.info('Using SyncBatchNorm()')

    # Trainloader
    dataloader, dataset = create_dataloader(train_path, imgsz, batch_size, gs, opt,
                                            hyp=hyp, augment=True, cache=opt.cache_images,
rect=opt.rect, rank=rank,
                                            world_size=opt.world_size, workers=opt.workers,
                                            image_weights=opt.image_weights, quad=opt.quad,
prefix=colorstr('train: '))
    mlc = np.concatenate(dataset.labels, 0)[:, 0].max()  # max label class
    nb = len(dataloader)  # number of batches
    assert mlc < nc, 'Label class %g exceeds nc=%g in %s. Possible class labels are 0-
%g' % (mlc, nc, opt.data, nc - 1)

    # Process 0
    if rank in [-1, 0]:
        testloader = create_dataloader(test_path, imgsz_test, batch_size * 2, gs, opt,  #
testloader
                                       hyp=hyp, cache=opt.cache_images and not opt.notest,
rect=True, rank=-1,
                                       world_size=opt.world_size, workers=opt.workers,
                                       pad=0.5, prefix=colorstr('val: '))[0]

        if not opt.resume:
            labels = np.concatenate(dataset.labels, 0)
            c = torch.tensor(labels[:, 0])  # classes
            # cf = torch.bincount(c.long(), minlength=nc) + 1.  # frequency
            # model._initialize_biases(cf.to(device))
            if plots:
                #plot_labels(labels, names, save_dir, loggers)
                if tb_writer:
                    tb_writer.add_histogram('classes', c, 0)

            # Anchors
            if not opt.noautoanchor:
                check_anchors(dataset, model=model, thr=hyp['anchor_t'], imgsz=imgsz)
            model.half().float()  # pre-reduce anchor precision
```

```
    # DDP mode
    if cuda and rank != -1:
        model = DDP(model, device_ids=[opt.local_rank], output_device=opt.local_rank,
                    # nn.MultiheadAttention incompatibility with DDP
https://github.com/pytorch/pytorch/issues/26698
                    find_unused_parameters=any(isinstance(layer, nn.MultiheadAttention) for
layer in model.modules()))

    # Model parameters
    hyp['box'] *= 3. / nl  # scale to layers
    hyp['cls'] *= nc / 80. * 3. / nl  # scale to classes and layers
    hyp['obj'] *= (imgsz / 640) ** 2 * 3. / nl  # scale to image size and layers
    hyp['label_smoothing'] = opt.label_smoothing
    model.nc = nc  # attach number of classes to model
    model.hyp = hyp  # attach hyperparameters to model
    model.gr = 1.0  # iou loss ratio (obj_loss = 1.0 or iou)
    model.class_weights = labels_to_class_weights(dataset.labels, nc).to(device) * nc  #
attach class weights
    model.names = names

    # Start training
    t0 = time.time()
    nw = max(round(hyp['warmup_epochs'] * nb), 1000)  # number of warmup iterations,
max(3 epochs, 1k iterations)
    # nw = min(nw, (epochs - start_epoch) / 2 * nb)  # limit warmup to < 1/2 of training
    maps = np.zeros(nc)  # mAP per class
    results = (0, 0, 0, 0, 0, 0, 0)  # P, R, mAP@.5, mAP@.5-.95, val_loss(box, obj, cls)
    scheduler.last_epoch = start_epoch - 1  # do not move
    scaler = amp.GradScaler(enabled=cuda)
    compute_loss_ota = ComputeLossOTA(model)  # init loss class
    compute_loss = ComputeLoss(model)  # init loss class
    logger.info(f'Image sizes {imgsz} train, {imgsz_test} test\n'
                f'Using {dataloader.num_workers} dataloader workers\n'
                f'Logging results to {save_dir}\n'
                f'Starting training for {epochs} epochs...')
    torch.save(model, wdir / 'init.pt')
    for epoch in range(start_epoch, epochs):  # epoch ------------------------------------------
----------------------
        model.train()

        # Update image weights (optional)
        if opt.image_weights:
            # Generate indices
            if rank in [-1, 0]:
                cw = model.class_weights.cpu().numpy() * (1 - maps) ** 2 / nc  # class
weights
                iw = labels_to_image_weights(dataset.labels, nc=nc, class_weights=cw)  #
image weights
                dataset.indices = random.choices(range(dataset.n), weights=iw, k=dataset.n)
# rand weighted idx
            # Broadcast if DDP
```

```
        if rank != -1:
            indices = (torch.tensor(dataset.indices) if rank == 0 else
torch.zeros(dataset.n)).int()
            dist.broadcast(indices, 0)
            if rank != 0:
                dataset.indices = indices.cpu().numpy()

    # Update mosaic border
    # b = int(random.uniform(0.25 * imgsz, 0.75 * imgsz + gs) // gs * gs)
    # dataset.mosaic_border = [b - imgsz, -b]  # height, width borders

    mloss = torch.zeros(4, device=device)  # mean losses
    if rank != -1:
        dataloader.sampler.set_epoch(epoch)
    pbar = enumerate(dataloader)
    logger.info(('\n' + '%10s' * 8) % ('Epoch', 'gpu_mem', 'box', 'obj', 'cls', 'total',
'labels', 'img_size'))
    if rank in [-1, 0]:
        pbar = tqdm(pbar, total=nb)  # progress bar
    optimizer.zero_grad()
    for i, (imgs, targets, paths, _) in pbar:  # batch -------------------------------------------
-----------------
        ni = i + nb * epoch  # number integrated batches (since train start)
        imgs = imgs.to(device, non_blocking=True).float() / 255.0  # uint8 to float32, 0-
255 to 0.0-1.0

        # Warmup
        if ni <= nw:
            xi = [0, nw]  # x interp
            # model.gr = np.interp(ni, xi, [0.0, 1.0])  # iou loss ratio (obj_loss = 1.0 or
iou)
            accumulate = max(1, np.interp(ni, xi, [1, nbs / total_batch_size]).round())
            for j, x in enumerate(optimizer.param_groups):
                # bias lr falls from 0.1 to lr0, all other lrs rise from 0.0 to lr0
                x['lr'] = np.interp(ni, xi, [hyp['warmup_bias_lr'] if j == 2 else 0.0,
x['initial_lr'] * lf(epoch)])
                if 'momentum' in x:
                    x['momentum'] = np.interp(ni, xi, [hyp['warmup_momentum'],
hyp['momentum']])

        # Multi-scale
        if opt.multi_scale:
            sz = random.randrange(imgsz * 0.5, imgsz * 1.5 + gs) // gs * gs  # size
            sf = sz / max(imgs.shape[2:])  # scale factor
            if sf != 1:
                ns = [math.ceil(x * sf / gs) * gs for x in imgs.shape[2:]]  # new shape
(stretched to gs-multiple)
                imgs = F.interpolate(imgs, size=ns, mode='bilinear', align_corners=False)

        # Forward
        with amp.autocast(enabled=cuda):
```

```
            pred = model(imgs)  # forward
            loss, loss_items = compute_loss_ota(pred, targets.to(device), imgs)  # loss
scaled by batch_size
            if rank != -1:
                loss *= opt.world_size  # gradient averaged between devices in DDP mode
            if opt.quad:
                loss *= 4.

        # Backward
        scaler.scale(loss).backward()

        # Optimize
        if ni % accumulate == 0:
            scaler.step(optimizer)  # optimizer.step
            scaler.update()
            optimizer.zero_grad()
            if ema:
                ema.update(model)

        # Print
        if rank in [-1, 0]:
            mloss = (mloss * i + loss_items) / (i + 1)  # update mean losses
            mem = '%.3gG' % (torch.cuda.memory_reserved() / 1E9 if
torch.cuda.is_available() else 0)  # (GB)
            s = ('%10s' * 2 + '%10.4g' * 6) % (
                '%g/%g' % (epoch, epochs - 1), mem, *mloss, targets.shape[0],
imgs.shape[-1])
            pbar.set_description(s)

            # Plot
            if plots and ni < 10:
                f = save_dir / f'train_batch{ni}.jpg'  # filename
                Thread(target=plot_images, args=(imgs, targets, paths, f),
daemon=True).start()
                # if tb_writer:
                #     tb_writer.add_image(f, result, dataformats='HWC', global_step=epoch)
                #     tb_writer.add_graph(torch.jit.trace(model, imgs, strict=False), [])  #
add model graph
            elif plots and ni == 10 and wandb_logger.wandb:
                wandb_logger.log({"Mosaics": [wandb_logger.wandb.Image(str(x),
caption=x.name) for x in
                                              save_dir.glob('train*.jpg') if x.exists()]})

        # end batch ------------------------------------------------------------------------
-------------
    # end epoch ------------------------------------------------------------------------
--------------

    # Scheduler
    lr = [x['lr'] for x in optimizer.param_groups]  # for tensorboard
    scheduler.step()
```

```
        # DDP process 0 or single-GPU
        if rank in [-1, 0]:
            # mAP
            ema.update_attr(model, include=['yaml', 'nc', 'hyp', 'gr', 'names', 'stride',
'class_weights'])
            final_epoch = epoch + 1 == epochs
            if not opt.notest or final_epoch:  # Calculate mAP
                wandb_logger.current_epoch = epoch + 1
                results, maps, times = test.test(data_dict,
                                                 batch_size=batch_size * 2,
                                                 imgsz=imgsz_test,
                                                 model=ema.ema,
                                                 single_cls=opt.single_cls,
                                                 dataloader=testloader,
                                                 save_dir=save_dir,
                                                 verbose=nc < 50 and final_epoch,
                                                 plots=plots and final_epoch,
                                                 wandb_logger=wandb_logger,
                                                 compute_loss=compute_loss,
                                                 is_coco=is_coco)

            # Write
            with open(results_file, 'a') as f:
                f.write(s + '%10.4g' * 7 % results + '\n')  # append metrics, val_loss
            if len(opt.name) and opt.bucket:
                os.system('gsutil cp %s gs://%s/results/results%s.txt' % (results_file,
opt.bucket, opt.name))

            # Log
            tags = ['train/box_loss', 'train/obj_loss', 'train/cls_loss',  # train loss
                    'metrics/precision', 'metrics/recall', 'metrics/mAP_0.5',
'metrics/mAP_0.5:0.95',
                    'val/box_loss', 'val/obj_loss', 'val/cls_loss',  # val loss
                    'x/lr0', 'x/lr1', 'x/lr2']  # params
            for x, tag in zip(list(mloss[:-1]) + list(results) + lr, tags):
                if tb_writer:
                    tb_writer.add_scalar(tag, x, epoch)  # tensorboard
                if wandb_logger.wandb:
                    wandb_logger.log({tag: x})  # W&B

            # Update best mAP
            fi = fitness(np.array(results).reshape(1, -1))  # weighted combination of [P, R,
mAP@.5, mAP@.5-.95]
            if fi > best_fitness:
                best_fitness = fi
            wandb_logger.end_epoch(best_result=best_fitness == fi)

            # Save model
            if (not opt.nosave) or (final_epoch and not opt.evolve):  # if save
                ckpt = {'epoch': epoch,
```

```
                'best_fitness': best_fitness,
                'training_results': results_file.read_text(),
                'model': deepcopy(model.module if is_parallel(model) else model).half(),
                'ema': deepcopy(ema.ema).half(),
                'updates': ema.updates,
                'optimizer': optimizer.state_dict(),
                'wandb_id': wandb_logger.wandb_run.id if wandb_logger.wandb else
None}

            # Save last, best and delete
            torch.save(ckpt, last)
            if best_fitness == fi:
                torch.save(ckpt, best)
            if (best_fitness == fi) and (epoch >= 200):
                torch.save(ckpt, wdir / 'best_{:03d}.pt'.format(epoch))
            if epoch == 0:
                torch.save(ckpt, wdir / 'epoch_{:03d}.pt'.format(epoch))
            elif ((epoch+1) % 25) == 0:
                torch.save(ckpt, wdir / 'epoch_{:03d}.pt'.format(epoch))
            elif epoch >= (epochs-5):
                torch.save(ckpt, wdir / 'epoch_{:03d}.pt'.format(epoch))
            if wandb_logger.wandb:
                if ((epoch + 1) % opt.save_period == 0 and not final_epoch) and
opt.save_period != -1:
                    wandb_logger.log_model(
                        last.parent, opt, epoch, fi, best_model=best_fitness == fi)
            del ckpt

        # end epoch ----------------------------------------------------------------------------------------
---------------
    # end training
    if rank in [-1, 0]:
        # Plots
        if plots:
            plot_results(save_dir=save_dir)  # save as results.png
            if wandb_logger.wandb:
                files = ['results.png', 'confusion_matrix.png', *[f'{x}_curve.png' for x in ('F1',
'PR', 'P', 'R')]]
                wandb_logger.log({"Results": [wandb_logger.wandb.Image(str(save_dir / f),
caption=f) for f in files
                                              if (save_dir / f).exists()]})
        # Test best.pt
        logger.info('%g epochs completed in %.3f hours.\n' % (epoch - start_epoch + 1,
(time.time() - t0) / 3600))
        if opt.data.endswith('coco.yaml') and nc == 80:  # if COCO
            for m in (last, best) if best.exists() else (last):  # speed, mAP tests
                results, _, _ = test.test(opt.data,
                                          batch_size=batch_size * 2,
                                          imgsz=imgsz_test,
                                          conf_thres=0.001,
                                          iou_thres=0.7,
```

```
                                model=attempt_load(m, device).half(),
                                single_cls=opt.single_cls,
                                dataloader=testloader,
                                save_dir=save_dir,
                                save_json=True,
                                plots=False,
                                is_coco=is_coco)

        # Strip optimizers
        final = best if best.exists() else last  # final model
        for f in last, best:
            if f.exists():
                strip_optimizer(f)  # strip optimizers
        if opt.bucket:
            os.system(f'gsutil cp {final} gs://{opt.bucket}/weights')  # upload
        if wandb_logger.wandb and not opt.evolve:  # Log the stripped model
            wandb_logger.wandb.log_artifact(str(final), type='model',
                                            name='run_' + wandb_logger.wandb_run.id + '_model',
                                            aliases=['last', 'best', 'stripped'])
        wandb_logger.finish_run()
    else:
        dist.destroy_process_group()
    torch.cuda.empty_cache()
    return results


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', type=str, default='yolo7.pt', help='initial weights
path')
    parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
    parser.add_argument('--data', type=str, default='data/coco.yaml', help='data.yaml
path')
    parser.add_argument('--hyp', type=str, default='data/hyp.scratch.p5.yaml',
help='hyperparameters path')
    parser.add_argument('--epochs', type=int, default=300)
    parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all
GPUs')
    parser.add_argument('--img-size', nargs='+', type=int, default=[640, 640],
help='[train, test] image sizes')
    parser.add_argument('--rect', action='store_true', help='rectangular training')
    parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume
most recent training')
    parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
    parser.add_argument('--notest', action='store_true', help='only test final epoch')
    parser.add_argument('--noautoanchor', action='store_true', help='disable autoanchor
check')
    parser.add_argument('--evolve', action='store_true', help='evolve hyperparameters')
    parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
    parser.add_argument('--cache-images', action='store_true', help='cache images for
faster training')
```

```
    parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
    parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
    parser.add_argument('--adam', action='store_true', help='use torch.optim.Adam() optimizer')
    parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
    parser.add_argument('--local_rank', type=int, default=-1, help='DDP parameter, do not modify')
    parser.add_argument('--workers', type=int, default=8, help='maximum number of dataloader workers')
    parser.add_argument('--project', default='runs/train', help='save to project/name')
    parser.add_argument('--entity', default=None, help='W&B entity')
    parser.add_argument('--name', default='exp', help='save to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
    parser.add_argument('--quad', action='store_true', help='quad dataloader')
    parser.add_argument('--linear-lr', action='store_true', help='linear LR')
    parser.add_argument('--label-smoothing', type=float, default=0.0, help='Label smoothing epsilon')
    parser.add_argument('--upload_dataset', action='store_true', help='Upload dataset as W&B artifact table')
    parser.add_argument('--bbox_interval', type=int, default=-1, help='Set bounding-box image logging interval for W&B')
    parser.add_argument('--save_period', type=int, default=-1, help='Log model after every "save_period" epoch')
    parser.add_argument('--artifact_alias', type=str, default="latest", help='version of dataset artifact to be used')
    opt = parser.parse_args()

    # Set DDP variables
    opt.world_size = int(os.environ['WORLD_SIZE']) if 'WORLD_SIZE' in os.environ else 1
    opt.global_rank = int(os.environ['RANK']) if 'RANK' in os.environ else -1
    set_logging(opt.global_rank)
    #if opt.global_rank in [-1, 0]:
    #    check_git_status()
    #    check_requirements()

    # Resume
    wandb_run = check_wandb_resume(opt)
    if opt.resume and not wandb_run:  # resume an interrupted run
        ckpt = opt.resume if isinstance(opt.resume, str) else get_latest_run()  # specified or most recent path
        assert os.path.isfile(ckpt), 'ERROR: --resume checkpoint does not exist'
        apriori = opt.global_rank, opt.local_rank
        with open(Path(ckpt).parent.parent / 'opt.yaml') as f:
```

```
            opt = argparse.Namespace(**yaml.load(f, Loader=yaml.SafeLoader))  # replace
        opt.cfg, opt.weights, opt.resume, opt.batch_size, opt.global_rank, opt.local_rank =
'', ckpt, True, opt.total_batch_size, *apriori  # reinstate
        logger.info('Resuming training from %s' % ckpt)
    else:
        # opt.hyp = opt.hyp or ('hyp.finetune.yaml' if opt.weights else 'hyp.scratch.yaml')
        opt.data, opt.cfg, opt.hyp = check_file(opt.data), check_file(opt.cfg),
check_file(opt.hyp)  # check files
        assert len(opt.cfg) or len(opt.weights), 'either --cfg or --weights must be specified'
        opt.img_size.extend([opt.img_size[-1]] * (2 - len(opt.img_size)))  # extend to 2
sizes (train, test)
        opt.name = 'evolve' if opt.evolve else opt.name
        opt.save_dir = increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok
| opt.evolve)  # increment run

    # DDP mode
    opt.total_batch_size = opt.batch_size
    device = select_device(opt.device, batch_size=opt.batch_size)
    if opt.local_rank != -1:
        assert torch.cuda.device_count() > opt.local_rank
        torch.cuda.set_device(opt.local_rank)
        device = torch.device('cuda', opt.local_rank)
        dist.init_process_group(backend='nccl', init_method='env://')  # distributed
backend
        assert opt.batch_size % opt.world_size == 0, '--batch-size must be multiple of
CUDA device count'
        opt.batch_size = opt.total_batch_size // opt.world_size

    # Hyperparameters
    with open(opt.hyp) as f:
        hyp = yaml.load(f, Loader=yaml.SafeLoader)  # load hyps

    # Train
    logger.info(opt)
    if not opt.evolve:
        tb_writer = None  # init loggers
        if opt.global_rank in [-1, 0]:
            prefix = colorstr('tensorboard: ')
            logger.info(f"{prefix}Start with 'tensorboard --logdir {opt.project}', view at
http://localhost:6006/")
            tb_writer = SummaryWriter(opt.save_dir)  # Tensorboard
        train(hyp, opt, device, tb_writer)

    # Evolve hyperparameters (optional)
    else:
        # Hyperparameter evolution metadata (mutation scale 0-1, lower_limit,
upper_limit)
        meta = {'lr0': (1, 1e-5, 1e-1),  # initial learning rate (SGD=1E-2, Adam=1E-3)
                'lrf': (1, 0.01, 1.0),  # final OneCycleLR learning rate (lr0 * lrf)
                'momentum': (0.3, 0.6, 0.98),  # SGD momentum/Adam beta1
                'weight_decay': (1, 0.0, 0.001),  # optimizer weight decay
```

```
            'warmup_epochs': (1, 0.0, 5.0),  # warmup epochs (fractions ok)
            'warmup_momentum': (1, 0.0, 0.95),  # warmup initial momentum
            'warmup_bias_lr': (1, 0.0, 0.2),  # warmup initial bias lr
            'box': (1, 0.02, 0.2),  # box loss gain
            'cls': (1, 0.2, 4.0),  # cls loss gain
            'cls_pw': (1, 0.5, 2.0),  # cls BCELoss positive_weight
            'obj': (1, 0.2, 4.0),  # obj loss gain (scale with pixels)
            'obj_pw': (1, 0.5, 2.0),  # obj BCELoss positive_weight
            'iou_t': (0, 0.1, 0.7),  # IoU training threshold
            'anchor_t': (1, 2.0, 8.0),  # anchor-multiple threshold
            'anchors': (2, 2.0, 10.0),  # anchors per output grid (0 to ignore)
            'fl_gamma': (0, 0.0, 2.0),  # focal loss gamma (efficientDet default
gamma=1.5)
            'hsv_h': (1, 0.0, 0.1),  # image HSV-Hue augmentation (fraction)
            'hsv_s': (1, 0.0, 0.9),  # image HSV-Saturation augmentation (fraction)
            'hsv_v': (1, 0.0, 0.9),  # image HSV-Value augmentation (fraction)
            'degrees': (1, 0.0, 45.0),  # image rotation (+/- deg)
            'translate': (1, 0.0, 0.9),  # image translation (+/- fraction)
            'scale': (1, 0.0, 0.9),  # image scale (+/- gain)
            'shear': (1, 0.0, 10.0),  # image shear (+/- deg)
            'perspective': (0, 0.0, 0.001),  # image perspective (+/- fraction), range 0-0.001
            'flipud': (1, 0.0, 1.0),  # image flip up-down (probability)
            'fliplr': (0, 0.0, 1.0),  # image flip left-right (probability)
            'mosaic': (1, 0.0, 1.0),  # image mixup (probability)
            'mixup': (1, 0.0, 1.0)}  # image mixup (probability)

        assert opt.local_rank == -1, 'DDP mode not implemented for --evolve'
        opt.notest, opt.nosave = True, True  # only test/save final epoch
        # ei = [isinstance(x, (int, float)) for x in hyp.values()]  # evolvable indices
        yaml_file = Path(opt.save_dir) / 'hyp_evolved.yaml'  # save best result here
        if opt.bucket:
            os.system('gsutil cp gs://%s/evolve.txt .' % opt.bucket)  # download evolve.txt if
exists

        for _ in range(300):  # generations to evolve
            if Path('evolve.txt').exists():  # if evolve.txt exists: select best hyps and mutate
                # Select parent(s)
                parent = 'single'  # parent selection method: 'single' or 'weighted'
                x = np.loadtxt('evolve.txt', ndmin=2)
                n = min(5, len(x))  # number of previous results to consider
                x = x[np.argsort(-fitness(x))][:n]  # top n mutations
                w = fitness(x) - fitness(x).min()  # weights
                if parent == 'single' or len(x) == 1:
                    # x = x[random.randint(0, n - 1)]  # random selection
                    x = x[random.choices(range(n), weights=w)[0]]  # weighted selection
                elif parent == 'weighted':
                    x = (x * w.reshape(n, 1)).sum(0) / w.sum()  # weighted combination

                # Mutate
                mp, s = 0.8, 0.2  # mutation probability, sigma
                npr = np.random
```

```
            npr.seed(int(time.time()))
            g = np.array([x[0] for x in meta.values()])  # gains 0-1
            ng = len(meta)
            v = np.ones(ng)
            while all(v == 1):  # mutate until a change occurs (prevent duplicates)
                v = (g * (npr.random(ng) < mp) * npr.randn(ng) * npr.random() * s +
1).clip(0.3, 3.0)
            for i, k in enumerate(hyp.keys()):  # plt.hist(v.ravel(), 300)
                hyp[k] = float(x[i + 7] * v[i])  # mutate

        # Constrain to limits
        for k, v in meta.items():
            hyp[k] = max(hyp[k], v[1])  # lower limit
            hyp[k] = min(hyp[k], v[2])  # upper limit
            hyp[k] = round(hyp[k], 5)  # significant digits

        # Train mutation
        results = train(hyp.copy(), opt, device)

        # Write mutation results
        print_mutation(hyp.copy(), results, yaml_file, opt.bucket)

    # Plot results
    plot_evolution(yaml_file)
    print(f'Hyperparameter evolution complete. Best results saved as: {yaml_file}\n'
          f'Command to train a new model with these hyperparameters: $ python train.py
--hyp {yaml_file}')
```

```
#YOLOv7
#detect.py


import argparse
import time
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, check_imshow,
non_max_suppression, apply_classifier, \
    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized,
TracedModel


def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source, opt.weights,
opt.view_img, opt.save_txt, opt.img_size, not opt.no_trace
    save_img = not opt.nosave and not source.endswith('.txt')  # save inference images
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))

    # Directories
    save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))
# increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  #
make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size

    if trace:
        model = TracedModel(model, device, opt.img_size)

    if half:
        model.half()  # to FP16
```

```
    # Second-stage classifier
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101', n=2)  # initialize
        modelc.load_state_dict(torch.load('weights/resnet101.pt',
map_location=device)['model']).to(device).eval()

    # Set Dataloader
    vid_path, vid_writer = None, None
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride)

    # Get names and colors
    names = model.module.names if hasattr(model, 'module') else model.names
    colors = [[random.randint(0, 255) for _ in range(3)] for _ in names]

    # Run inference
    if device.type != 'cpu':
        model(torch.zeros(1, 3, imgsz,
imgsz).to(device).type_as(next(model.parameters())))  # run once
    old_img_w = old_img_h = imgsz
    old_img_b = 1

    t0 = time.time()
    for path, img, im0s, vid_cap in dataset:
        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Warmup
        if device.type != 'cpu' and (old_img_b != img.shape[0] or old_img_h !=
img.shape[2] or old_img_w != img.shape[3]):
            old_img_b = img.shape[0]
            old_img_h = img.shape[2]
            old_img_w = img.shape[3]
            for i in range(3):
                model(img, augment=opt.augment)[0]

        # Inference
        t1 = time_synchronized()
        with torch.no_grad():   # Calculating gradients would cause a GPU memory leak
            pred = model(img, augment=opt.augment)[0]
        t2 = time_synchronized()
```

```
        # Apply NMS
        pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres,
classes=opt.classes, agnostic=opt.agnostic_nms)
        t3 = time_synchronized()

        # Apply Classifier
        if classify:
            pred = apply_classifier(pred, modelc, img, im0s)

        # Process detections
        for i, det in enumerate(pred):  # detections per image
            if webcam:  # batch_size >= 1
                p, s, im0, frame = path[i], '%g: ' % i, im0s[i].copy(), dataset.count
            else:
                p, s, im0, frame = path, '', im0s, getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # img.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else
f'_{frame}')  # img.txt
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if opt.save_conf else (cls, *xywh)  # label
format
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or view_img:  # Add bbox to image
                        label = f'{names[int(cls)]} {conf:.2f}'
                        plot_one_box(xyxy, im0, label=label, color=colors[int(cls)],
line_thickness=3)

            # Print time (inference + NMS)
            print(f'{s}Done. ({(1E3 * (t2 - t1)):.1f}ms) Inference, ({(1E3 * (t3 - t2)):.1f}ms)
NMS')

            # Stream results
            if view_img:
```

```
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

        # Save results (image with detections)
        if save_img:
            if dataset.mode == 'image':
                cv2.imwrite(save_path, im0)
                print(f" The image with the result is saved in: {save_path}")
            else:  # 'video' or 'stream'
                if vid_path != save_path:  # new video
                    vid_path = save_path
                    if isinstance(vid_writer, cv2.VideoWriter):
                        vid_writer.release()  # release previous video writer
                    if vid_cap:  # video
                        fps = vid_cap.get(cv2.CAP_PROP_FPS)
                        w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                        h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                    else:  # stream
                        fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path += '.mp4'
                    vid_writer = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                vid_writer.write(im0)

    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}"
if save_txt else ''
        #print(f"Results saved to {save_dir}{s}")

    print(f'Done. ({time.time() - t0:.3f}s)')


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default='yolov7.pt',
help='model.pt path(s)')
    parser.add_argument('--source', type=str, default='inference/images', help='source')  #
file/folder, 0 for webcam
    parser.add_argument('--img-size', type=int, default=640, help='inference size
(pixels)')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for
NMS')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='display results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --
save-txt labels')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
```

```
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --class 0, or
--class 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default='runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok,
do not increment')
    parser.add_argument('--no-trace', action='store_true', help='don`t trace model')
    opt = parser.parse_args()
    print(opt)
    #check_requirements(exclude=('pycocotools', 'thop'))

    with torch.no_grad():
        if opt.update:  # update all models (to fix SourceChangeWarning)
            for opt.weights in ['yolov7.pt']:
                detect()
                strip_optimizer(opt.weights)
        else:
            detect()
```

#YOLOv8

#train.py

```
# Ultralytics YOLO 🚀, GPL-3.0 license

from copy import copy

import hydra
import torch
import torch.nn as nn

from ultralytics.nn.tasks import DetectionModel
from ultralytics.yolo import v8
from ultralytics.yolo.data import build_dataloader
from ultralytics.yolo.data.dataloaders.v5loader import create_dataloader
from ultralytics.yolo.engine.trainer import BaseTrainer
from ultralytics.yolo.utils import DEFAULT_CONFIG, colorstr
from ultralytics.yolo.utils.loss import BboxLoss
from ultralytics.yolo.utils.ops import xywh2xyxy
from ultralytics.yolo.utils.plotting import plot_images, plot_results
from ultralytics.yolo.utils.tal import TaskAlignedAssigner, dist2bbox, make_anchors
from ultralytics.yolo.utils.torch_utils import de_parallel


# BaseTrainer python usage
class DetectionTrainer(BaseTrainer):

    def get_dataloader(self, dataset_path, batch_size, mode="train", rank=0):
        # TODO: manage splits differently
        # calculate stride - check if model is initialized
        gs = max(int(de_parallel(self.model).stride.max() if self.model else 0), 32)
        return create_dataloader(path=dataset_path,
                                 imgsz=self.args.imgsz,
                                 batch_size=batch_size,
                                 stride=gs,
                                 hyp=dict(self.args),
                                 augment=mode == "train",
                                 cache=self.args.cache,
                                 pad=0 if mode == "train" else 0.5,
                                 rect=self.args.rect,
                                 rank=rank,
                                 workers=self.args.workers,
                                 close_mosaic=self.args.close_mosaic != 0,
                                 prefix=colorstr(f'{mode}: '),
                                 shuffle=mode == "train",
                                 seed=self.args.seed)[0] if self.args.v5loader else \
            build_dataloader(self.args, batch_size, img_path=dataset_path, stride=gs,
rank=rank, mode=mode)[0]

    def preprocess_batch(self, batch):
```

```
        batch["img"] = batch["img"].to(self.device, non_blocking=True).float() / 255
        return batch

    def set_model_attributes(self):
        nl = de_parallel(self.model).model[-1].nl  # number of detection layers (to scale
hyps)
        self.args.box *= 3 / nl  # scale to layers
        # self.args.cls *= self.data["nc"] / 80 * 3 / nl  # scale to classes and layers
        self.args.cls *= (self.args.imgsz / 640) ** 2 * 3 / nl  # scale to image size and layers
        self.model.nc = self.data["nc"]  # attach number of classes to model
        self.model.args = self.args  # attach hyperparameters to model
        # TODO: self.model.class_weights = labels_to_class_weights(dataset.labels,
nc).to(device) * nc
        self.model.names = self.data["names"]

    def get_model(self, cfg=None, weights=None, verbose=True):
        model = DetectionModel(cfg, ch=3, nc=self.data["nc"], verbose=verbose)
        if weights:
            model.load(weights)

        return model

    def get_validator(self):
        self.loss_names = 'box_loss', 'cls_loss', 'dfl_loss'
        return v8.detect.DetectionValidator(self.test_loader,
                                            save_dir=self.save_dir,
                                            logger=self.console,
                                            args=copy(self.args))

    def criterion(self, preds, batch):
        if not hasattr(self, 'compute_loss'):
            self.compute_loss = Loss(de_parallel(self.model))
        return self.compute_loss(preds, batch)

    def label_loss_items(self, loss_items=None, prefix="train"):
        """
        Returns a loss dict with labelled training loss items tensor
        """
        # Not needed for classification but necessary for segmentation & detection
        keys = [f"{prefix}/{x}" for x in self.loss_names]
        if loss_items is not None:
            loss_items = [round(float(x), 5) for x in loss_items]  # convert tensors to 5
decimal place floats
            return dict(zip(keys, loss_items))
        else:
            return keys

    def progress_string(self):
        return ('\n' + '%11s' *
                (4 + len(self.loss_names))) % ('Epoch', 'GPU_mem', *self.loss_names,
'Instances', 'Size')
```

```
    def plot_training_samples(self, batch, ni):
        plot_images(images=batch["img"],
                    batch_idx=batch["batch_idx"],
                    cls=batch["cls"].squeeze(-1),
                    bboxes=batch["bboxes"],
                    paths=batch["im_file"],
                    fname=self.save_dir / f"train_batch{ni}.jpg")

    def plot_metrics(self):
        plot_results(file=self.csv)  # save results.png


# Criterion class for computing training losses
class Loss:

    def __init__(self, model):  # model must be de-paralleled

        device = next(model.parameters()).device  # get model device
        h = model.args  # hyperparameters

        m = model.model[-1]  # Detect() module
        self.bce = nn.BCEWithLogitsLoss(reduction='none')
        self.hyp = h
        self.stride = m.stride  # model strides
        self.nc = m.nc  # number of classes
        self.no = m.no
        self.reg_max = m.reg_max
        self.device = device

        self.use_dfl = m.reg_max > 1
        self.assigner = TaskAlignedAssigner(topk=10, num_classes=self.nc, alpha=0.5,
beta=6.0)
        self.bbox_loss = BboxLoss(m.reg_max - 1, use_dfl=self.use_dfl).to(device)
        self.proj = torch.arange(m.reg_max, dtype=torch.float, device=device)

    def preprocess(self, targets, batch_size, scale_tensor):
        if targets.shape[0] == 0:
            out = torch.zeros(batch_size, 0, 5, device=self.device)
        else:
            i = targets[:, 0]  # image index
            _, counts = i.unique(return_counts=True)
            out = torch.zeros(batch_size, counts.max(), 5, device=self.device)
            for j in range(batch_size):
                matches = i == j
                n = matches.sum()
                if n:
                    out[j, :n] = targets[matches, 1:]
            out[..., 1:5] = xywh2xyxy(out[..., 1:5].mul_(scale_tensor))
        return out
```

```
    def bbox_decode(self, anchor_points, pred_dist):
        if self.use_dfl:
            b, a, c = pred_dist.shape  # batch, anchors, channels
            pred_dist = pred_dist.view(b, a, 4, c //
4).softmax(3).matmul(self.proj.type(pred_dist.dtype))
            # pred_dist = pred_dist.view(b, a, c // 4,
4).transpose(2,3).softmax(3).matmul(self.proj.type(pred_dist.dtype))
            # pred_dist = (pred_dist.view(b, a, c // 4, 4).softmax(2) *
self.proj.type(pred_dist.dtype).view(1, 1, -1, 1)).sum(2)
        return dist2bbox(pred_dist, anchor_points, xywh=False)

    def __call__(self, preds, batch):
        loss = torch.zeros(3, device=self.device)  # box, cls, dfl
        feats = preds[1] if isinstance(preds, tuple) else preds
        pred_distri, pred_scores = torch.cat([xi.view(feats[0].shape[0], self.no, -1) for xi in
feats], 2).split(
            (self.reg_max * 4, self.nc), 1)

        pred_scores = pred_scores.permute(0, 2, 1).contiguous()
        pred_distri = pred_distri.permute(0, 2, 1).contiguous()

        dtype = pred_scores.dtype
        batch_size = pred_scores.shape[0]
        imgsz = torch.tensor(feats[0].shape[2:], device=self.device, dtype=dtype) *
self.stride[0]  # image size (h,w)
        anchor_points, stride_tensor = make_anchors(feats, self.stride, 0.5)

        # targets
        targets = torch.cat((batch["batch_idx"].view(-1, 1), batch["cls"].view(-1, 1),
batch["bboxes"]), 1)
        targets = self.preprocess(targets.to(self.device), batch_size, scale_tensor=imgsz[[1,
0, 1, 0]])
        gt_labels, gt_bboxes = targets.split((1, 4), 2)  # cls, xyxy
        mask_gt = gt_bboxes.sum(2, keepdim=True).gt_(0)

        # pboxes
        pred_bboxes = self.bbox_decode(anchor_points, pred_distri)  # xyxy, (b, h*w, 4)

        _, target_bboxes, target_scores, fg_mask, _ = self.assigner(
            pred_scores.detach().sigmoid(), (pred_bboxes.detach() *
stride_tensor).type(gt_bboxes.dtype),
            anchor_points * stride_tensor, gt_labels, gt_bboxes, mask_gt)

        target_bboxes /= stride_tensor
        target_scores_sum = target_scores.sum()

        # cls loss
        # loss[1] = self.varifocal_loss(pred_scores, target_scores, target_labels) /
target_scores_sum  # VFL way
        loss[1] = self.bce(pred_scores, target_scores.to(dtype)).sum() / target_scores_sum
# BCE
```

```
        # bbox loss
        if fg_mask.sum():
            loss[0], loss[2] = self.bbox_loss(pred_distri, pred_bboxes, anchor_points,
target_bboxes, target_scores,
                                              target_scores_sum, fg_mask)

        loss[0] *= self.hyp.box  # box gain
        loss[1] *= self.hyp.cls  # cls gain
        loss[2] *= self.hyp.dfl  # dfl gain

        return loss.sum() * batch_size, loss.detach()  # loss(box, cls, dfl)


@hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent),
config_name=DEFAULT_CONFIG.name)
def train(cfg):
    cfg.model = cfg.model or "yolov8n.yaml"
    cfg.data = cfg.data or "coco128.yaml"  # or yolo.ClassificationDataset("mnist")
    cfg.device = cfg.device if cfg.device is not None else ''
    # trainer = DetectionTrainer(cfg)
    # trainer.train()
    from ultralytics import YOLO
    model = YOLO(cfg.model)
    model.train(**cfg)


if __name__ == "__main__":
    """
    CLI usage:
    python ultralytics/yolo/v8/detect/train.py model=yolov8n.yaml data=coco128
epochs=100 imgsz=640

    TODO:
    yolo task=detect mode=train model=yolov8n.yaml data=coco128.yaml epochs=100
    """
    train()
```

```
#YOLOv8
#predict.py


# Ultralytics YOLO 🚀, GPL-3.0 license

import hydra
import torch

from ultralytics.yolo.engine.predictor import BasePredictor
from ultralytics.yolo.utils import DEFAULT_CONFIG, ROOT, ops
from ultralytics.yolo.utils.checks import check_imgsz
from ultralytics.yolo.utils.plotting import Annotator, colors, save_one_box


class DetectionPredictor(BasePredictor):

    def get_annotator(self, img):
        return Annotator(img, line_width=self.args.line_thickness,
example=str(self.model.names))

    def preprocess(self, img):
        img = torch.from_numpy(img).to(self.model.device)
        img = img.half() if self.model.fp16 else img.float()  # uint8 to fp16/32
        img /= 255  # 0 - 255 to 0.0 - 1.0
        return img

    def postprocess(self, preds, img, orig_img):
        preds = ops.non_max_suppression(preds,
                                        self.args.conf,
                                        self.args.iou,
                                        agnostic=self.args.agnostic_nms,
                                        max_det=self.args.max_det)

        for i, pred in enumerate(preds):
            shape = orig_img[i].shape if self.webcam else orig_img.shape
            pred[:, :4] = ops.scale_boxes(img.shape[2:], pred[:, :4], shape).round()

        return preds

    def write_results(self, idx, preds, batch):
        p, im, im0 = batch
        log_string = ""
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        self.seen += 1
        im0 = im0.copy()
        if self.webcam:  # batch_size >= 1
            log_string += f'{idx}: '
            frame = self.dataset.count
        else:
```

```
            frame = getattr(self.dataset, 'frame', 0)

        self.data_path = p
        # save_path = str(self.save_dir / p.name)  # im.jpg
        self.txt_path = str(self.save_dir / 'labels' / p.stem) + ('' if self.dataset.mode ==
'image' else f'_{frame}')
        log_string += '%gx%g ' % im.shape[2:]  # print string
        self.annotator = self.get_annotator(im0)

        det = preds[idx]
        if len(det) == 0:
            return log_string
        for c in det[:, 5].unique():
            n = (det[:, 5] == c).sum()  # detections per class
            log_string += f"{n} {self.model.names[int(c)]}{'s' * (n > 1)}, "

        if self.return_outputs:
            self.output["det"] = det.cpu().numpy()

        # write
        gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
        for *xyxy, conf, cls in reversed(det):
            if self.args.save_txt:  # Write to file
                xywh = (ops.xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()
# normalized xywh
                line = (cls, *xywh, conf) if self.args.save_conf else (cls, *xywh)  # label
format
                with open(f'{self.txt_path}.txt', 'a') as f:
                    f.write(('%g ' * len(line)).rstrip() % line + '\n')

            if self.args.save or self.args.save_crop or self.args.show:  # Add bbox to image
                c = int(cls)  # integer class
                label = None if self.args.hide_labels else (
                    self.model.names[c] if self.args.hide_conf else f'{self.model.names[c]}
{conf:.2f}')
                self.annotator.box_label(xyxy, label, color=colors(c, True))
            if self.args.save_crop:
                imc = im0.copy()
                save_one_box(xyxy,
                             imc,
                             file=self.save_dir / 'crops' / self.model.model.names[c] /
f'{self.data_path.stem}.jpg',
                             BGR=True)

        return log_string


@hydra.main(version_base=None, config_path=str(DEFAULT_CONFIG.parent),
config_name=DEFAULT_CONFIG.name)
def predict(cfg):
    cfg.model = cfg.model or "yolov8n.pt"
```

```
    cfg.imgsz = check_imgsz(cfg.imgsz, min_dim=2)  # check image size
    cfg.source = cfg.source if cfg.source is not None else ROOT / "assets"
    predictor = DetectionPredictor(cfg)
    predictor.predict_cli()


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

Παράρτημα IV – Πίνακας Σύγκρισης Παραμέτρων YOLOv8

Πίνακας 1: Σύγκριση παραμέτρων YOLOv8, παράμετροι 1-33

	A	B	C	D	E	F	G
1	yolov8	yolov8196	yolov8198	yolov8199	yolov8200	yolov8204	yolov8205
2	task	detect	detect	detect	detect	detect	detect
3	mode	train	train	train	train	train	train
4	model	yolov8l.pt	yolov8l.pt	yolov8s.pt	yolov8n.pt	yolov8n.pt	yolov8n.pt
5	data	robotCoco2.yaml	robotCoco2.yaml	robotCoco2.yaml	robotCoco2.yaml	robotCoco2.yaml	robotCoco2.yaml
6	epochs	100	100	100	100	100	100
7	patience	50	50	50	50	50	50
8	batch	16	32	32	64	64	32
9	imgsz	640	320	640	640	480	640
10	save	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
11	save_period	-1	-1	-1	-1	-1	-1
12	cache	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
13	device	null	null	null	null	null	null
14	workers	8	8	8	8	8	8
15	project	null	null	null	null	null	null
16	name	null	null	null	null	null	null
17	exist_ok	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
18	pretrained	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
19	optimizer	SGD	SGD	SGD	SGD	SGD	SGD
20	verbose	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
21	seed	0	0	0	0	0	0
22	deterministic	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
23	single_cls	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
24	image_weights	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
25	rect	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
26	cos_lr	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
27	close_mosaic	0	0	0	0	0	0
28	resume	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
29	amp	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
30	overlap_mask	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
31	mask_ratio	4	4	4	4	4	4
32	dropout	0.2	0.2	0.2	0.2	0.2	0.2
33	val	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

Πίνακας 2: Σύγκριση παραμέτρων YOLOv8, παράμετροι 34-66

	A	B	C	D	E	F	G
34	split	val	val	val	val	val	val
35	save_json	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
36	save_hybrid	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
37	conf	null	null	null	null	null	null
38	iou	0.7	0.7	0.7	0.7	0.7	0.7
39	max_det	300	300	300	300	300	300
40	half	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
41	dnn	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
42	plots	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
43	source	null	null	null	null	null	null
44	show	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
45	save_txt	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
46	save_conf	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
47	save_crop	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
48	show_labels	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
49	show_conf	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
50	vid_stride	1	1	1	1	1	1
51	line_thickness	3	3	3	3	3	3
52	visualize	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
53	augment	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
54	agnostic_nms	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
55	classes	null	null	null	null	null	null
56	retina_masks	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
57	boxes	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
58	format	torchscript	torchscript	torchscript	torchscript	torchscript	torchscript
59	keras	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
60	optimize	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
61	int8	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
62	dynamic	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
63	simplify	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
64	opset	null	null	null	null	null	null
65	workspace	4	4	4	4	4	4
66	nms	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Πίνακας 3: Σύγκριση παραμέτρων YOLOv8, παράμετροι 67-97

	A	B	C	D	E	F	G
67	lr0	0.01	0.01	0.01	0.01	0.01	0.01
68	lrf	0.01	0.01	0.01	0.01	0.01	0.01
69	momentum	0.937	0.937	0.937	0.937	0.937	0.937
70	weight_decay	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
71	warmup_epoch	3	3	3	3	3	3
72	warmup_mome	0.8	0.8	0.8	0.8	0.8	0.8
73	warmup_bias_	0.1	0.1	0.1	0.1	0.1	0.1
74	box	0.05	0.05	0.05	0.05	0.05	0.05
75	cls	0.5	0.5	0.5	0.5	0.5	0.5
76	dfl	0.1	0.1	0.1	0.1	0.1	0.1
77	pose	12	12	12	12	12	12
78	kobj	1	1	1	1	1	1
79	label_smoothi	0	0	0	0	0	0
80	nbs	64	64	64	64	64	64
81	hsv_h	0.015	0.015	0.015	0.015	0.015	0.015
82	hsv_s	0.7	0.7	0.7	0.7	0.7	0.7
83	hsv_v	0.4	0.4	0.4	0.4	0.4	0.4
84	degrees	0	0	0	0	0	0
85	translate	0.1	0.1	0.1	0.1	0.1	0.1
86	scale	0.5	0.5	0.5	0.5	0.5	0.5
87	shear	0	0	0	0	0	0
88	perspective	0	0	0	0	0	0
89	flipud	0	0	0	0	0	0
90	fliplr	0.5	0.5	0.5	0.5	0.5	0.5
91	mosaic	1	1	1	1	1	1
92	mixup	0	0	0	0	0	0
93	copy_paste	0	0	0	0	0	0
94	cfg	null	null	null	null	null	null
95	v5loader	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
96	tracker	botsort.yaml	botsort.yaml	botsort.yaml	botsort.yaml	botsort.yaml	botsort.yaml
97	save_dir	runs/detect/train196	runs/detect/train198	runs/detect/train199	uns/detect/train20(	runs/detect/train204	runs/detect/train205

Παράρτημα V – Πίνακας Σύγκρισης Παραμέτρων YOLOv5 - YOLOv7 - YOLOv8

Πίνακας 4: Σύγκριση παραμέτρων YOLOv5 – YOLOv7 – YOLOv8, παράμετροι 1-33

YOLOv8		YOLOv7		YOLOv5	
task	detect	**weights**	yolov7l.pt	**weights**	yolov5l.pt
mode	train	**cfg**	''	**cfg**	''
model	yolov8l.pt	**data**	robotCoco2.yaml	**data**	robotCoco2.yaml
data	robotCoco2.yaml	**hyp**	hyp.scratch.tiny.yaml	**lr0**	0.01
epochs	60	**epochs**	60	**lrf**	0.01
patience	50	**batch_size**	16	**momentum**	0.937
batch	16	**img_size**	480	**weight_decay**	0.0005
imgsz	480	**rect**	false	**warmup_epochs**	3
save	TRUE	**resume**	false	**warmup_moment**	0.8
save_period	-1	**nosave**	false	**warmup_bias_lr**	0.1
cache	FALSE	**notest**	false	**box**	0.05
device	null	**noautoanchor**	false	**cls**	0.5
workers	8	**evolve**	false	**cls_pw**	1
project	null	**bucket**	''	**obj**	0.1
name	null	**cache_images**	false	**obj_pw**	1
exist_ok	FALSE	**image_weights**	false	**iou_t**	0.7
pretrained	FALSE	**device**	''	**anchor_t**	4
optimizer	SGD	**multi_scale**	false	**fl_gamma**	0
verbose	TRUE	**single_cls**	false	**hsv_h**	0.015
seed	0	**adam**	false	**hsv_s**	0.7
deterministic	TRUE	**sync_bn**	false	**hsv_v**	0.4
single_cls	FALSE	**local_rank**	-1	**degrees**	0
image_weights	FALSE	**workers**	8	**translate**	0.1
rect	FALSE	**project**	runs/train	**scale**	0.5
cos_lr	FALSE	**entity**	null	**shear**	0
close_mosaic	0	**name**	exp	**perspective**	0
resume	FALSE	**exist_ok**	false	**flipud**	0
amp	TRUE	**quad**	false	**fliplr**	0.5
overlap_mask	TRUE	**linear_lr**	false	**mosaic**	1
mask_ratio	4	**label_smoothing**	0	**mixup**	0
dropout	0.2	**upload_dataset**	false	**copy_paste**	0
val	TRUE	**bbox_interval**	-1	**epochs**	60

Πίνακας 5: Σύγκριση παραμέτρων YOLOv5 – YOLOv7 – YOLOv8, παράμετροι 34-66

34	**split**	val	**save_period**	-1	**batch_size**	16
35	**save_json**	FALSE	**artifact_alias**	latest	**imgsz**	480
36	**save_hybrid**	FALSE	**freeze**	0	**rect**	false
37	**conf**	null	**v5_metric**	false	**resume**	false
38	**iou**	0.7	**world_size**	1	**nosave**	false
39	**max_det**	300	**global_rank**	-1	**noval**	false
40	**half**	FALSE	**save_dir**	runs/train/exp30	**noautoanchor**	false
41	**dnn**	FALSE	**total_batch_size**	16	**noplots**	false
42	**plots**	TRUE	**lr0**	0.01	**evolve**	null
43	**source**	null	**lrf**	0.01	**bucket**	''
44	**show**	FALSE	**momentum**	0.937	**cache**	null
45	**save_txt**	FALSE	**weight_decay**	0.0005	**image_weights**	false
46	**save_conf**	FALSE	**warmup_epochs**	3	**device**	''
47	**save_crop**	FALSE	**warmup_moment**	0.8	**multi_scale**	false
48	**show_labels**	TRUE	**warmup_bias_lr**	0.1	**single_cls**	false
49	**show_conf**	TRUE	**box**	0.05	**optimizer**	SGD
50	**vid_stride**	1	**cls**	0.5	**sync_bn**	false
51	**line_thickness**	3	**cls_pw**	1	**workers**	8
52	**visualize**	FALSE	**obj**	0.1	**project**	runs/train
53	**augment**	FALSE	**obj_pw**	1	**name**	exp
54	**agnostic_nms**	FALSE	**iou_t**	0.2	**exist_ok**	false
55	**classes**	null	**anchor_t**	4	**quad**	false
56	**retina_masks**	FALSE	**fl_gamma**	0	**cos_lr**	false
57	**boxes**	TRUE	**hsv_h**	0.015	**label_smoothing**	0
58	**format**	torchscript	**hsv_s**	0.7	**patience**	100
59	**keras**	FALSE	**hsv_v**	0.4	**freeze**	0
60	**optimize**	FALSE	**degrees**	0	**save_period**	-1
61	**int8**	FALSE	**translate**	0.1	**seed**	0
62	**dynamic**	FALSE	**scale**	0.5	**local_rank**	-1
63	**simplify**	FALSE	**shear**	0	**entity**	null
64	**opset**	null	**perspective**	0	**upload_dataset**	false
65	**workspace**	4	**flipud**	0	**bbox_interval**	-1
66	**nms**	FALSE	**fliplr**	0.5	**artifact_alias**	latest

Πίνακας 6: Σύγκριση παραμέτρων YOLOv5 – YOLOv7 – YOLOv8, παράμετροι 67-97

67	**lr0**	0.01	**mosaic**	1	**save_dir**	runs/train/exp61
68	**lrf**	0.01	**mixup**	0.05		
69	**momentum**	0.937	**copy_paste**	0		
70	**weight_decay**	0.0005	**paste_in**	0.05		
71	**warmup_epochs**	3	**loss_ota**	1		
72	**warmup_moment**	0.8				
73	**warmup_bias_lr**	0.1				
74	**box**	0.05				
75	**cls**	0.5				
76	**dfl**	0.1				
77	**pose**	12				
78	**kobj**	1				
79	**label_smoothing**	0				
80	**nbs**	64				
81	**hsv_h**	0.015				
82	**hsv_s**	0.7				
83	**hsv_v**	0.4				
84	**degrees**	0				
85	**translate**	0.1				
86	**scale**	0.5				
87	**shear**	0				
88	**perspective**	0				
89	**flipud**	0				
90	**fliplr**	0.5				
91	**mosaic**	1				
92	**mixup**	0				
93	**copy_paste**	0				
94	**cfg**	null				
95	**v5loader**	FALSE				
96	**tracker**	botsort.yaml				
97	**save_dir**	runs/detect/train194				

Παράρτημα VI – SUMMIT XL GEN Datasheet

Robotnik

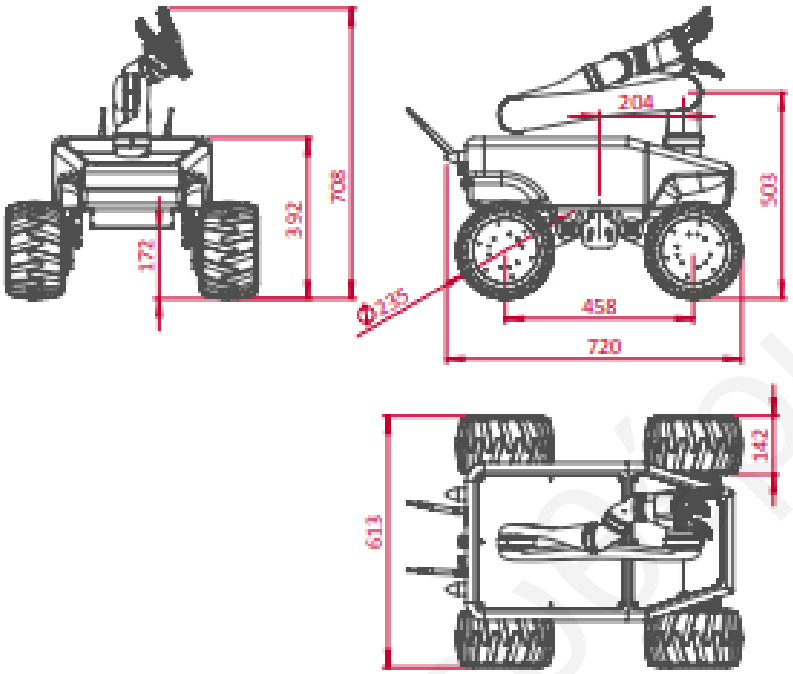
XL-GEN

MOBILE MANIPULATOR
FOR **RESEARCH**
& **DEVELOPMENT**

RESEARCH & DEVELOPMENT
INSPECTION
LOGISTICS
REMOTE MONITORING
ACCESS TO HARSH ENVIRONMENTS

XL-GEN

Robotnik



TECHNICAL SPECIFICATIONS

MECHANICAL

Dimensions	720 x 613 x 708 mm (Folded)
Weight	50 Kg
Payload	15 kg
Speed	3 m/s
Environment	Indoor/Outdoor
Enclosure class	IP54
Autonomy	5-10 h
Batteries	LiFePO, 15Ah@24V
Traction system	4 wheels
Traction motors	4 x 250 W brushless servomotors
Temperature range	0° to +50°C
Max. slope	80 %

CONTROL

Controller	Open architecture ROS Embedded PC with Linux
Communication	WiFi 802.11n
Connectivity	Internal: USB, RS232 and GPIO External: USB, RJ45 and power supply 12 VDC

Arm load capacity (Kinova)

6 DOF	2,6 Kg (Half range) 2,2 Kg (Complete Range)
7 DOF	2,4 Kg (Half range) 2,1 Kg (Complete Range)

Arm reach

6 DOF	900 mm
7 DOF	984 mm

ROS

• C/ Ciudad de Barcelona, 3-A, 46988 Pl Fuente del Jarro, Paterna, Valencia (Spain) • Phone +34 96 147 54 00 • www.robotnik.eu •

Παράρτημα VII – Network Configuration – Remote PC

4. Network Configuration

Install the matching ros version in the remote machine (follow instructions in www.ros.org).

Configure the network following the steps in http://www.ros.org/wiki/ROS/NetworkSetup

4.1 Ethernet

Connect to the router by plugging an Ethernet cable to the robot back pannel:

Summit - Static IP address 192.168.0.200
Subnet Mask 255.255.255.0
Gateway address: 192.168.0.1

The router is configured to give the connected computer an IP address via DHCP.

4.2 Wireless

The wifi router configuration by default is the following:

You can start connecting with

Wifi:
Wifi SSIDs: Summit XL (A..Z)
Wifi Password: R0b0tn1K (R and K capital letters)

Router:
User/Password: root / R0b0tn1K
Router IP Address : 192.168.0.1

Robot Computer:
User/Password: summit / R0b0tn1K

And the password for the AXIS camera is also R0b0tn1K

6. Remote PC

6.1 Software installation and PC configuration

Some metapackages of the software have to be present in the remote computer in order to operate the robot remotely. In order to test the different components, the most useful tools are rviz (ROS Visualization tool) and rqt. In order to use these **in the remote machine**, we will need to:

1. Install ROS
2. Install summit_xl_common and summit_xl_sim stacks in the remote computer (see gitHub repositories). The second is not necessary, but is useful if we want to simulate the robot.
3. Compile the stacks:

   Configure a catkin workspace if you don't have it:

```
>mkdir ~/catkin_ws/src
>cd ~/catkin_ws/src
>catkin_init_workspace

>mkdir -p ~/catkin_ws/src
>cd ~/catkin_ws/src
>catkin_init_workspace
```

Even though the workspace is empty (there are no packages in the 'src' folder, just a single CMakeLists.txt link) you can still "build" the workspace:

```
>cd ~/catkin_ws/
>catkin_make
```

The creation of catkin_workspaces is described in:
http://www.ros.org/wiki/catkin/Tutorials/create_a_workspace

After that copy or link the summit_xl_common and the summit_xl_sim folders to this workspace

```
>cd ~/sources
>git clone https://github.com/RobotnikAutomation/summit_xl_common
>git clone https://github.com/RobotnikAutomation/summit_xl_sim
>cd ~/catkin_ws/src
>ln -sf ~/sources/summit_xl_common
>ln -sf ~/sources/summit_xl_sim
```

And compile:

```
>cd ~/catkin_ws
>catkin_make
```

Once ROS is installed in your machine you will need to configure this machine so that ROS connects to the Summit XL ROS_MASTER

Add the following line to your /etc/hosts file

```
192.168.0.200    summit
```

Start the Summit XL robot, connect to its wifi network, open a terminal and type:

```
>export ROS_MASTER_URI=http://summit:11311

>rostopic list
```

Add your computer hostname in the rover /etc/hosts file

```
>ssh summit@summit
>sudo vim /etc/hosts
>   …  // add your hostname and ip address there
```

If everything worked you should be able to see all the topics published by the robot and the services offered by the robot controller:

```
>rostopic list
…
>rosservice list
…
```

You can view the values published by the nodes with rostopic echo, e.g.:

```
>rostopic echo /summit_xl_control/odom
```

At this stage you can have a first look at the robot with

```
>rqt
```

and

```
>rosrun rviz rviz
```

Παράρτημα VIII – Mavic 2 Specifications

Specifications

• Aircraft	
Weight	905 g
Max Speed	44.7 mph (72 kph) in Sport mode without wind
Max Service Ceiling Above Sea Level	19685 ft (6000 m)
Operating Temperature	14° to 104° F (-10° to 40° C)
GNSS	GPS + GLONASS
Operating Frequency	2.4-2.4835 GHz; 5.725-5.850 GHz
Transmitter Power (EIRP)	2.4 GHz FCC: ≤26 dBm; CE/MIC: ≤20 dBm; SRRC: ≤20 dBm 5.8 GHz FCC: ≤26 dBm; CE: ≤14 dBm; SRRC: ≤26 dBm
Internal Storage	24 GB
• Gimbal	
Controllable Range	Pitch: -90° to +30°
• Camera	
Sensor	1/2.3" CMOS; Effective pixels: 12M
Lens	FOV: approx. 83° (24 mm), approx. 48° (48 mm) 35 mm format equivalent: 24-48 mm Aperture: f/2.8 (24 mm) - f/3.8 (48 mm) Focus: 0.5 m to ∞
ISO Range	Video: 100-3200 Photo: 100-1600 (auto); 100-12800 (manual)
Electronic Shutter Speed	8-1/8000 s
Max Image Size	4000×3000
Still Photography Modes	Single shot Burst shooting: 3/5/7 frames Auto Exposure Bracketing (AEB): 3/5 bracketed frames at 0.7 EV Bias Interval
Video Recording Modes	4K Ultra HD: 3840×2160 24/25/30p 2.7K: 2688×1512 24/25/30/48/50/60p FHD: 1920×1080 24/25/30/48/50/60/120p
Video Storage Bitrate	100 Mbps
Photo	JPEG, DNG (RAW)
Video	MP4, MOV (MPEG-4 AVC/H.264, HEVC)
Supported SD Cards	microSD Max Capacity: 128 GB (UHS-I Speed Grade 3 rating required)
• Remote Controller	
Operating Frequency	2.4-2.4835 GHz; 5.725-5.850 GHz
Max Transmission Distance (Unobstructed and free of interference)	FCC: 10 km; CE/MIC: 6 km; SRRC: 6 km
Operating Temperature	32° to 104° F (0° to 40° C)

Battery	3950mAh @ 3.83V
Transmitter Power (EIRP)	2.4 GHz FCC: ≤26 dBm; CE/MIC: ≤20 dBm; SRRC: ≤20 dBm 5.8 GHz FCC: ≤26 dBm; CE: ≤14 dBm; SRRC: ≤26 dBm
Operating Voltage	1800mA @ 3.83V (when charging the mobile device)
Supported Mobile Device Size	Thickness supported: 6.5 - 8.5 mm, Max length: 160 mm Supported USB port types: Lightning, Micro USB (Type-B) USB-C

• **Charger**

Voltage	17.6±0.1 V
Rated Power	60 W

• **Intelligent Flight Battery**

Capacity	3850 mAh
Voltage	17.6 V (max) 15.4 V (typical)
Battery Type	LiPo 4S
Energy	59.29 Wh
Net Weight	Approx. 297 g
Charging Temperature Range	41° to 104° F (5° to 40° C)
Max Charging Power	80 W
Auto Heating Temperature Range	-4° to 43° F (-20° to 6° C)
Auto Heating Time	600 s (max)
Auto Heating Power	35 W (max)

For more information, read the User Manual:
www.dji.com/mavic-2-enterprise

※ This content is subject to change without prior notice.

Designed by DJI. Printed in China.

Παράρτημα IX – Scanning Laser Manual

Date: 2014.6.12

Scanning Laser Range Finder
Smart-URG mini
UST-20LX (UUST004)
Specification

CE
RoHS

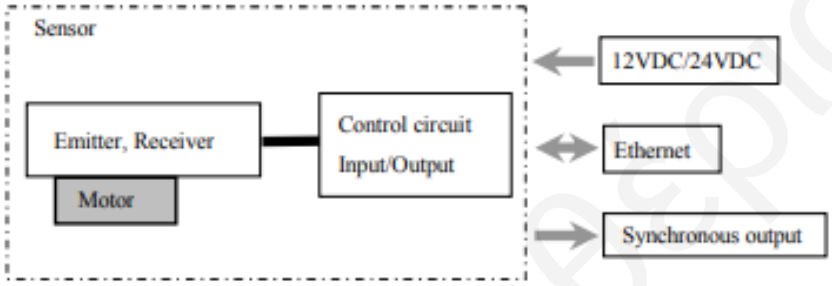
Symbol	Amended Reason				Pages	Date	Amended by	Ref.No
Approved by	Checked by	Drawn by	Designed by	Title	UST-20LX (UUST004) Specification		
Kamitani	Utsugi	Kamon	Yamamoto	Drawing No.	C-42-04078		1/6

1. General

This sensor uses a laser source to scan 270° field of view. Positions of objects in the range are calculated with step angle and distance. Sensor outputs these data through communication channel.

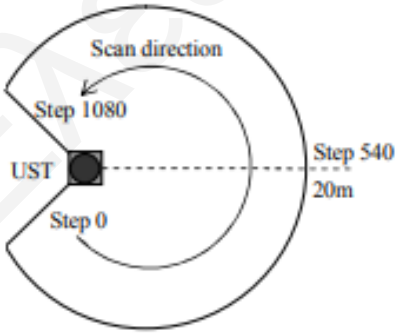
2. Structure

2-1.Strucure diagram



2-2. Laser scanning image

Measurement steps 1081
Detection angle 270°
Angular resolution 0.25°



3. Important notes

(1) This sensor is not a safety device/tool.

(2) This sensor cannot be used for human body detection as per the machinery directives.

(3) Hokuyo products are not developed and manufactured for the use in weapons, equipments or related technologies intended for destroying human lives or causing mass destruction. If such possibilities or usages are revealed, the sales of Hokuyo products to those customers might be halted by the laws of Japan such as Foreign Exchange Law, Foreign Trade Law or Export Trade control order. In addition, Hokuyo products are for the purpose of maintaining the global peace and security in accordance with the above law of Japan.

4. Specifications

Product name	Scanning Laser Range Finder
Model	UST-20LX
Supply voltage	12VDC/24VDC (Operation range 10 to 30V ripple within 10%)
Supply current	150mA or less (during start up 450mA is necessary.)
Light source	Laser semiconductor (905nm) Laser class 1 (IEC60825-1:2007)
Detection range	0.06m to 20m (white Kent sheet) 0.06m to 8m (diffuse reflectance 10%) Max. detection distance : 60m
Accuracy	±40mm (*1)
Repeated accuracy	σ< 30mm (*1)
Scan angle	270°
Scan speed	25ms (Motor speed 2400rpm)
Angular resolution	0.25°
Start up time	Within 10 sec (start up time differs if malfunction is detected during start up)
Input	IP reset input, photo-coupler input (current 4mA at ON)
Output	Synchronous Output, photo coupler open collector output 30VDC 50mA MAX.
Interface	Ethernet 100BASE-TX
LED display	Power supply LED display (Blue): Blinks during start up and malfunction state.
Surrounding intensity	Less than 15,000lx Note : Avoid direct sunlight or other illumination sources as it may cause sensor malfunction
Ambient temperature and humidity	-10°C to +50°C, below 85%RH (without dew, frost)
Storage temperature and humidity	-30°C to +70°C, below 85%RH (without dew, frost)
Vibration resistance	10 to 55Hz double amplitude of 1.5mm for 2hrs in each X, Y, and Z direction 55 to 200Hz 98m / s^2 sweep of 2min for 1hr in each X,Y and Z direction
Vibration resistance (Operating)	55 to 150Hz 19.6m / s^2 sweep of 2min for 30min in each X,Y and Z direction
Shock resistance	196m/s^2 (20G) X,Y and Z direction 10 times.
EMC standards	(EMI) EN61326-1:2013 EN55011:2009 + A1:2010 (EMS) EN61326-1:2013 EN61000-4-2:2009 EN61000-4-3:2006 + A1:2008 + A2:2010 EN61000-4-4:2012 EN61000-4-6:2009 EN61000-4-8:2010
Protective Structure	IP65
Weight	130g (Excluding cable)
Material	Front case: Polycarbonate, Rear case: Aluminum
Dimensions (W×D×H)	50×50×70mm (sensor only)

(*1) Under the factory standard testing conditions using white Kent sheet.

5. Measurement Data

Distance Value (x)	Meaning
x < 21	Output numerical number "4" as Measurement error
21 ≦ x ≦ 60000	Valid distance [mm]
x > 60000	Output numerical number "65533" as Measurement error (object does not exists or object has low reflectivity)

6. Connection

6-1. Power source, I/O cable

Cable length: 1000mm Flying lead cable (AWG28)

Color	Signal
Red	COM Input +
Gray	COM Output -
Light Blue	IP Reset Input
Orange	Synchronous Output
Brown	+VIN (12VDC/24VDC)
Blue	-VIN

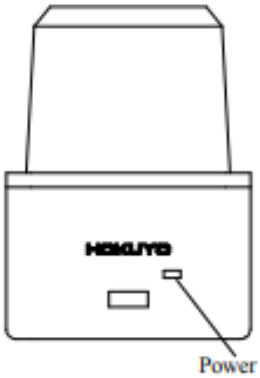
Note: Direction of Inputs and Outputs are mentioned from the sensor's side.

6-2. Ethernet cable

Cable length: 300mm

Color	Signal
Blue	TX+
White	TX-
Orange	RX+
Yellow	RX-

7. LED display

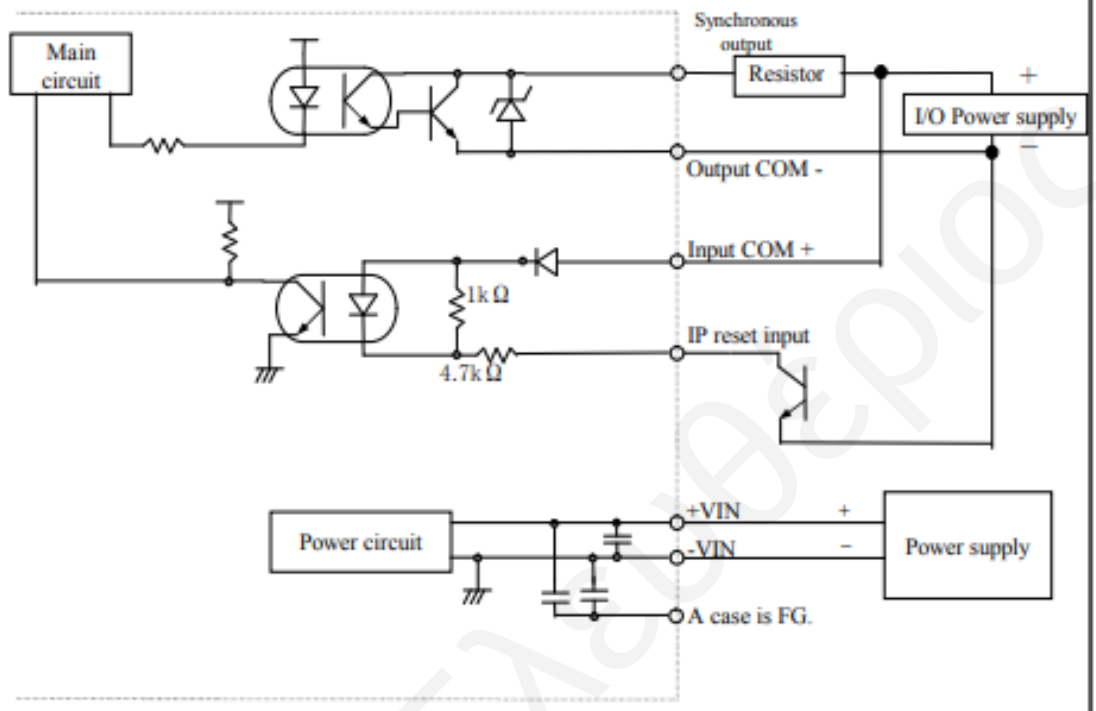


Power supply display

(Blinks during start up and malfunction state)

Title	UST-20LX Specification	Drawing No	C-42-04078	4/6

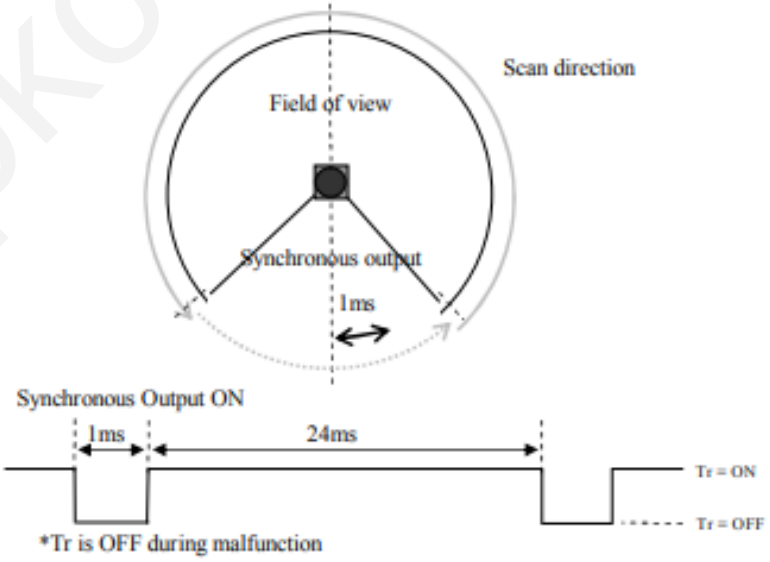
8. Output circuit



9. Control signal

9-1. Synchronous output

1 pulse is approximately 1ms. Output signal synchronization timing chart is shown as below.



10. Ethernet Setting

1. The setting value is as below.

   IP Initial value :192.168.0.10

   Port number :10940

2. About Initialization of IP address

   To reset IP address to the factory default value, connect IP RESET LINE to COM- for more than 2 sec. After IP RESET LINE disconnected from COM- or opened, the sensor LED blinks and the sensor start to reboot.

11. Cautions for operation

This sensor uses high speed processing components that generate heat during operation.
The heat is concentrated at the bottom of the unit. When mounting, please attach the bottom of the unit to a good heat sink. A 200mm x 200mm x 2mm aluminum plate is recommended as a heat sink.

If multiple sensors are installed side by side, a sensor might mistake the laser pulses of other units as its own and the detection error occurs. When it happens, usually the error lasts for one or two steps of measurement. Please use software filters to handle this type of error.

Παράρτημα Χ – Τεχνικά Χαρακτηριστικά φορητού υπολογιστή που συνδέεται με το SUMMIT XL

id: display
description: VGA compatible controller
product: GP104BM [GeForce GTX 1080 Mobile]
vendor: NVIDIA Corporation
physical id: 0
bus info: pci@0000:01:00.0
version: a1
width: 64 bits
clock: 33MHz
capabilities: *pm msi pciexpress vga_controller bus_master cap_list rom*
configuration: driver = nvidia
latency = 0
resources: irq : 145
memory : ab000000-abffffff
memory : 80000000-8fffffff
memory : 90000000-91ffffff
ioport : 3000(size=128)
memory : c0000-dffff

id: memory
description: System Memory
physical id: 8
slot: System board or motherboard
size: 16GiB

id: cpu
description: CPU
product: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
vendor: Intel Corp.
physical id: 16
bus info: cpu@0
version: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz
serial: To Be Filled By O.E.M.
slot: U3E1
size: 3901MHz
capacity: 4100MHz
width: 64 bits
clock: 100MHz
capabilities: *lm fpu fpu_exception wp vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp x86-64 constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d arch_capabilities cpufreq*
configuration: cores = 6
enabledcores = 6
threads = 12

id: pci
description: Host bridge
product: 8th Gen Core Processor Host Bridge/DRAM Registers
vendor: Intel Corporation
physical id: 100
bus info: pci@0000:00:00.0
version: 07
width: 32 bits
clock: 33MHz
configuration: driver = skl_uncore
resources: irq : 0

id: display
description: VGA compatible controller
product: GP104BM [GeForce GTX 1080 Mobile]
vendor: NVIDIA Corporation
physical id: 0
bus info: pci@0000:01:00.0
version: a1
width: 64 bits
clock: 33MHz
capabilities: *pm msi pciexpress vga_controller bus_master cap_list rom*
configuration: driver = nvidia
latency = 0
resources: irq : 145
memory : ab000000-abffffff
memory : 80000000-8fffffff
memory : 90000000-91ffffff
ioport : 3000(size=128)
memory : c0000-dffff