ΣΤΡΑΤΙΩΤΙΚΗ ΣΧΟΛΗ ΕΥΕΛΠΙΔΩΝ
Τμήμα Στρατιωτικών Επιστημών

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΔΙΙΔΡΥΜΑΤΙΚΟ ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΧΕΔΙΑΣΗ & ΕΠΕΞΕΡΓΑΣΙΑ ΣΥΣΤΗΜΑΤΩΝ
(SYSTEMS ENGINEERING)
(ΠΔ 96/2015/ΦΕΚ 163 Α'/20.08.2014)

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Σχολή Μηχανικών Παραγωγής & Διοίκησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

Ανάπτυξη μη επανδρωμένου συστήματος για αναγνώριση στόχων με χρήση ανάλυσης εικόνας με τη βοήθεια τεχνικών μηχανικής μάθησης

Υπό:

Αναστάσιου Αντωνίου

Α.Μ.: 2014018002

ΙΑΝΟΥΑΡΙΟΣ 2022

Η Μεταπτυχιακή Διατριβή του Αναστάσιου Αντωνίου, εγκρίνεται:

ΤΡΙΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Καθηγήτρια Ειρήνη Καρανάσιου (Επιβλέπουσα),

Καθηγητής Νικόλαος Δάρας,

Αναπληρωτής Καθηγητής Στυλιανός Τσαφαράκης

Stelios Tsafarakis

Digitally signed by Stelios Tsafarakis
DN: c=GR, l=Chania, o=Technical University of Crete, ou=Class B - Private Key created and stored in software CSP, sn=Tsafarakis, givenName=Stelios, serialNumber=4644645843, cn=Stelios Tsafarakis, email=tsafarakis@dpem.tuc.gr
Date: 2022.05.09 13:42:17 +03'00'

Στην αγαπημένη μου οικογένεια

Πίνακας περιεχομένων

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Τμημάτων κώδικα

Κατάλογος Συντμήσεων

AI	Artificial Intelligence
CPU	Central Processing Unit
CV	Computer Vision
GPS	Global Positioning System
GPU	Graphics processing unit
GTK	Gnome Tool Kit
GUI	Graphic User Interface
HOTOL	Horizontal Take Off and Landing
OS	Operating System
RAM	Random Access Memory
SBC	Single Board Computer
SUAV	Small UAV
UAV	Unmanned Aerial Vehicle
UV	Unmanned Vehicle
VPU	Vision Processing Unit
VTOL	Vertical Take Off and Landing

Περίληψη

Στόχος της διπλωματικής εργασίας αυτής είναι η ανάπτυξη τόσο σε υλικό όσο και σε λογισμικό ενός συστήματος ικανού να εντοπίζει προκαθορισμένου τύπου αντικείμενο και στη συνέχεια να το παρακολουθεί αδιάκοπτα οδηγώντας ταυτόχρονα μία κάμερα η οποία με την κίνησή της θα διατηρεί το στόχο πάντα εντός του τρέχοντος στιγμιοτύπου. Για το σκοπό αυτό προτείνεται μία αρχιτεκτονική βασισμένη σε Raspberry Pi που συνεργάζεται με έναν συνεπεξεργαστή intel movidius και ελέγχει ένα σύστημα σερβοκινητήρων υπεύθυνο για την κίνηση της κάμερας. Για την αρχιτεκτονική αυτή στο πλαίσιο της διπλωματικής εργασίας αναπτύχθηκε κώδικας σε γλώσσα προγραμματισμού Python που εκτελείται το Raspberry Pi που διαθέτει λειτουργικό σύστημα Raspbian και λαμβάνει και επεξεργάζεται δεδομένα από το μοντέλο μηχανικής μάθησης που εκπαιδεύτηκε και εκτελείται στον συνεπεξεργαστή movidius. Το προτεινόμενο σύστημα δύναται να εφαρμοστεί σε μη επανδρωμένα αεροσκάφη με σκοπό την προσθήκη τεχνητής νοημοσύνης και υπολογιστικής όρασης σε αυτά.

Λέξεις Κλειδιά: αναγνώριση αντικειμένου, μηχανική μάθηση, ιχνηλάτηση αντικειμένου, τεχνητή νοημοσύνη

Abstract

The aim of the present thesis is the development of both hardware and software components of a system capable of detecting a predefined type of object and then continuously tracking it while simultaneously moving a camera which will always keep the target within the current frame. For this purpose, a Raspberry Pi-based architecture is proposed that cooperates with an intel movidius co-processor and a servomotor system (gimbal) that controls the camera movements. Contribution of the present thesis is also the source code that was developed using the Python programming language running on a Raspberry Pi powered by Raspbian OS that is receiving and processing data from the trained machine learning model that is executed on the movidius co-processor. The proposed system can be part of any unmanned aircraft in order to equip it with artificial intelligence and computer vision capabilities.

Key-words: object detection, machine learning, object tracking, artificial intelligence

Εισαγωγή

Μη επανδρωμένο θεωρείται το όχημα το οποίο μπορεί να καθοδηγηθεί χωρίς την παρουσία ανθρώπου μέσα σε αυτό. Η καθοδήγηση δύναται να πραγματοποιηθεί είτε απομακρυσμένα (με τηλεκατεύθυνση) είτε αυτόνομα (π.χ. με χρήση GPS). Συνεπώς ένα μη επανδρωμένο όχημα έχει τη δυνατότητα να αντιλαμβάνεται το περιβάλλον γύρω του και να ορίζει την πορεία του χωρίς να κρίνεται αναγκαία η φυσική ύπαρξη του οδηγού εντός του οχήματος.

Το χαρακτηριστικό αυτό των μη επανδρωμένων οχημάτων, δηλαδή η μη ύπαρξη οδηγού προσδίδει στα οχήματα αυτά τη δυνατότητα χρησιμοποίησής τους σε πλήθος εφαρμογών όπου η ύπαρξη ανθρώπου είναι είτε ανέφικτη είτε επικίνδυνη. Χαρακτηριστικά παραδείγματα χρήσης των μη επανδρωμένων οχημάτων είναι στην πυρασφάλεια, στην διανομή προϊόντων και σε πληθώρα στρατιωτικών εφαρμογών.

Η αποτελεσματικότητα της χρήσης των μη επανδρωμένων οχημάτων μπορεί να αυξηθεί σημαντικά αν η καθοδήγησή τους γίνει με τεχνικές τεχνητής νοημοσύνης (Artificial Intelligence[14]). Συνηθισμένες συμβατικές μέθοδοι καθοδήγησης, όπως αναφέρθηκε παραπάνω, είναι είτε με τηλεκατεύθυνση είτε με χρήση GPS. Οι δύο αυτές μέθοδοι παρουσιάζουν σημαντικά μειονεκτήματα. Η καθοδήγηση μη επανδρωμένου οχήματος με τηλεκατεύθυνση περιορίζει το εύρος κίνησης του αεροσκάφους στη μέγιστη εμβέλεια της τηλεκατεύθυνσης. Ενώ η καθοδήγηση μη επανδρωμένου οχήματος με χρήση GPS μπορεί να μη καταστεί εφικτή σε περιοχές (π.χ. στρατιωτικές) όπου το σήμα GPS μπορεί σκοπίμως να υποστεί σημαντικές παρεμβολές.

Γίνεται συνεπώς εύκολα κατανοητό ότι η αυτόματη καθοδήγηση ενός μη επανδρωμένου οχήματος με τεχνικές τεχνητής νοημοσύνης μπορεί να προσδώσει σημαντικά πλεονεκτήματα στην κατασκευή και ανάπτυξη τέτοιων οχημάτων. Η παρούσα διπλωματική εργασία σκοπεύει να αναπτύξει ένα σύστημα το οποίο με μηχανική μάθηση (machine learning[15]) να αναγνωρίζει προκαθορισμένους στόχους (object detection[16]) και στη συνέχεια με ευφυείς αλγορίθμους ιχνηλάτησης αντικειμένου (object tracking[17]) να το παρακολουθεί απρόσκοπτα ρυθμίζοντας παράλληλα τη θέση στους δύο άξονες ενός gimbal (μιας βάση/αναρτήρα εξισορρόπησης) που στηρίζει την κάμερα με σκοπό ο εν λόγω στόχος να βρίσκεται πάντα μέσα στο οπτικό πεδίο (field of view) της κάμερας. Μελλοντικός στόχος είναι οι πληροφορίες αυτές να μεταφέρονται και στον ελεγκτή πτήσης (flight controller π.χ. Pixhawk cube) έτσι ώστε η κίνηση του στόχου να προσδιορίζει την κίνηση του μη επανδρωμένου το οποίο θα μπορεί να είναι απαλλαγμένο από τη χρήση τηλεκατεύθυνσης και σήματος GPS.

Η ιχνηλάτηση αντικειμένου (object tracking) είναι ένα πεδίο της επιστήμης των υπολογιστών που αποσκοπεί στον συνεχιζόμενο εντοπισμό ενός προτύπου (pattern) σε διαδοχικά frame. Πιο συγκεκριμένα η ιχνηλάτηση αντικειμένου ασχολείται με τον εντοπισμό

στιγμιότυπων σημασιολογικών αντικειμένων μίας συγκεκριμένης κατηγορίας (όπως άνθρωποι, κτίρια ή αμάξια) σε ψηφιακές εικόνες και βίντεο. Η ανίχνευση αντικειμένου χρησιμοποιείται σε πολλές εφαρμογές, όπως ανάκτηση εικόνας και παρακολούθηση βίντεο. Οι τομείς που έχουν ευρεία εξέλιξη σχετικά με τον εντοπισμό αντικειμένου είναι η ανίχνευση προσώπου και η ανίχνευση πεζών.

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης και της επιστήμης υπολογιστών ή οποία επικεντρώνεται στη χρήση δεδομένων και αλγορίθμων για να μιμηθεί τον τρόπο με τον οποίο μαθαίνουν οι άνθρωποι, βελτιώνοντας σταδιακά την ακρίβειά του. Στην εν λόγω εργασία χρησιμοποιούμε την βιβλιοθήκη ανοιχτού κώδικα εικόνας υπολογιστή (OpenCV) που περιέχει αλγορίθμους μηχανικής μάθησης.

Η ιχνηλάτηση αντικειμένου είναι μία εφαρμογή της βαθιάς μάθησης (deep learning[18]) όπου το πρόγραμμα παίρνει τα αρχικά στιγμιότυπα του αντικειμένου, δημιουργεί ξεχωριστά δεδομένα για το καθένα και έπειτα ιχνηλατεί τα εντοπισμένα αντικείμενα όσο κινούνται ανάμεσα στα στιγμιότυπα(frames) ενός βίντεο. Πιο συγκεκριμένα η ιχνηλάτηση αντικειμένου είναι η διαδικασία που αναγνωρίσει αυτόματα τα αντικείμενα σε ένα βίντεο και ερμηνεύει ως σύνολο τροχιών με υψηλή ακρίβεια. Συνήθως υπάρχει μία ένδειξη γύρω από το αντικείμενο , π.χ. ένα τετράγωνο δείχνοντας στον χρήστη σε ποιο σημείο βρίσκεται το αντικείμενο πάνω στην οθόνη. Μερικοί από τους γνωστούς αλγορίθμους οι οποίοι περιέχονται στο OpenCV είναι οι: α) Boosting Tracker, β) MIL Tracker, γ) KCF Tracker, δ) CSRT Tracker, ε) MedianFlow Tracker, στ) TLD Tracker, ζ) MOSSE Tracker και η) GOTURN Tracker

Στην προτεινόμενη αρχιτεκτονική η διαδικασία της αρχικής αναγνώρισης αντικειμένου ενδιαφέροντος γίνεται με χρήση τεχνικών μηχανικής μάθησης με μοντέλο που έχει αναπτυχθεί και εκπαιδευτεί και εκτελείται σε επεξεργαστή ειδικού σκοπού movidius[22]. Το movidius επικοινωνεί με ένα raspberry pi το οποίο είναι το core module της προτεινόμενης αρχιτεκτονικής υπεύθυνο για την λήψη βίντεο, επικοινωνίας με το movidius, εκτέλεση του προγράμματος υλοποίησης του αλγορίθμου ιχνηλάτησης και καθοδήγησης του gimbal της κάμερας.

Το gimbal είναι μία βάση στήριξης της κάμερας που επιτρέπει την περιστροφή της γύρω από δύο άξονες. Χαρακτηριστικά παραδείγματα χρήσης gimbal είναι σε πλοία, τα γυροσκόπια και οι πυξίδες που χρησιμοποιούν gimbal για να τα κρατήσουν παράλληλα προς τον ορίζοντα, παρά τις κινήσεις του πλοίου λόγω κύματος (pitching and rolling). Η αρχή λειτουργίας του gimbal στηρίζεται στην ύπαρξη δύο σερβοκινητήρων οι οποίοι είναι υπεύθυνοι για τον έλεγχο της κίνησης στον κάθε άξονα. Στην υλοποίηση της διπλωματικής αυτής εργασίας οι σερβοκινητήρες του gimbal ελέγχονται με ειδικό πρόγραμμα που έχει αναπτυχθεί (σε γλώσσα προγραμματισμού python) και εκτελείται στο raspberry pi.

Το Raspberry Pi είναι ένας χαμηλού κόστους υπολογιστής που συνήθως έχει μέγεθος . 56.5 mm * 85.6 mm. Για να λειτουργήσει χρειάζεται μία οθόνη, πληκτρολόγιο και ποντίκι τα οποία συνδέονται με την ηλεκτρονική πλακέτα. Χρησιμοποιείται κυρίως για προγραμματισμό με τις γλώσσες Scratch και Python. Είναι ικανό να κάνει οτιδήποτε μπορεί να κάνει ένας σταθερός

υπολογιστής, όπως πρόσβαση στο διαδίκτυο, προβολή βίντεο υψηλής ανάλυσης, επεξεργασία κειμένου.

ΚΕΦΑΛΑΙΟ 1
Μη επανδρωμένα οχήματα (UV)

Στο κεφάλαιο αυτό γίνεται μία σύντομη περιγραφή και κατηγοριοποίηση των μη επανδρωμένων οχημάτων (Unmanned Vehicles[19]) στα οποία είναι εφικτό να προστεθεί το προτεινόμενο σύστημα με σκοπό να αυξήσει την ευφυία τους. Ένα σύγχρονο μη επανδρωμένο αεροσκάφος φαίνεται στην Εικόνα 1

Εικόνα 1 Σύγχρονο μη επανδρωμένο αεροσκάφος[27]

Η παλαιότερη καταγεγραμμένη χρήση μη επανδρωμένου αεροσκάφους έγινε τον Ιούλιο του 1849 για στρατιωτικούς σκοπούς, ήταν ένα αερόστατο που χρησιμοποιήθηκε ως φορέας βόμβας και ήταν η πρώτη φορά που χρησιμοποιήθηκε ως αεροπορική ισχύς από το πολεμικό ναυτικό(πρόδρομος αεροπλανοφόρου). Στη Βενετία που πολιορκούνταν από την Αυστρία επιχειρήθηκαν να σταλθούν περίπου 200 αερόστατα με μια βόμβα 24 έως 30 λιβρών πάνω στο καθένα. Στο αερόστατο για να μην υπάρχει άνθρωπος να το χειρίζεται χρησιμοποιήθηκε χρονόμετρο με το οποίο θα έπεφταν οι βόμβες πάνω από την πολιορκημένη πόλη. Τα αερόστατα απογειώθηκαν κυρίως από τη στεριά και μερικά εκτοξεύθηκαν από τη θάλασσα μέσω του

πλοίου SMS Vulcano. Οι Αυστριακές δυνάμεις χρησιμοποίησαν μικρότερα πιλοτικά μπαλόνια για να καθορίσουν τις σωστές ρυθμίσεις ασφάλειας. Τουλάχιστον μία βόμβα έπεσε στην πόλη. Ωστόσο, λόγω της αλλαγής του ανέμου μετά την εκτόξευση, τα περισσότερα από τα μπαλόνια έχασαν τον στόχο τους και μερικά παρασύρθηκαν πίσω από τις γραμμές της Αυστρίας και το πλοίο εκτόξευσης Vulcano. Εκ τότε σημαντικά βήματα έχουν γίνει στην εξέλιξη των μη επανδρωμένων οχημάτων με αποτέλεσμα να χρησιμοποιούνται όχι μόνο σε στρατιωτικές εφαρμογές, αλλά και σε πλήθος άλλων εφαρμογών όπως εμπορικές, ιατρικές, γεωλογικές, διαστημικές κτλ. Παράλληλα, τα είδη των μη επανδρωμένων οχημάτων έχουν αυξηθεί σημαντικά και μία κατηγοριοποίηση αυτών παρουσιάζονται στο επόμενο υποκεφάλαιο.

1.1 Κατηγορίες μη επανδρωμένων οχημάτων

Τα μη επανδρωμένα οχήματα διακρίνονται στις παρακάτω κατηγορίες:

α) Τηλεκατευθυνόμενο όχημα

β) Μη επανδρωμένο όχημα εδάφους

γ) Μη επανδρωμένο αεροσκάφος (Unmanned Aerial Vehicle ή UAV) (στο οποίο βασίζεται η παρούσα εργασία)

δ) Αυτόνομη πλατφόρμα προσγείωσης διαστημικών πυραύλων

ε) Μη επανδρωμένο όχημα θαλάσσης που διακρίνεται σε μη επανδρωμένο όχημα επιφάνειας θαλάσσης και μη επανδρωμένο υποβρύχιο όχημα.

στ) Μη επανδρωμένο διαστημόπλοιο

1.2 Είδη μη επανδρωμένων αεροσκαφών

Στην πλειοψηφία τους τα αεροσκάφη χειρίζονται από έμψυχο δυναμικό το οποίο διαθέτει την κατάλληλη εκπαίδευση. Με την πρόοδο της τεχνολογίας ο χειρισμός των αεροσκαφών είναι εφικτό να επιτευχθεί από υπολογιστικά συστήματα που διαθέτουν κατάλληλη τεχνητή νοημοσύνη. Στην περίπτωση αυτή αναφερόμαστε σε μη επανδρωμένα αεροσκάφη (Unmanned Aerial Vehicle - UAV) τα οποία έχουν σχήμα παρόμοιο με αυτών των πραγματικών αεροσκαφών ή την μορφή των τετρακόπτερων και οχτακόπτερων (drones) και έχουν την ικανότητα να πραγματοποιήσουν πτήσεις χωρίς την απαραίτητη παρουσία χειριστή στην άτρακτό τους, αλλά πραγματοποιούν πτήσεις είτε με προκαθορισμένη πορεία (mission

planning) είτε μέσω της τηλεκατεύθυνσης από χειριστή που δέχεται την εικόνα από το αεροσκάφος και το κατευθύνει(first person view-FPV).

Με βάση τον τύπο τους διακρίνονται σε:

α. Μη επανδρωμένο μαχητικό αεροσκάφος

β. Μέτριου υψομέτρου UAV υψηλής αυτονομίας

γ. Μίνι UAV (SUAV)

δ. UAV μεταφοράς (Delivery drone)

ε. Μικρού μεγέθους UAV (MAV)

στ. Target Drone ή αλλιώς «Καμικάζι» Drone

ζ. Vertical Take Off and Landing(VTOL) Drone

η. Horizontal Take Off and Landing(HOTOL) Drone

Μπορούν επίσης να κατηγοριοποιηθούν όπως και όλα τα υπόλοιπα αεροσκάφη, αναλόγως το βάρος τους ή τους κινητήρες τους, μέγιστο ύψος πτήσης, βαθμός αυτονομίας, επιχειρησιακός ρόλος κτλ.

Με βάση το ύψος τους διακρίνονται σε:

α. 2000 πόδια, 2km εμβέλεια

β. 5000 πόδια, 10 km εμβέλεια

γ. 10000 πόδια, 50 km εμβέλεια

δ. 18000 πόδια, 160 km εμβέλεια

ε. 30000 πόδια, 200 km εμβέλεια

στ. πάνω από 30000 πόδια έως 50000 πόδια

ζ. 50000 πόδια και πάνω από 200 km εμβέλεια

Στις μέρες γίνεται συστηματική έρευνα και ανάπτυξη κυρίως μικρών VTOL drones που δραστηριοποιούνται έως 5000 πόδια κυρίως για εμπορικές εφαρμογές και διανομές-μεταφορές μικρών φορτίων. Σε αντίθεση για στρατιωτικές εφαρμογές η έρευνα επικεντρώνεται στην υλοποίηση HOTOL μεγάλης εμβέλειας.

ΚΕΦΑΛΑΙΟ 2
Ιχνηλάτηση αντικειμένου

2.1 Ορισμός ιχνηλάτησης αντικειμένου (object tracking)

Ιχνηλάτηση αντικειμένου είναι ένα πεδίο της βαθιάς μάθησης όπου το σύστημα δέχεται ως είσοδο ένα αρχικό σύνολο προτύπων που έχουν ανιχνευθεί, αναπτύσσει μία μοναδική ταυτοποίηση για κάθε μία από τις αρχικές ανιχνεύσεις και στη συνέχεια εντοπίζει παρόμοιο πρότυπο σε επόμενο καρέ. Με τον τρόπο αυτό το σύστημα συνεχόμενα εντοπίζει τα ανιχνευμένα αντικείμενα καθώς κινούνται μέσα στα στιγμιότυπα (καρέ) σε ένα βίντεο. Με άλλα λόγια, η παρακολούθηση αντικειμένων είναι η εργασία αυτόματης αναγνώρισης αντικειμένων σε ένα βίντεο και η παρακολούθηση της τροχιάς κίνησής τους με μεγάλη ακρίβεια. Συχνά, υπάρχει μια ένδειξη γύρω από το αντικείμενο που παρακολουθείται, για παράδειγμα, ένα τετράγωνο που ακολουθεί το αντικείμενο, δείχνοντας στο χρήστη πού βρίσκεται το αντικείμενο στην οθόνη. Το τετράγωνο αυτό συνήθως αναφέρεται ως bounding box(bb).

2.1.1 Χρησιμότητα και μεθοδολογίες ιχνηλάτησης αντικειμένων

Η παρακολούθηση αντικειμένων χρησιμοποιείται σε ένα μεγάλο πλήθος εφαρμογών(case studies) που περιλαμβάνουν βίντεο εισόδου διαφορετικού τύπου. Το υπό εξέταση καρέ μπορεί να προκύψει από μία εικόνα, ένα αποθηκευμένο βίντεο ή ένα βίντεο πραγματικού χρόνου. Οι τρείς αυτές περιπτώσεις εμμέσως υποδηλώνουν τους αλγόριθμους που χρησιμοποιούνται για τη δημιουργία εφαρμογών παρακολούθησης αντικειμένων. Το είδος της εισαγωγής επηρεάζει επίσης την κατηγορία, τις περιπτώσεις χρήσης και τις εφαρμογές παρακολούθησης αντικειμένων. Στην υποενότητα αυτή θα περιγραφούν εν συντομία γνωστές μεθοδολογίες και τύποι παρακολούθησης αντικειμένων, όπως παρακολούθηση βίντεο, οπτική παρακολούθηση και παρακολούθηση εικόνας.

2.1.2 Μεθοδολογίες

2.1.2.1 Παρακολούθηση βίντεο πραγματικού χρόνου

Η παρακολούθηση βίντεο πραγματικού χρόνου είναι μια εφαρμογή παρακολούθησης αντικειμένων όπου κινούμενα αντικείμενα βρίσκονται μέσα σε πληροφορίες βίντεο. Ως εκ

τούτου, τα συστήματα παρακολούθησης βίντεο είναι σε θέση να επεξεργάζονται ζωντανά, σε πραγματικό χρόνο πλάνα και επίσης εγγεγραμμένα αρχεία βίντεο.

Οι διαδικασίες που χρησιμοποιούνται για την εκτέλεση εργασιών παρακολούθησης βίντεο διαφέρουν ανάλογα με τον τύπο εισόδου βίντεο που επιλέγεται.

2.1.2.2 Οπτική ιχνηλάτηση

Η οπτική ιχνηλάτηση στόχων είναι ένα ερευνητικό θέμα της υπολογιστικής όρασης (computer vision) που υλοποιείται σε ένα μεγάλο εύρος καθημερινών εφαρμογών. Ο στόχος της οπτικής ιχνηλάτησης είναι η εκτίμηση της μελλοντικής θέσης ενός στόχου στο επόμενο καρέ χωρίς τη διαθεσιμότητα του υπόλοιπου βίντεο.

2.1.2.3 Ιχνηλάτηση εικόνας

Η παρακολούθηση εικόνας προορίζεται για τον εντοπισμό δισδιάστατων εικόνων που ενδιαφέρουν σε μια δεδομένη είσοδο. Στη συνέχεια, η εικόνα παρακολουθείται συνεχώς καθώς κινούνται στη ρύθμιση. Η ιχνηλάτηση εικόνας είναι ιδανική για εικόνες με υψηλή αντίθεση (π.χ. ασπρόμαυρες), ασύμμετρες, με λίγα pattern και με πολλαπλές διαφορές μεταξύ της εικόνας ενδιαφέροντος και του συνόλου των εικόνων μέσα στην οποία γίνεται η ιχνηλάτηση. Η τεχνική αυτή βασίζεται στην υπολογιστική όραση έτσι ώστε να εμπλουτίσει τις εικόνες με την εικόνα στόχου που έχει προκαθοριστεί.

2.1.2.4 Ιχνηλάτηση αντικειμένου από κάμερα

Οι σύγχρονες μέθοδοι ιχνηλάτησης αντικειμένων μπορούν να εφαρμοστούν σε ροή βίντεο πραγματικού χρόνου για κάθε είδος κάμερας. Επομένως, η ροή βίντεο μιας κάμερας USB ή μιας κάμερας IP μπορεί να χρησιμοποιηθεί για την εκτέλεση της ιχνηλάτησης αντικειμένων, τροφοδοτώντας τα μεμονωμένα καρέ σε έναν αλγόριθμο ιχνηλάτησης. Η παράλειψη καρέ ή η επεξεργασία των καρέ από παράλληλες διεργασίες είναι συνηθισμένες μέθοδοι για τη βελτίωση της απόδοσης ιχνηλάτησης αντικειμένων με ροή βίντεο σε πραγματικό χρόνο μίας ή πολλών πηγών εικόνας.

2.2 Αλγόριθμοι ιχνηλάτησης αντικειμένων

2.2.1 Ορισμός

Αλγόριθμος ιχνηλάτησης αντικειμένου ορίζεται ως η διαδικασία συνεχόμενης αναγνώρισης ενός αντικειμένου σε σύνολο καρέ που αποτελούν ένα βίντεο. Η διαδικασία ξεκινάει με τον προσδιορισμό του αντικειμένου (object detection) ο οποίος μπορεί να γίνει είτε από το χρήστη με την απευθείας υπόδειξη αντικειμένου στο καρέ (συνήθως με χρήση πλαισίου bounding box), είτε με αυτοματοποιημένη μέθοδο αναγνώρισης μέσω τεχνικών μηχανικής μάθησης όπου το σύστημα έχει εκπαιδευτεί να αναγνωρίσει αντικείμενα συγκεκριμένου τύπου. Στη συνέχεια ο αλγόριθμος προσπαθεί να εντοπίσει το προσδιορισμένο αντικείμενο σε κάθε ένα από τα επόμενα frame. Κάθε φορά που το αντικείμενο αναγνωρίζεται σε ένα καρέ το bounding box μεταφέρεται έτσι ώστε να το περιβάλλει. Η διαδικασία αυτή δίνει την αίσθηση στο χρήστη ότι το bounding box ακολουθεί-ιχνηλατεί το αντικείμενο στόχου.

2.2.2 Τύποι Αλγορίθμων ιχνηλάτησης αντικειμένων

2.2.2.1 BOOSTING Tracker

Ο αλγόριθμος αυτός ιχνηλάτησης βασίζεται στην έκδοση AdaBoost δηλαδή του αλγορίθμου που χρησιμοποιεί εσωτερικά ο αλγόριθμος αναζήτησης προσώπου του HAAR cascade[2].

Αυτός ο αλγόριθμος ιχνηλάτησης θεωρείται αργός με αρκετά προβλήματα στη λειτουργία του. Σήμερα κυρίως χρησιμοποιείται σε παλιές εφαρμογές και ως μέτρο σύγκρισης με άλλους αλγορίθμους ως δείκτης αναφοράς (benchmark). Ο αλγόριθμος αυτός εκπαιδεύεται κατά την εκτέλεση δημιουργώντας θετικά και αρνητικά πρότυπα του αντικειμένου. Το αρχικό πλαίσιο οριοθέτησης (bounding box), που εισάγεται από τον χρήστη ή από άλλον αλγόριθμο ιχνηλάτησης αντικειμένων, λαμβάνεται ως το πρώτο θετικό πρότυπο για το αντικείμενο και το υπόλοιπο τμήμα της εικόνας έξω από το πλαίσιο οριοθέτησης θεωρείται ως φόντο. Δεδομένου ενός νέου πλαισίου, ο αλγόριθμος εφαρμόζεται σε όλα τα γειτονικά υποσύνολα εικονοστοιχείων της αρχικής τοποθεσίας του πλαισίου ενδιαφέροντος υπολογίζοντας μία βαθμολογία ομοιότητας με το αρχικό αντικείμενο. Η νέα θέση του πλαισίου ενδιαφέροντος και κατ' επέκταση η εκτιμώμενη θέση του αντικειμένου είναι αυτή που έχει λάβει τη μέγιστη βαθμολογία. Το περιεχόμενο του πλαισίου ενδιαφέροντος αποτελεί ένα ακόμα θετικό πρότυπο για τον

αλγόριθμο. Όσο ο αλγόριθμος διαβάζει νέα καρέ, αυξάνεται το πλήθος των θετικών προτύπων με αποτέλεσμα να έχει μεγαλύτερη ικανότητα ιχνηλάτησης.

2.2.2.2Multiple Instance Learning (MIL) Tracker

Ο αλγόριθμος αυτός [2] στηρίζεται σε παρόμοια ιδέα με τον BOOSTING Tracker. Διαφοροποιείται όμως στο γεγονός ότι σε κάθε αναζήτηση του αντικειμένου σε νέο καρέ λαμβάνεται υπόψη μόνο η θέση του αντικειμένου στο προηγούμενο καρέ και αναζητείται το αντικείμενο μόνο σε κοντινά σημεία του προηγούμενου πλαισίου ενδιαφέροντος. Μειονέκτημα αυτής της μεθοδολογίας είναι ότι δημιουργεί την εντύπωση ότι αντικείμενο δεν είναι κεντραρισμένο. Το πρόβλημα αυτό αντιμετωπίζεται με την τεχνική της μάθησης πολλαπλών περιπτώσεων (Multiple Instance Learning). Στον αλγόριθμο MIL, δεν συσσωρεύονται θετικά και αρνητικά παραδείγματα, αλλά θετικά και αρνητικά σύνολα προτύπων. Η συλλογή εικόνων στο θετικό σύνολο δεν είναι όλα τα θετικά παραδείγματα όπως θα περίμενε κάποιος. Αντ 'αυτού, μόνο ένα πλαίσιο ενδιαφέροντος στο θετικό σύνολο πρέπει να είναι θετικό πρότυπο. Στην πράξη το σύνολο θετικών προτύπων περιέχει το κύριο μέρος της εικόνας του αντικειμένου αναζήτησης σε συνδυασμό με συμπληρώματα της εικόνας αυτής που γειτνιάζουν με αυτό. Ακόμα κι αν η τρέχουσα θέση του αντικειμένου που αναζητείται δεν είναι ακριβής, το γεγονός ότι δείγματα από τη γειτονιά της τρέχουσας θέσης τοποθετούνται στο σύνολο θετικών προτύπων, αυξάνει την πιθανότητα το σύνολο αυτό να περιέχει τουλάχιστον μία εικόνα στην οποία το αντικείμενο είναι όμορφα κεντραρισμένο.

2.2.2.3Kernelized Correlation Filters (KCF) Tracker

Ο αλγόριθμος ιχνηλάτησης KCF αποτελεί μία υβριδική εκδοχή των δύο tracker που αναφέρθηκαν προηγουμένως. Πιο συγκεκριμένα ο KCF εκμεταλλευόμενος τη βασική ιδέα του MIL tracker, δηλαδή τη δημιουργία συνόλου πολλαπλών θετικών προτύπων, προσπαθεί να εντοπίσει επικαλυπτόμενες περιοχές μεταξύ αυτών. Το γεγονός αυτό σε συνδυασμό και με και με τη χρήση ανώτερων μαθηματικών υπολογισμών που χρησιμοποιούνται, καθιστούν την παρακολούθηση γρηγορότερη και ακριβέστερη. Ο αλγόριθμος αυτός προτείνεται για μεγάλο εύρος γενικών περιπτώσεων καθώς έχει αποδειχτεί ότι έχει καλύτερα αποτελέσματα.

2.2.2.4Discriminative Correlation Filter with Channel and Spatial Reliability (CSRT) Tracker

Ο αλγόριθμος αυτός κατασκευάζει έναν χάρτη χωρικής αξιοπιστίας (Spatial Reliability Map) με σκοπό να χρησιμοποιηθεί αυτός ως φίλτρο στο τμήμα της επιλεγμένης περιοχής που ορίζεται από το πλαίσιο παρακολούθησης. Η μεθοδολογία αυτή επιτυγχάνει τη διεύρυνση του

πλαισίου παρακολούθησης σε συνδυασμό με τη βελτίωση της ιχνηλάτησης των μη ορθογώνιων περιοχών ή αντικειμένων. Ο αλγόριθμος κάνει χρήση μόνο δύο σταθερών χαρακτηριστικών (HoGs και Colornames). Έχει αποδειχτεί ότι ο αλγόριθμος MIL λειτουργεί αρκετά ικανοποιητικά σε βίντεο με χαμηλό ρυθμό αναπαραγωγής καρέ (π.χ. 25 fps) και τότε επιτυγχάνει μεγάλη ακρίβεια στην παρακολούθηση αντικειμένου.

2.2.2.5 MedianFlow Tracker

Ο αλγόριθμος ιχνηλάτησης MedianFlow βασίζεται στην παρακολούθηση της κίνησης στη μονάδα του χρόνου σε δύο κατευθύνσεις εμπρός και πίσω. Στη συνέχεια ο αλγόριθμος προσπαθεί να υπολογίσει τις αποκλίσεις μεταξύ αυτών των δύο τροχιών. Στόχος είναι να ελαχιστοποιηθεί η απόκλιση-σφάλμα έτσι ώστε να εκτιμάται με ακρίβεια η τροχιά και αν επιτυγχάνεται η αξιόπιστη ιχνηλάτηση του αντικειμένου. Προφανώς ο αλγόριθμος αυτός λειτουργεί καλύτερα όταν η κίνηση του αντικειμένου προς ιχνηλάτηση είναι προβλέψιμη και μικρή ανά frame. Σημαντικό πλεονέκτημα του αλγορίθμου αυτού είναι ότι μπορεί άμεσα να αντιληφθεί αποτυχία εντοπισμού του αντικειμένου στο καρέ και να σταματήσει την ιχνηλάτηση, σε αντίθεση με άλλους αλγορίθμους ιχνηλάτησης που συνεχίζουν να παρακολουθούν όταν αποτυγχάνει η παρακολούθηση.

2.2.2.6 Tracking, learning, and detection (TLD) Tracker

Ο αλγόριθμος αυτός αποτελείται από τρία βασικά δομικά μέρη, αυτό της ιχνηλάτησης, της μάθησης και της αναγνώρισης τα οποία εκτελούνται παράλληλα. Ο αλγόριθμος παρακολουθεί όλα τα πρότυπα που αναγνωρίστηκαν μέχρι εκείνη τη στιγμή και βελτιώνει τη διαδικασία της ιχνηλάτησης όταν αυτό είναι εφικτό. Παράλληλα, εκτιμώνται υφιστάμενα σφάλματα του ιχνηλάτη έτσι ώστε αυτός να εκπαιδευτεί και να είναι ικανός να αποφύγει παρόμοια σφάλματα σε επόμενα καρέ. Αυτό έχει σαν αποτέλεσμα το πλαίσιο ενδιαφέροντος να εμφανίζεται να έχει σημαντικές μετατοπίσεις μετά τη δυναμική μάθηση του ιχνηλάτη. Για παράδειγμα, όταν σε ένα καρέ αναζητούμε ένα όχημα και είναι πιθανή η ύπαρξη άλλων οχημάτων πλησίον αυτού, ο ιχνηλάτης ενδέχεται προσωρινά να ανιχνεύσει ένα άλλο όχημα μετά τη δυναμική μάθηση του ιχνηλάτη είναι πιθανή η ανίχνευση του αρχικού σωστού αντικειμένου με αποτέλεσμα την μετατόπιση του πλαισίου ενδιαφέροντος. Επίσης ο ιχνηλάτης έχει τη δυνατότητα να παρακολουθεί αντικείμενα που αλλάζουν κλίμακα και ταχύτητα κίνησης. Συνεπώς ο συγκεκριμένος ιχνηλάτης αποτελεί μία αξιόπιστη επιλογή σε περιπτώσεις που το αντικείμενο ενδιαφέροντος κρύβεται πίσω από άλλα αντικείμενα, αλλάζει κλίμακα ή ταχύτητα.

2.2.2.7Minimum Output Sum of Squared Error (MOSSE) Tracker

Ο αλγόριθμος αυτός κάνει χρήση μίας προσαρμόσιμης συσχέτισης για το αντικείμενο ενδιαφέροντος, η οποία παράγει σταθερής συσχέτισης φίλτρα όταν αρχικοποιείται χρησιμοποιώντας μόνο ένα πλαίσιο. Ο tracker αυτός δεν επηρεάζεται εύκολα από το «θόρυβο» της εικόνας ή του αντικειμένου, δηλαδή δεν επηρεάζεται από το φωτισμό, την διαφορά μεγέθους(κλίμακα), την αλλαγή της στάσης του αντικειμένου και από μη άκαμπτες παραμορφώσεις. Ανιχνεύει επίσης προσωρινή απόκρυψη του αντικειμένου κάνοντας χρήση της αναλογίας των πλάγιων αιχμών, η οποία επιτρέπει στον ιχνηλάτη να σταματήσει και να συνεχίσει από εκεί που σταμάτησε όταν επανεμφανιστεί το αντικείμενο. Ο αλγόριθμος λειτουργεί ικανοποιητικά σε υψηλή αναπαραγωγή καρέ ανά δευτερόλεπτο(fps), ακόμα και παραπάνω από 450 fps. Ο αλγόριθμος είναι εύκολα εφαρμόσιμος, εξαιρετικά ακριβής, αλλά υστερεί σε απόδοση δεδομένου ότι στηρίζεται σε τεχνικές βαθιάς μηχανικής μάθησης (deep learning).

2.2.2.8GOTURN Tracker

Ο αλγόριθμος αυτός σε αντίθεση με τους προηγούμενους στηρίζεται σε τεχνικές νευρωνικών δικτύων και πιο συγκεκριμένα σε Convolutional Neural Network (CNN). Ο αλγόριθμος αποκρίνεται επιτυχώς σε σημαντικές αλλαγές του φωτισμού και παραμορφώσεις του αντικείμενου παρακολούθησης αλλά δεν μπορεί να χειριστεί προσωρινές αποκρύψεις του αντικειμένου.

Οι παραπάνω περιγραφές των αλγορίθμων ιχνηλάτησης συνοψίζονται στον **Error! Reference source not found.** όπου παρουσιάζονται τα πλεονεκτήματα και τα μειονεκτήματα καθενός.

Πίνακας 1 Συγκριτικός πίνακας αλγορίθμων ιχνηλάτησης

Tracker	Πλεονεκτήματα	Μειονεκτήματα
BOOSTING Tracker	-	1. Η απόδοση παρακολούθησης είναι μέτρια. 2. Αδυναμία αξιόπιστης αναγνώρισης μη επιτυχημένης παρακολούθησης. 3. Ο αλγόριθμος είναι απαρχαιωμένος

MIL Tracker	1. Απόδοση πολύ καλή 2. Έχει μερική ανοχή στο σφάλμα όταν συμβαίνει μερική απόκρυψη.	1. Αδυναμία αξιόπιστης αναγνώρισης μη επιτυχημένης παρακολούθησης. 2. Αδυναμία επαναφοράς μετά από πλήρη απόκρυψη αντικειμένου.
KCF Tracker	1. Πιο αποδοτικός και πιο ακριβής σε σχέση με MIL και BOOSTING. 2. Δυνατότητα αναγνώρισης μη επιτυχημένης παρακολούθησης	Αδυναμία επαναφοράς μετά από πλήρη απόκρυψη αντικειμένου.
CSRT Tracker	Πιο ακριβής από τον KCF	Χαμηλότερη απόδοση σε σχέση με τον KCF
MedianFlow Tracker	1. Δυνατότητα αντίληψης αποτυχίας ιχνηλάτησης. 2. Αξιόπιστος όταν η τροχιά της κίνησης είναι προβλέψιμη και δεν υπάρχει απόκρυψη αντικειμένου	Αποτυγχάνει μετά από απότομη κίνηση
TLD Tracker	1. Ικανότητα ιχνηλάτησης μετά από απόκρυψη σε πολλά καρέ(frames) 2. Μεγάλη ανοχή σε αλλαγές κλίμακας αντικειμένου.	Αδυναμία ιχνηλάτησης μετά από πολλά συνεχόμενα λανθασμένα θετικά αποτελέσματα.
MOSSE Tracker	Ισχυρή απόδοση	Λιγότερο ακριβής από τους CSRT και KCF.
GOTURN Tracker	Μεγάλη ανοχή σε αλλαγές οπτικού πεδίου, φωτισμού και παραμορφώσεις σχήματος	Αδυναμία ιχνηλάτησης αντικειμένου μετά από πλήρη απόκρυψη.

2.3 Υλοποίηση αλγορίθμου σε python

Οι αλγόριθμοι ιχνηλάτησης που αναλύθηκαν στο προηγούμενο κεφάλαιο έχουν υλοποιηθεί και διάφορες πλατφόρμες και γλώσσες. Στην παρούσα διπλωματική διατριβή γίνεται χρήση της υλοποίησης που περιέχεται στη βιβλιοθήκη OpenCV η οποία αναλύεται στη συνέχεια του κεφαλαίου.

2.3.1 Βιβλιοθήκη OpenCV

Η βιβλιοθήκη OpenCV [3] [20] είναι μία τεράστια βιβλιοθήκη ανοιχτού κώδικα που ειδικεύεται σε συναρτήσεις χρήσιμες για τα πεδία της υπολογιστικής όρασης, της μηχανικής μάθησης και επεξεργασίας εικόνας. Χρησιμοποιείται ευρέως ως εργαλείο σε πραγματικού χρόνου εφαρμογές οι οποίες υλοποιούνται σε μεγάλη κλίμακα στα σημερινά πληροφοριακά συστήματα όπου το βασικό μοντέλο είναι αυτό της υπολογιστικής όρασης. Η βιβλιοθήκη αρχικά είχε υλοποιηθεί σε βελτιστοποιημένη C,C++, γεγονός που της επιτρέπει να διαχειριστεί την επεξεργασία πολλαπλών πυρήνων από έναν επεξεργαστή και από το 2008 υποστηρίζεται η διαδραστικότητα (interface) και με άλλες γλώσσες προγραμματισμού όπως Python, Ruby, Matlab, Java κτλ. Η βιβλιοθήκη OpenCV είναι συμβατή με τα ευρέως διαδεδομένα λειτουργικά συστήματα Windows, Linux, Mac OS X, iOS και Android. Η υλοποίηση της διπλωματικής διατριβής στηρίζεται στη βιβλιοθήκη OpenCV που ενσωματώνεται σε κώδικα Python και εκτελείται στο λειτουργικό σύστημα Raspbian[24]. Η χρήση της σε συνδυασμό με άλλες μαθηματικές βιβλιοθήκες όπως NumPy επαυξάνει τη μαθηματική της ικανότητα για ανάλυση. Οι σημερινές εφαρμογές που τη χρησιμοποιούν μπορούν να επεξεργαστούν βίντεο και εικόνες για αναγνώριση αντικειμένων, ακόμη και αναγνώριση της ανθρώπινης γραφής. Για να προσδιοριστεί το πρότυπο εικόνας και τα διάφορα χαρακτηριστικά του, η βιβλιοθήκη OpenCV συνήθως χρησιμοποιεί διανυσματικό χώρο και εκτελούνται μαθηματικές πράξεις σε αυτά τα χαρακτηριστικά. Στις μέρες μας χρησιμοποιείται η έκδοση 3.0. Το OpenCV κυκλοφορεί με άδεια BSD και ως εκ τούτου είναι δωρεάν τόσο για ακαδημαϊκή όσο και για εμπορική χρήση.

Υπάρχουν πολλές εφαρμογές που χρησιμοποιούν OpenCV. Μερικές είναι:

- Αναγνώριση προσώπου
- Αυτόματη ιχνηλάτηση
- Καταμέτρηση ατόμων (κυκλοφορία πεζών σε πλατεία κτλ)
- Σύνολο οχημάτων σε αυτοκινητοδρόμους καθώς και οι ταχύτητές τους
- Διαδραστικές καλλιτεχνικές εγκαταστάσεις
- Ανίχνευση ανωμαλίας σε διαδικασία παραγωγής(τα ελαττωματικά προϊόντα)

- Ραφές εικόνας με θέα στο δρόμο
- Αναζήτηση και ανάκτηση βίντεο/εικόνων
- Προσανατολισμός και καθοδήγηση ρομπότ και αυτοκινήτων χωρίς οδηγό
- Αναγνώριση αντικειμένων
- Ιατρική ανάλυση
- Ταινίες – Τρισδιάστατη δομή από κίνηση
- Αναγνώριση διαφημίσεων τηλεοπτικών καναλιών

2.3.2 Βασικές συναρτήσεις OpenCV

Η υλοποίηση της διπλωματικής εργασίας κάνει χρήση των παρακάτω χρήσιμων συναρτήσεων της βιβλιοθήκης OpenCV

- `import cv2`

Εισαγωγή βιβλιοθήκης OpenCV2 σε κώδικα python

- `cv2.TrackerCSRT_create( )`

Η συνάρτηση ορίζεται ως:

`cv.TrackerCSRT_create([, parameters]) ->retval`

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο CSRT

- `cv2.TrackerKCF_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerKCF_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο KCF

- `cv2.TrackerBoosting_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerBOOSTING_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο BOOSTING

- `cv2.TrackerMIL_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerMIL_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο MIL

- `cv2.TrackerTLD_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerTLD_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο TLD

- `cv2.TrackerMedianFlow_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerMedianFlow_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο MedianFlow

- `cv2.TrackerMOSSE_create( )`

Η συνάρτηση ορίζεται ως:

cv.TrackerMOSSE_create([, parameters]) ->retval

Με χρήση αυτής της εντολής δημιουργείται ένας ιχνηλάτης με αρχή λειτουργίας τον αλγόριθμο MOSSE

- `cv2.imshow("target", frame)`

Η συνάρτηση ορίζεται ως

```
cv.imshow(winname, mat) ->None
```

Με χρήση αυτής της εντολής δημιουργείται παράθυρο με όνομα που ορίζεται στην πρώτη παράμετρο και περιεχόμενο το frame που ορίζεται στη δεύτερη παράμετρο.

- `cv2.waitKey(1)`

Η συνάρτηση ορίζεται ως

```
cv.waitKey([, delay]) ->retval
```

Με χρήση αυτής της εντολής περιμένει το πρόγραμμα το χρήστη για χρόνο=delay να πληκτρολογήσει ένα αλφαριθμητικό το οποίο το επιστρέφει η συνάρτηση σαν retval.

- `cv2.selectROI`

Η συνάρτηση ορίζεται ως

```
cv.selectROI(windowName, img[, showCrosshair[, fromCenter]]) ->retval
```

ή

```
cv.selectROI(img[, showCrosshair[, fromCenter]]) ->retval
```

Με χρήση αυτής της εντολής το πρόγραμμα δίνει τη δυνατότητα στο χρήστη να προσδιορίσει ορθογώνιο ενδιαφέροντος (Rectangle Of Interest-ROI) με σκοπό να γίνει προσδιορισμός αντικειμένου από το χρήστη.

- `cv2.rectangle( )`

Η συνάρτηση ορίζεται ως

```
cv.rectangle(img, pt1, pt2, color[, thickness[,lineType[,shift]]])->img
```

ή

```
cv.rectangle(img, rec, color[, thickness[, lineType[, shift]]]) ->img
```

Με χρήση αυτής της εντολής το πρόγραμμα δύναται να προσθέσει ένα ορθογώνιο πάνω στο καρέ που εμφανίζεται στο χρήστη με σκοπό να είναι ορατή η ανίχνευση ή η ιχνηλάτηση του αντικειμένου.

- `cv2.putText( )`

Η συνάρτηση ορίζεται ως

```
cv.putText(img, text, org, fontFace, fontScale, color[, thickness[, line-
Type[, bottomLeftOrigin]]]) ->img
```

Με χρήση αυτής της εντολής το πρόγραμμα δύναται να προσθέσει ένα κείμενο πάνω στο καρέ που εμφανίζεται στο χρήστη με σκοπό να εμφανίσει πληροφορίες όπως αν υπήρξε επιτυχής αναγνώριση του αντικειμένου στόχου την τιμή του fps και άλλα.

- `cv2.destroyAllWindows( )`

Η συνάρτηση ορίζεται ως

```
cv.destroyAllWindows() ->None
```

Με χρήση αυτής της εντολής το πρόγραμμα κλείνει όλα τα παράθυρα που έχει ανοίξει το ίδιο

ΚΕΦΑΛΑΙΟ 3
Χειρισμός κίνησης κάμερας μέσω gimbal

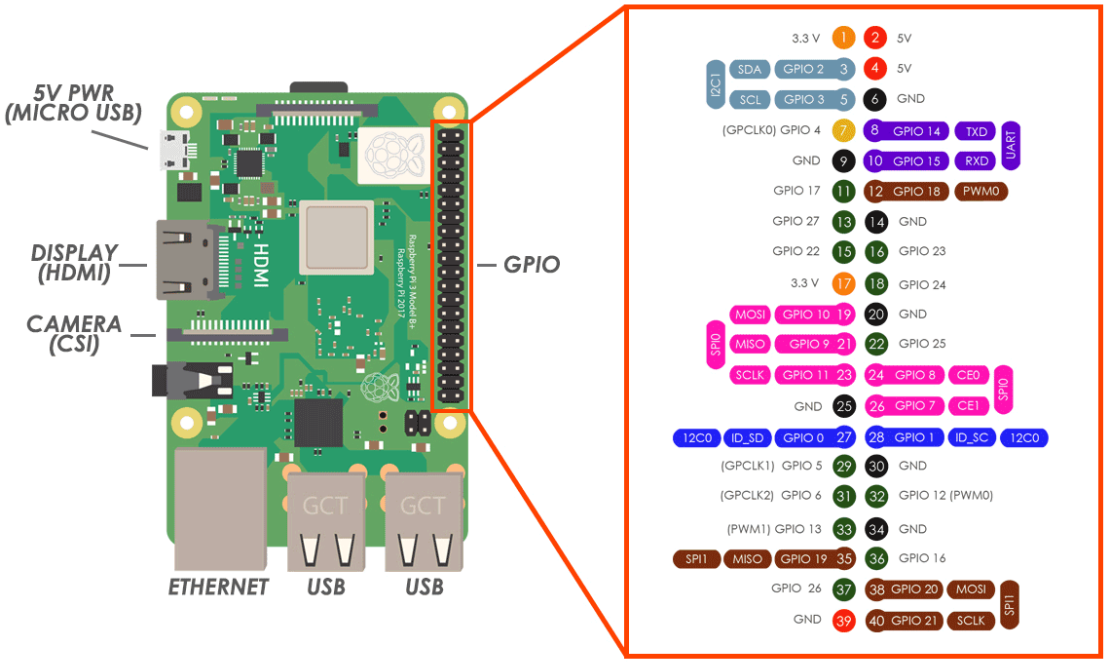
3.1 Raspberry pi 4

Το Raspberry Pi [4] [21] είναι ένα project το οποίο ξεκίνησε το 2008 με σκοπό να προσφέρει στην αγορά ένα ενσωματωμένο σύστημα με μικρές διαστάσεις το οποίο όμως θα δίνει στο χρήστη όλες τις δυνατότητες ενός συμβατικού υπολογιστή με εξαιρετικά χαμηλό κόστος.

Εικόνα 2 Raspberry Pi 4 Model B [28]

Πιο συγκεκριμένα το σύστημα αυτό διαθέτει θύρες USB, micro HDMI, Ethernet και υποδοχής κάμερας (CSI) με αποτέλεσμα να έχει τη δυνατότητα να δεχθεί είσοδο από USB πληκτρολόγιο, ποντίκι ή οποιαδήποτε άλλη συσκευή εισόδου εξόδου, να δώσει έξοδο σε οθόνη και να συνδεθεί στο διαδίκτυο. Επιπρόσθετα διαθέτει ένα σύνολο από 40 pins γενικού σκοπού (General Purpose Input Output-GPIO) τα οποία αυξάνουν τα πεδία εφαρμογής του συστήματος αυτού αφού μπορεί να δεχθεί πλήθος ψηφιακών συσκευών-αισθητήρων. Αξίζει να σημειωθεί ότι το raspberry Pi τροφοδοτείται εύκολα από θύρα Micro USB αφού χρειάζεται 5 volt για να λειτουργήσει γεγονός που επιτρέπει στο χρήστη να το τροφοδοτήσει ακόμα και από τον προσωπικό του υπολογιστή ή από φορτιστή «έξυπνης» συσκευής.

Οι θύρες και τα pins του Raspberry Pi 4 φαίνονται αναλυτικά στην Εικόνα 3



Εικόνα 3 Ανάλυση διεπαφών Raspberry Pi 4[29]

3.1.1 Ιστορική αναδρομή

Το Raspberry Pi αρχικά δημιουργήθηκε με σκοπό να τροφοδοτήσει μεγάλο πλήθος χρηστών, που δεν έχουν την οικονομική δυνατότητα αγοράς προσωπικού υπολογιστή, με έναν υπολογιστή χαμηλού κόστους(≈40€). Παράλληλα το Raspbery Pi άμεσα χρησιμοποιήθηκε και για εκπαιδευτικούς σκοπούς έτσι ώστε να διδαχθεί στα σχολεία η επιστήμη των υπολογιστών και των ενσωματωμένων συστημάτων. Το συγκεκριμένο ενσωματωμένο σύστημα έγινε πιο διάσημο από ό,τι αναμενόταν και ξεκίνησε να χρησιμοποιείται σε διάφορους τομείς, όπως ρομποτική, αναγνώρισης αντικειμένων, ιχνηλάτηση αντικειμένων, προγραμματισμό, πρόγνωση καιρού, συστήματα συναγερμού κτλ, λόγω χαμηλού κόστους, εύκολης παραμετροποίησης και ανοιχτού σχεδιασμού.

Από το 2006 όταν και βγήκε στην αγορά η πρώτη έκδοση του Raspberry Pi έχουν ακολουθήσει πολλές εκδόσεις με διάφορες τροποποιήσεις αναφορικά με τον επεξεργαστή, τη μνήμη RAM και το πλήθος των διεπαφών που διαθέτει. Οι πιο σημαντικές εκδόσεις αναφέρονται παρακάτω:

- Raspberry Pi Pico με system on a chip (SOC) βασισμένο σε RP2040 συμβατό με ARM αρχιτεκτονική.
- Model A
- Model B
- Model A+ με CPU ARM11
- Model B+ με CPU ARM11
- Raspberry Pi 2 με quad-core CPU ARM Cortex-A7 900 MHz των 32-bit και 1 GB μνήμη RAM.
- Raspberry Pi Zero το οποίο έχει μικρότερες διαστάσεις και λιγότερες δυνατότητες Input/Output(I/O) το οποίο όμως διαθέτει interface γενικού σκοπού general purpose input/output (GPIO)
- Raspberry Pi Zero W, με ασύρματη κάρτα δικτύου Wi-Fi και Bluetooth.
- Raspberry Pi 3 Model B με quad-core 1,2 GHz αρχιτεκτονικής ARM Cortex-A53 64-bit.
- Το Raspberry Pi 4 Model B κυκλοφόρησε τον Ιούνιο του 2019 με quad-core 1,5 GHz τεχνολογίας ARM Cortex-A72 και 64-bit, αύσρματη κάρτα δικτύου Wi-Fi με πρωτόκολλο 802.11ac, Bluetooth 5

3.1.2 Υλικό προτεινόμενου συστήματος

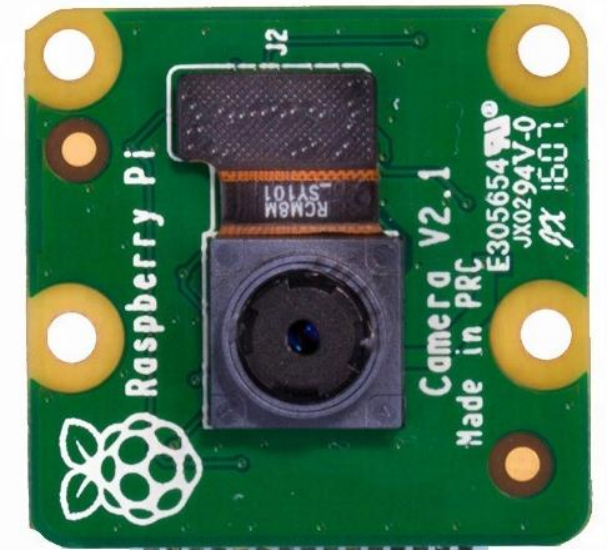
Στην παρούσα εργασία το σύστημα αποτελείται από:

- Ένα Raspberry Pi 4 model B

Όπως φαίνεται στην Εικόνα 2

- Μία Raspberry Pi Camera Module V2 (8MP,1080p)

Η Raspberry Pi Camera Module V2 (βλ. Εικόνα 4) είναι μία νέα και βελτιωμένη κάμερα από το Raspberry Pi Foundation. Σχεδιασμένη και κατασκευασμένη στο Ηνωμένο Βασίλειο, η Raspberry Pi Camera Board V2 διαθέτει αισθητήρα εικόνας Sony IMX219 εξαιρετικά υψηλής ποιότητας 8 megapixel (από 5MP που ήταν στην V1) και σταθερό φακό κάμερας. Η μονάδα κάμερας V2 είναι ικανή για στατικές εικόνες 3280 x 2464 pixel και υποστηρίζει επίσης βίντεο 1080p με 30 fps, 720p με 60 fps και 640x480p με 90 fps.



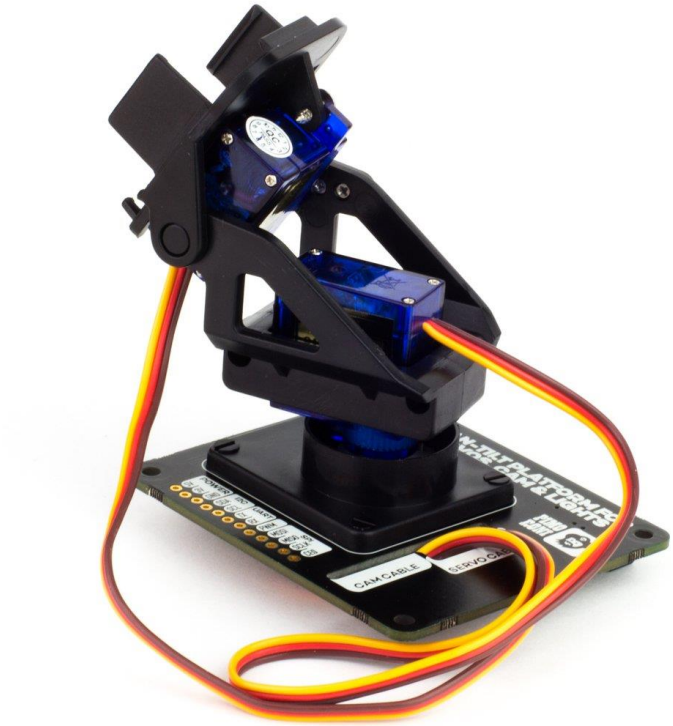
Εικόνα 4 raspberry-pi-camera-module-v2-8mp-1080p[30]

Η μονάδα προσαρμόζεται στο Raspberry Pi μέσω μίας ταινίας καλωδίου 15 ακίδων (βλ Εικόνα 5), στην ειδική σειριακή διεπαφή κάμερας 15 ακίδων MIPI (CSI), η οποία σχεδιάστηκε ειδικά για διασύνδεση με κάμερες. Ο δίαυλος CSI είναι ικανός για εξαιρετικά υψηλό ρυθμό μετάδοσης δεδομένων και μεταφέρει αποκλειστικά δεδομένα pixel στον ε-πεξεργαστή. Η πλακέτα στην οποία είναι προσκολλημένη η κάμερα είναι μικρή, περίπου 25mm x 23mm x 9mm, ζυγίζει περίπου 3gr, καθιστώντας την ιδανική πολλές εφαρμογές που απαιτούν μικρό βάρος όπως π.χ. τα drones.

Εικόνα 5 Καλωδιοταινία (flex cable)[31]

- Ένα Pimoroni Pan-Tilt HAT - Full kit (gimbal)

Το Pimoroni Pan-Tilt HAT [5] (βλ. Εικόνα 6) είναι ένα gimbal δύο αξόνων το οποίο τοποθετείται πάνω στο Raspberry Pi. Στο συγκεκριμένο σύστημα πάνω από την armor case. Είναι κατασκευασμένο ώστε να κινούνται ανεξάρτητα οι σερβοκινητήρες (servos). Υπάρχει μία οπή ώστε να διέρχονται από μέσα τα καλώδια των servos, της κάμερας και του LED. Δεν απαιτείται κόλληση, αφού οι ακροδέκτες των καλωδίων των servos και του LED είναι θηλυκοί. Στην κορυφή του μπορεί να τοποθετηθεί η κάμερα αφού πρώτα συνδεθεί στο RaspberryPi. Το κάθε servo είναι ικανό να περιστραφεί συνολικά 180^{o}, 90^{o} έως -90^{o} από τον άξονά του. Έχει δύο κανάλια (channels) για τα servos και ένα PWM LED channel. Είναι συμβατό με όλα τα μοντέλα Raspberry Pi που έχουν 40 header pins.



Εικόνα 6 Pan-Tilt_HAT[32]

- Μία θήκη προστασίας με αερόψυξη (Armor Case for Raspberry Pi 4 B with Dual Fan)

Η Armor Case for Raspberry Pi 4 B with Dual Fan [6] (βλ. Εικόνα 7) είναι κατασκευασμένη από αλουμίνιο προστατεύοντας το Raspberry από υπερθέρμανση και χτυπήματα. Οπότε επιτρέπει στο σύστημα να χρησιμοποιηθεί σε δύσκολες συνθήκες. Επίσης, δεδομένου ότι οι διαστάσεις του Armor Case for Raspberry Pi 4 B with Dual Fan (μήκος και πλάτος) δεν ξεπερνούν τις διαστάσεις της πλακέτας Raspeberry Pi , επιτρέπουν στο συνολικό σύστημα να χρησιμοποιηθεί σε εφαρμογές που απαιτούν χώρο παρόμοιο με αυτό ενός απλού raspberry. Περιέχει δύο μικρούς ανεμιστήρες (fans) για να προσδίδει περισσότερη ψύξη στην πλακέτα. Επιτρέπει να χρησιμοποιηθούν όλες οι υποδοχές του Raspberry Pi όταν η θήκη προστασίας είναι προσαρμοσμένη στην πλακέτα, οπότε αποφεύγεται η αποσυναρμολόγηση και επανασυναρμολόγηση στις περιπτώσεις αλλαγής SD card, καλωδίων USB και καλωδίων GPIO.

Εικόνα 7 Armor Case for Raspberry Pi 4 B with Dual Fan[33]

- Ένα σετ από pin (GPIO Header for Raspberry Pi 2x20 Stackable) (βλ. Εικόνα 8)για προέκταση των pin του Raspberry ώστε να μπορεί να ενωθεί με το gimbal.

Εικόνα 8 2x20 Header pins[34]

3.2 Οδήγηση κάμερας

Η οδήγηση της κάμερας γίνεται με χρήση τεσσάρων παράλληλων διεργασιών, συνεπώς για διευκόλυνση του αναγνώστη θα αναλυθούν αρχικά βασικές έννοιες δημιουργίας και επικοινωνίας διεργασιών σε λειτουργικό σύστημα Raspbian με χρήση γλώσσας προγραμματισμού Python.

3.2.1 Διεργασίες και επικοινωνία αυτών

Μια διεργασία είναι ένα τμήμα ενός προγράμματος υπολογιστή που εκτελείται από ένα ή πολλά νήματα(threads). Αποτελείται από τον κώδικα του προγράμματος και συγκεκριμένο χώρο μνήμης. Ανάλογα με το λειτουργικό σύστημα (OS), μια διεργασία μπορεί να αποτελείται από πολλά threads που εκτελούνται ταυτόχρονα. Μία διεργασία όταν δημιουργείται αποκτά έναν μοναδικό αριθμό που ονομάζεται αριθμός διεργασίας (process identifier - PID) Στο λειτουργικό σύστημα linux είναι ένας ακέραιος μεταξύ του 2 και του 32.768 καθώς ο αριθμός 1 είναι για την διεργασία init που διαχειρίζεται άλλες διεργασίες. Όταν τρέχει ένα πρόγραμμα, ο κώδικας που εκτελείται μεταφέρεται στη μνήμη RAM. Όταν εκτελούνται δύο ή περισσότερες διεργασίες, υπάρχουν μεταβλητές μοναδικές για την κάθε διεργασία, οι οποίες είναι αποθηκευμένες σε διαφορετικό χώρο δεδομένων της κάθε διεργασία και συνήθως ο χώρος μνήμης δεν είναι μοιραζόμενος(shared). Δηλαδή μία

διεργασία χρησιμοποιεί το δικό της χώρο μνήμης για τις μεταβλητές της. Όταν μία διεργασία δημιουργεί μία άλλη διεργασία, η νέα διεργασία ονομάζεται παιδί (child process) και η αρχική ονομάζεται πατέρας (parent process).

3.2.1.1 Δημιουργία παράλληλων διεργασιών

Ενώ ένα πρόγραμμα υπολογιστή είναι ένα σύνολο εντολών που είναι συντακτικά ορθό, έχει μεταγλωττιστεί και έχει αποθηκευτεί στον δίσκο ως εκτελέσιμο αρχείο. Σε αντίθεση μία διεργασία είναι ένα πρόγραμμα που εκτελείται και θα χρειαστεί πόρους του συστήματος όπως CPU, RAM κτλ. Η διεργασία λοιπόν φορτώνεται από το δίσκο στη μνήμη έτσι ώστε να χρονοδρομολογηθεί και να εκτελούνται οι εντολές της στη CPU. Διάφορες διεργασίες μπορεί να σχετίζονται με το ίδιο πρόγραμμα. Για παράδειγμα, το άνοιγμα πολλών εμφανίσεων του ίδιου προγράμματος συχνά έχει ως αποτέλεσμα περισσότερες από μία διεργασίες να εκτελούνται. Το Multitasking είναι μια μέθοδος που επιτρέπει σε πολλές διεργασίες να μοιράζονται την CPU και άλλους πόρους του συστήματος. Κάθε πυρήνας του επεξεργαστή εκτελεί μια εργασία(task) κάθε φορά. Ωστόσο, η εκτέλεση πολλαπλών εργασιών επιτρέπει σε κάθε επεξεργαστή να κάνει εναλλαγή μεταξύ εργασιών (context switching) που εκτελούνται χωρίς να χρειάζεται να περιμένει να τελειώσει κάθε εργασία. Ανάλογα με την εφαρμογή του λειτουργικού συστήματος, οι διακοπές του (interrupts) θα μπορούσαν να εκτελεστούν όταν οι εργασίες ξεκινούν να περιμένουν την ολοκλήρωση των λειτουργιών εισόδου/εξόδου, όταν μια διεργασία αποδίδει οικειοθελώς την CPU, σε διακοπές υλικού και όταν το λειτουργικό σύστημα αποφασίσει ότι μια διεργασία έχει λήξει ο χρόνος που της δόθηκε από το CPU[23]. Στην python ένα παράδειγμα δημιουργίας παράλληλων διεργασιών είναι το παρακάτω που φαίνεται στο Τμήμα κώδικα 1.

Τμήμα κώδικα 1 Παράδειγμα δημιουργίας διεργασιών

```
1   # importing the multiprocessing module
2   import multiprocessing
3
4   def print_cube(num):
5     #function to print cube of given num
6     print("Cube: {}".format(num * num * num))
7
8   def print_square(num):
9     #function to print square of given num
10    print("Square: {}".format(num * num))
11
12  if __name__ == "__main__":
13    # creating processes
14    p1=multiprocessing.Process(target=print_square, args=(10, ))
15    p2=multiprocessing.Process(target=print_cube, args=(10, ))
16
17    # starting process 1
18    p1.start()
19    # starting process 2
20    p2.start()
21
22    # wait until process 1 is finished
23    p1.join()
24    # wait until process 2 is finished
25    p2.join()
26
27    # both processes finished
28    print("Done!")
```

Στον παραπάνω κώδικα η διεργασία πατέρας δημιουργεί δύο διεργασίες παιδιά οι οποίες εκτελούνται παράλληλα με τη διεργασία πατέρα και έχουν ως σκοπό την εκτύπωση του εμβαδού και του κύβου αντίστοιχα δεδομένης μίας πλευράς.

Στη γλώσσα προγραμματισμού python η δημιουργία διεργασιών επιτυγχάνεται μέσω της βιβλιοθήκης `multiprocessing`, ακριβώς για αυτό το λόγο στη γραμμή 2 γίνεται `import` η βιβλιοθήκη `multiprocessing`.

Στις γραμμές 4-7 και 8-11 ορίζονται οι συναρτήσεις print_cube και print_square που θα δηλωθούν ως σώμα των διεργασιών παιδιά. Στις γραμμές 14 και 15 με την κλήση συστήματος `multiprocessing.Process` δημιουργούμε τις δύο διεργασίες παιδιά οι οποίες θα εκτελέσουν από μία συνάρτηση όπως αυτή ορίζεται στην παράμετρο `target` με αρχικό όρισμα 10 όπως αυτό ορίζεται στην παράμετρο args. Οι διεργασίες αυτές θα εκτελεστούν παράλληλα όταν εκκινήσουν, το οποίο συμβαίνει στις γραμμές 18 και 20 όπου γίνεται η έναρξη των διεργασιών, στην γραμμή 23 και 25 η διεργασία πατέρας περιμένει να τερματίσουν οι διεργασίες παιδιά του.

3.2.1.2 Επικοινωνία διεργασιών μέσω signals

Οι διεργασίες που έχουν δημιουργηθεί και εκτελούνται σε ένα υπολογιστικό σύστημα όπως αναφέρθηκε προηγουμένως δεν μοιράζονται χώρο μνήμης. Παρόλα αυτά τα λειτουργικά συστήματα μας προσφέρουν διάφορους τρόπους επικοινωνίας των διεργασιών, όπως τα σήματα, οι διοχετεύσεις (pipes) και τα socket. Το σήμα (signal) είναι μία διαδικασία επικοινωνίας της διεργασίας με άλλες διεργασίες. Όταν στέλνεται ένα signal, το λειτουργικό σύστημα διακόπτει τη λειτουργία της διεργασίας για να διαχειριστεί το χειριστή του σήματος (signal handler). Κάθε λειτουργικό σύστημα έχει ένα εύρος σημάτων και κάθε σήμα αναγνωρίζεται από μία ακέραιη τιμή όπως φαίνεται στον Πίνακας 2.

Πίνακας 2 Σήματα λειτουργικού συστήματος linux

Σήμα	Τιμή	Ενέργεια	Περιγραφή
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference

SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

Μία απλή χρήση σημάτων φαίνεται στο παρακάτω Τμήμα κώδικα 2.

Τμήμα κώδικα 2 Παράδειγμα χρήσης σημάτων

```
    def signal_handler(sig, frame):
        #print a status message
        print("[INFO] You pressed 'ctrl+c'!Exiting...")
        time.sleep(10)
        #disable the srvos
        pth.servo_enable(1, False)
        pth.servo_enable(2, False)

        #exit
        sys.exit()


    def obj_center(args, tracker, objX, objY, centerX, centerY):
        #signal trap to handle keyboard interrupt
        signal.signal(signal.SIGINT, signal_handler)
```

Ο κώδικας στο Τμήμα κώδικα 2 έχει σκοπό να ειδοποιήσει μία διεργασία να σταματάει την εκτέλεσή της και να εκτελεί τη συνάρτηση signal_handler(sig, frame) όταν ο χρήστης πατήσει «ctrl+c». Δεδομένου ότι το πάτημα «ctrl+c» έχει ως αποτέλεσμα το

λειτουργικό σύστημα να στέλνει στη διεργασία το σήμα SIGINT, στη γραμμή 15 γίνεται η κληση συστήματος signal η οποία δένει το σήμα SIGINT με τον χειριστή σήματος (handler) signal_handler. Από το σημείο αυτό και μετά η διεργασία με κώδικα τη συνάρτηση obj_center οποτεδήποτε λάβει σήμα SIGINT, δηλαδή ο χρήστης πατάει το «ctrl+c» η διεργασία θα εκτελεί τη συνάρτηση signal_handler. Πιο συγκεκριμένα, ο χειριστής δέχεται δύο ορίσματα, δηλαδή την αρίθμηση του σήματος από το λειτουργικό σύστημα και σαν δεύτερο όρισμα ένα πλαίσιο. Εκτυπώνει στην οθόνη με την συνάρτηση print ότι πληκτρολογήθηκε από τον χρήστη το «ctrl+c» (γραμμή 3). Στη συνέχεια, με χρήση της συνάρτησης time.sleep(10) (γραμμή 4) η διεργασία αδρανοποιείται για δέκα δευτερόλεπτα. Μετά την αδρανοποίηση η διεργασία με την κλήση των συναρτήσεων pth.servo_enable(1, False) και pth.servo_enable(2, False) (γραμμές 6,7) απενεργοποιεί τους δύο σερβοκινητήρες και τερματίζει στη γραμμή 10.

3.2.2 Οδήγηση κάμερας συγκεκριμένου αντικειμένου ενδιαφέροντος

Στην ενότητα αυτή θα παρουσιαστεί ο κώδικας (βλ. Τμήμα κώδικα 3)με χρήση του οποίου επιτυγχάνεται η οδήγηση της κάμερας για συγκεκριμένο αντικείμενο ενδιαφέροντος που προσδιορίζεται από τον χρήστη μέσω του ποντικιού.

Στον κώδικα αυτόν αρχικά γίνονται import τα βασικά πακέτα της python που θα μας χρειαστούν με τα πιο βασικά να είναι τα computer vision(cv2) και image utilities(imutils). Η εισαγωγή των πακέτων αυτών γίνονται στις γραμμές 2-17.

Στις γραμμές 19-22 ορίζονται οι μεταβλητές που θα αποθηκεύουν τα PID των διεργασιών που θα δημιουργηθούν στην κύρια συνάρτηση (main). Πιο συγκεκριμένα θα δημιουργηθούν τέσσερις διεργασίες παιδιά της κύριας που θα εκτελούνται παράλληλα και θα έχουν ως έργο τα εξής: i) Προσδιορισμός κέντρου frame και κέντρου πλαισίου ενδιαφέροντος (processObjectCenter), ii) Υπολογισμός εκτιμώμενης pan-κίνησης του gimbal της κάμερας(processPanning) iii) Υπολογισμός εκτιμώμενης tilt-κίνησης του gimbal της κάμερας(processTilting) και iv) Δημιουργία εντολών προς τους σερβοκινητήρες του gimbal (processSetServos). Το σώμα των διεργασιών αυτών περιγράφεται αναλυτικά στη συνέχεια του κεφαλαίου στην αντίστοιχη για κάθε διεργασία συνάρτηση.

Στις γραμμές 26-119 ορίζεται η συνάρτηση Center_Detection που θα αποτελέσει σώμα μίας από τις τέσσερις παράλληλες διεργασίες που θα δημιουργήσει το σύστημα. Αρχικά στις γραμμές 30-39 ελέγχεται η τιμή στον δείκτη "vFile" στο dictionary args το οποίο η συνάρτηση έχει δεχτεί ως πρώτο όρισμα. Η τιμή του δείκτη αυτή μας προσδιορίζει αν γίνεται η εισαγωγή της εικόνας από την PiCamera ή από αποθηκευμένο βίντεο. Αν η τιμή του δείκτη αυτού είναι κενή σημαίνει ότι θα πάρουμε την εικόνα από την PiCamera,

για το λόγο αυτό στη γραμμή 32 εκκινούμε το videostreaming από την PiCamera και αποθηκεύουμε τον προσδιοριστή της πηγής εικόνας στη μεταβλητή Video_Source. Η εντολή αυτή συνοδεύεται από μία καθυστέρηση χρόνου ,ο οποίος έχει προσδιοριστεί στο τμήμα κώδικα(constants), (γραμμή 33) για να δοθεί χρόνος στην κάμερα να ενεργοποιηθεί. Να σημειωθεί εδώ ότι για χάρη οργάνωσης του κώδικα όλες οι βασικές παράμετροι του συστήματος έχουν συγκεντρωθεί σε ένα αρχείο, αυτό με όνομα constants.py στο οποίο ο χρήστης μπορεί εύκολα να βρει και να μεταβάλλει την τιμή κρίσιμων ορισμάτων παραμετροποιώντας με αυτόν τον τρόπο το σύστημα σύμφωνα με τις δικές του ανάγκες. Το αρχείο αυτό φαίνεται στο Τμήμα κώδικα 5 και έχει γίνει ενσωμάτωση στον τρέχων κώδικα ώς con στη γραμμή 15 και θα εξηγηθεί στη συνέχεια. Συνεχίζοντας την επεξήγηση της συνάρτησης Center_Detection, αν η τιμή του δείκτη βίντεο δεν είναι κενή (γραμμή 34) σημαίνει ότι μας έχει περάσει ως όρισμα το όνομα αρχείου αποθηκευμένου βίντεο το οποίο εκκινούμε στη γραμμή 36 αρχικοποιώντας αντίστοιχα τον προσδιοριστή της πηγής εικόνας στη μεταβλητή Video_Source. Και στην περίπτωση αυτή υπάρχει αντίστοιχη καθυστέρηση χρόνου, που έχει ομοίως προσδιοριστεί στο τμήμα κώδικα(constants), (γραμμή 37). Σε κάθε περίπτωση εμφανίζεται στο χρήστη αντίστοιχο μήνυμα (γραμμή 31 ή γραμμή 35). Στη συνέχεια αρχικοποιούμε τις μεταβλητές Initial_Bounding_Box, Frames_Per_Second σε κενές (γραμμές 38, 39). Η μεταβλητή Initial_Bounding_Box θα αποθηκεύει το bounding box του αντικειμένου ενώ η μεταβλητή Frames_Per_Second θα αποθηκεύει την τρέχουσα τιμή των frames per second.

Στη γραμμή 42-64 διαβάζονται σειριακά σε ένα βρόχο while ένα-ένα τα frames είτε από την κάμερα (γραμμή 43 και 48) είτε από το αρχείο (γραμμή 43 και 46). Στην περίπτωση που το frame διαβάζεται από την κάμερα χρειάζεται μία περιστροφή της κατά 180^{o} εξαιτίας του τρόπου τοποθέτησής της στο gimbal. Η περιστροφή αυτή γίνεται με τη χρήση της εντολής flip. Στην περίπτωση που δεν διαβαστεί frame ο βρόχος σπάει (γραμμή 49,50) δεδομένης αδυναμίας του συστήματος να διαβάσει frame είτε από την κάμερα είτε από το αρχείο. Δεδομένης επιτυχούς ανάγνωσης ενός frame είτε από την κάμερα είτε από το αρχείο γίνεται σε αυτό scaling σε πλάτος 640 (γραμμή 51) και υπολογίζονται οι νέες διαστάσεις του frame που αποθηκεύονται στις μεταβλητές (Height, Weight) ώς tuple (γραμμή 52). Ο χρήστης μπορεί να επιλέξει άλλη τιμή διαφορετική από 640 αν το επιθυμεί τροποποιώντας την αντίστοιχη παράμετρο στο αρχείο con Στη συνέχεια τα ορίσματα Frame_Center_X.value, Frame_Center_Y.value λαμβάνουν τιμές ίσες με το μισό πλάτος και μισό ύψος αντίστοιχα(γραμμές 53,54). Στο σημείο αυτό το frame εμφανίζεται στο χρήστη με την εντολή imshow του πακέτου cv2 (γραμμή 55) και περιμένουμε από το χρήστη να πατήσει το πλήκτρο "s" (select) γραμμή 56,58. Αν συμβεί αυτό η συνάρτηση selectROI (select Rectangle Of Interest) του πακέτου cv2 δίνει τη δυνατότητα στο χρήστη να σχεδιάσει με το ποντίκι του ένα ορθογώνιο πάνω στο frame για να μας υποδείξει

το αντικείμενο ενδιαφέροντος (γραμμή 59). Τότε δημιουργείται ένα αντικείμενο (obj1) της κλάσης objCenter (βλ. Τμήμα κώδικα 4: θα εξηγηθεί παρακάτω). Μέσω του αντικειμένου αυτού δημιουργούμε έναν object Tracker όπως μας δηλώνει το δεύτερο όρισμα της συνάρτησης Center_Detection. Ο tracker αυτός αποθηκεύεται στον περιγραφητή selectedTracker ο οποίος αρχικοποιείται στη γραμμή 62 με σκοπό να μας αρχικοποιήσει και την μεταβλητή Initial_Bounding_Box γεγονός που δεν θα μας ξαναβάλει στο βρόχο while των γραμμών 42-64(βλ συνθήκη while, γραμμή 42). Στη συνέχεια εκκινεί η μέτρηση των frames per second και αρχικοποιείται η μεταβλητή Frames_Per_Second. Αν λοιπόν ο χρήστης σε κάποιο frame πατήσει "s" και διαλέξει αντικείμενο με το ποντίκι του θα δημιουργηθεί ένας tracker και θα αρχικοποιηθούν οι μεταβλητές Initial_Bounding_Box και Frames_Per_Second και το πρόγραμμα θα οδηγηθεί στο βρόχο επανάληψης while των γραμμών 72-121.

Στο βρόχο των γραμμών 72-121, στο σημείο 72-82 το πρόγραμμα εξακολουθεί να διαβάζει επαναληπτικά frames είτε από το βίντεο είτε από την κάμερα, τους αλλάζει το μέγεθος όπως πριν (γραμμή 85) και ενημερώνει τις μεταβλητές Height, Width, Frame_Center_X.value, Frame_Center_Y.value (γραμμές 86-88). Στη συνέχεια ζητάει από τον tracker να γίνει update και να αναγνωρίσει το αντικείμενο στο νέο frame. (γραμμή 90,91). Αν αυτό γίνει με επιτυχία οι συντεταγμένες του αντικειμένου (x, y, w, h), όπου x,y οι συντεταγμένες της πάνω αριστερής γωνίας στο ορθογώνιο σύστημα αξόνων και w,h το πλάτος και ύψος του ορθογωνίου, στο νέο frame επιστρέφονται από τον tracker στην tuple Target_Box (γραμμή 91). Αν το Target_Box δεν είναι κενό (γραμμή 92) δημιουργείται ένα ορθογώνιο στις διαστάσεις του Target_Box και τοποθετείται μέσα στο frame(γραμμή 93,94) για να δοθεί η ψευδαίσθηση στον χρήστη ότι το ορθογώνιο ιχνηλατεί το αντικείμενο. Στη συνέχεια η μεταβλητή Frames_Per_Second γίνεται update και σταματάει μέχρι να διαβαστεί το νέο frame. Προς καλύτερη ενημέρωση του χρήστη δημιουργείται (γραμμές 101-106) ένα dictionary πληροφοριών το SCREEN_MESSAGE το οποίο περιέχει πληροφορίες αναφορικά με τον τύπο του tracker, με επιτυχή αναγνώριση ή όχι και τα fps, το οποίο προστίθεται στο frame (γραμμές 111-114) πριν αυτό εμφανιστεί (γραμμή 117). Η διαδικασία αυτή (βρόχος while γραμμών 72-121) επαναλαμβάνεται έως ότου ο χρήστης πατήσει «ctrl+c» όταν και ο χειριστής του σήματος θα τερματίσει τη διεργασία (γραμμή 181).

Στις γραμμές 125-141 ορίζεται η συνάρτηση set_servos που θα αποτελέσει σώμα της τέταρτης παράλληλης διεργασίας. Πιο συγκεκριμένα, επαναληπτικά (βρόχος while γραμμές 128-140) υπολογίζονται οι τιμές των γωνιών οριζόντιας και κάθετης μετακίνησης (γραμμές 130-131) δεδομένων των παραμέτρων pan και tlt που δέχεται η συνάρτηση και αν οι γωνίες αυτές είναι εντός ορίων, όπως τα όρια αυτά ορίστηκαν στην tuple Servo_Angle_Range του αρχείου constants.py, μέσω της συνάρτησης Range_Validation

καλεί την αντίστοιχη συνάρτηση ενημέρωσης των σερβοκινητήρων (γραμμές 135 και 139) που ανήκουν στο πακέτο pantilthat. Η συνάρτηση Range_Validation επιστρέφει μία Boolean μεταβλητή η οποία είναι true αν το πρώτο όρισμα είναι μεταξύ του δεύτερου και του τρίτου (con.servoRange[0],con.servoRange[1]), διαφορετικά επιστρέφει false. Τα εύρη τιμών λειτουργίας των σερβοκινητήρων μπορεί ο χρήστης να τα μεταβάλλει στο αρχείο constants.py.

Στις γραμμές 143-156 ορίζεται η συνάρτηση Pid_Parameters_Calculation που θα αποτελέσει σώμα της δεύτερης και τρίτης παράλληλης διεργασίας. Πιο συγκεκριμένα δημιουργείται και αρχικοποιείται (γραμμές 145,146) ένα αντικείμενο της κλάσης PID (βλ αρχείο κώδικα pid.py). Το αντικείμενο αυτό έχει ως στόχο τον συνεχόμενο (βρόχος while γραμμές 149-155) υπολογισμό του σφάλματος και των PID παραμέτρων που θα χρησιμοποιηθούν στην οδήγηση των σερβοκινητήρων του gimbal που γίνεται από τις δύο αυτές διεργασίες. Η συνάρτηση αυτή όπως αναφέρθηκε αποτελεί σώμα για δύο διεργασίες η διαφορετική λειτουργία της όμως για κάθε διεργασία προσδιορίζεται μέσω της πρώτης παραμέτρου output όπως φαίνεται στις γραμμές 226, 227 όπου η μία διεργασία καλείται με πρώτο argument pan ενώ η δεύτερη καλείται με πρώτο argument tlt.

Στις γραμμές 165-172 ορίζεται η συνάρτηση kill_child_processes η οποία δέχεται δύο ορίσματα. Το πρώτο όρισμα είναι ένας ακέραιος αριθμός signum και πιο συγκεκριμένα είναι ο SIGINT που είναι ο αριθμός σήματος όταν ο χρήστης πατήσει στο πληκτρολόγιο «ctrl+c». Το δεύτερο όρισμα είναι το frame, δηλαδή σε ποιο frame πατήθηκε το «ctrl+c». Στις γραμμές 166,167 εκτελούνται οι συναρτήσεις servo_enable του πακέτου pth που απενεργοποιούν τους δύο σερβοκινητήρες αφού στο δεύτερο όρισμά τους έχουν την Boolean τιμή false. Στις γραμμές 168-171 έχουμε τον τερματισμό των τεσσάρων διεργασιών και στη γραμμή 172 έχουμε τερματισμό της κύριας διεργασίας.

Αφού έχουν δηλωθεί αυτές οι συναρτήσεις που θα αποτελέσουν σώμα των τεσσάρων διεργασιών ορίζεται η main στη γραμμή 179-244 που θα δημιουργήσει τις διεργασίες αυτές. Στην main αρχικά ορίζεται (γραμμή 181) ότι ο χειριστής του σήματος SIGINT είναι η συνάρτηση kill_child_processes που ορίστηκε στη γραμμή 165. Στη συνέχεια διαβάζονται τα ορίσματα που πέρασε ο χρήστης κατά την εκτέλεση του προγράμματος. Ο χρήστης μπορεί να περάσει δύο ορίσματα i) ένα μετά το όρισμα --vFile το οποίο μας προσδιορίζει το path του αποθηκευμένου αρχείου βίντεο και η απουσία αυτού του ορίσματος θα μας οδηγήσει στη χρήση κάμερας και ii) μετά το όρισμα --tracker που μας προσδιορίζει τον αλγόριθμο ιχνηλάτησης ενώ η απουσία του ορίσματος αυτού θα μας οδηγήσει στη χρήση by default του αλγορίθμου kcf (γραμμές 184, 185). Τα δύο ορίσματα αυτά αποθηκεύονται στο dictionary main_args(γραμμές 183-186). Έπειτα ξεκινάει η διαδικασία δημιουργίας παράλληλων διεργασιών (γραμμές 225-240) όπως εξηγήθηκε στην προηγούμενη

υποενότητα αφού όμως πρώτα έχει γίνει αρχικοποίηση βασικών μεταβλητών P,I,D και άλλων (γραμμές 195-214). Οι τιμές αυτές έχουν προκύψει από τη διαδικασία καλιμπραρίσματος (βλ. ενότητα 3.2.4). Στη main επίσης ενεργοποιούνται (γραμμές 191-192) οι σερβοκινητήρες του gimbal στην αρχή και απενεργοποιούνται (γραμμές 243-244) στο τέλος. Στο Τμήμα κώδικα 3 υπάρχει επίσης πλήθος σχολίων ανά γραμμή προς διευκόλυνση του αναγνώστη. Ο κώδικας αυτός αποτελεί βάση του προς παρουσίαση συστήματος, θα τροποποιηθεί όμως σε επόμενα κεφάλαια με σκοπό την αφαίρεση τμήματος κώδικα που ο χρήστης επιλέγει το αντικείμενο με το ποντίκι του και αντικατάσταση του κώδικα αυτού με αυτόματο εντοπισμό του αντικειμένου ενδιαφέροντος από το movidius (βλ.ενότητα 3.4.2) το οποίο θα έχει κατάλληλα εκπαιδευτεί. Ο κώδικας στηρίχθηκε στον ιστότοπο [9].

Τμήμα κώδικα 3 Κώδικας οδήγησης κάμερας με επιλογή από τον χρήστη του αντικειμένου ενδιαφέροντος

```
1  #import necessary packages
2  from multiprocessing import Manager
3  from multiprocessing import Process
4  from imutils.video import VideoStream
5  from imutils.video import FPS
6  from pyimagesearch.objcenter import objCenter
7  from pyimagesearch.pid import PID
8  import pantilthat as pth
9  import argparse
10 import signal
11 import time
12 import sys
13 import cv2
14 import imutils
15 import constants as con
16 import os
```

```
17  import subprocess
18
19  processObjectCenter = 0
20  processPanning = 0
21  processTilting = 0
22  processSetServos = 0
23
24
25
26  def Center_Detection(args, tracker, objX, objY, Frame_Center_X, Frame_Center_Y):
27
28
29
30      if args["vFile"] is None:
31          print("[ATD MESSAGE] Starting Camera Streaming")
32          Video_Source = VideoStream(usePiCamera=True).start()
33          time.sleep(con.Minimum_Delay)
34      else:
```

```
35          print("[ATD MESSAGE] Starting video Streaming from file")
36          Video_Source = cv2.VideoCapture(args["vFile"])
37          time.sleep(con.Minimum_Delay)
38      Initial_Bounding_Box = None
39      Frames_Per_Second = None
40
41
42      while (Initial_Bounding_Box == None):
43          frame = Video_Source.read()
44
45          if args["vFile"] != None:
46              frame = frame[1]
47          else:
48              frame = cv2.flip(frame, 0)
49          if frame is None:
50              break
51          frame = imutils.resize(frame, width=640)
52          (Height, Width) = frame.shape[:2]
```

```
53          Frame_Center_X.value = Width//2
54          Frame_Center_Y.value = Height//2
55          cv2.imshow(con.System_Title, frame)
56          Key_Selection = cv2.waitKey(con.Minimum_Delay) & 0xFF
57
58          if Key_Selection == ord("s"):
59              Initial_Bounding_Box = cv2.selectROI(con.System_Title, frame, fromCenter=False,showCross-
hair=True)
60              obj1 = objCenter(tracker)
61              selectedTracker = obj1.newTracker(tracker)
62              selectedTracker.init(frame, Initial_Bounding_Box)
63              Frames_Per_Second = FPS().start()
64              break
65
66      #print(Initial_Bounding_Box)
67      #end of while 1
68
69      #loop indefinitely
70
```

```
71
72      while True:
73
74
75          frame = Video_Source.read()
76
77          if args["vFile"] != None:
78              frame = frame[1]
79          else:
80              frame = cv2.flip(frame, 0)
81          if frame is None:
82              break
83
84
85          frame = imutils.resize(frame, width=640)
86          (Height, Width) = frame.shape[:2]
87          Frame_Center_X.value = Width//2
88          Frame_Center_Y.value = Height//2
```

```
89
90          updatedObject = obj1.update(selectedTracker, frame, (Frame_Center_X.value, Frame_Center_Y.value), Initial_Bounding_Box)
91          ((objX.value,objY.value),Target_Box) = updatedObject
92          if Target_Box is not None:
93              (x, y, w, h) = Target_Box
94              cv2.rectangle(frame, (x, y), (x + w, y + h),(0, 255, 0), 2)
95
96
97          Frames_Per_Second.update()
98          Frames_Per_Second.stop()
99
100
101         SCREEN_MESSAGE = [
102             ("Tracker", args["tracker"]),
103             ("Success", "Yes" if Target_Box is not None else "No"),
104             ("Frames_Per_Second", "{:.2f}".format(Frames_Per_Second.fps())),
105             ("System", con.System_Title)
106         ]
```

```
107
108
109
110
111         for (i, (Message_Title, Message_Text)) in enumerate(SCREEN_MESSAGE):
112             text = "{}: {}".format(Message_Title, Message_Text)
113             cv2.putText(frame, text, (10, Height - ((i * 20) + 20)),
114             cv2.FONT_HERSHEY_TRIPLEX, 0.6, (0, 255, 0), 2)
115
116
117         cv2.imshow(con.System_Title, frame)
118         Key_Selection = cv2.waitKey(con.Minimum_Delay) & 0xFF
119
120     #end of while 2
121 #end of function Center_Detection
122
123
124
```

```
125 def Drive_Servos (pan, tlt):
126
127     #loop indefinitely
128     while True:
129         #the pan and tilt angles are reversed
130         panAngle = -1 * pan.value
131         tiltAngle = -1 * tlt.value
132
133         #if the pan angle is within the range, pan
134         if Range_Validation(panAngle, con.Servo_Angle_Range[0], con.Servo_Angle_Range[1]):
135             pth.pan(panAngle)
136
137         #if the tilt angle is within the range, tilt
138         if Range_Validation(tiltAngle, con.Servo_Angle_Range[0], con.Servo_Angle_Range[1]):
139             pth.tilt(tiltAngle)
140     #end of while True
141 #end of function Drive_Servos
142
```

```
143 def Pid_Parameters_Calculation(output, p, i, d, objCoord, centerCoord):
144     #create a PID and initialize it
145     p = PID(p.value, i.value, d.value)
146     p.initialize()
147 
148     #loop indefinitely
149     while True:
150         #calculate the error
151         error = centerCoord.value - objCoord.value
152 
153         #update the value
154         output.value = p.update(error)
155     #end of loop
156     #end of function Pid_Parameters_Calculation
157 
158 def Range_Validation(val, start, end):
159     #determine the input value is in the supplied range
160     return (val >= start and val <= end)
```

```
161     #end of function Range_Validation
162
163
164
165 def kill_child_processes(signum, frame):
166     pth.servo_enable(con.Servo_Pan, False)
167     pth.servo_enable(con.Servo_Tilt, False)
168     processObjectCenter.kill()
169     processPanning.kill()
170     processTilting.kill()
171     processSetServos.kill()
172     sys.exit()
173
174
175
176
177
178 #check to see if this is the main body of execution
```

```
179 if __name__=="__main__":
180     #signal trap to handle keyboard interrupt
181     signal.signal(signal.SIGINT, kill_child_processes)
182     #construct the argument parser and parse the arguments
183     arguments = argparse.ArgumentParser()
184     arguments.add_argument("-v", "--vFile", type=str, help="path to input video file")
185     arguments.add_argument("-t", "--tracker", type=str, default="kcf", help="OpenCV object tracker type")
186     main_args = vars(arguments.parse_args())
187 
188     #start a manager for managing process-safe variable
189     with Manager() as manager:
190         #enable the servos
191         pth.servo_enable(con.Servo_Pan, True)
192         pth.servo_enable(con.Servo_Tilt, True)
193 
194         #set integer values for the object center (x, y)-coordinates
195         Frame_Center_X = manager.Value("i", 0)
196         Frame_Center_Y = manager.Value("i", 0)
```

```
197
198         #set integer values for the object's (x, y)-coordinates
199         objX = manager.Value("i", 0)
200         objY = manager.Value("i", 0)
201
202         #pan and tilt values will be managed by independed PIDs
203         pan = manager.Value("i", 0)
204         tlt = manager.Value("i", 0)
205
206         #set PID values for panning
207         panP = manager.Value("f", 0.025)
208         panI = manager.Value("f", 0.000008)
209         panD = manager.Value("f", 0.0000009)
210
211         #set PID values for tilting
212         tiltP = manager.Value("f", 0.022)
213         tiltI = manager.Value("f", 0.000005)
214         tiltD = manager.Value("f", 0.0000002)
```

215	
216	
217	#we have 4 independent processes
218	#1. objectCenter - finds/localizes the object
219	#2. panning - PID control loop determines panning angle
220	#3. tilting - PID control loop determines tilting angle
221	#4. setServos - drives the servos to proper angles based
222	# on PID feedback to keep object in center
223	
224	#def Center_Detection(args, tracker, objX, objY, Frame_Center_X, Frame_Center_Y):
225	processObjectCenter = Process(target=Center_Detection, args=(main_args, main_args["tracker"], objX, objY, Frame_Center_X, Frame_Center_Y))
226	processPanning = Process(target=Pid_Parameters_Calculation, args=(pan, panP, panI, panD, objX, Frame_Center_X))
227	processTilting = Process(target=Pid_Parameters_Calculation, args=(tlt, tiltP, tiltI, tiltD, objY, Frame_Center_Y))
228	processSetServos = Process(target=Drive_Servos, args=(pan, tlt))
229	
230	#start all 4 processes

```
231         processObjectCenter.start()
232         processPanning.start()
233         processTilting.start()
234         processSetServos.start()
235
236         #join all 4 processes
237         processObjectCenter.join()
238         processPanning.join()
239         processTilting.join()
240         processSetServos.join()
241
242         #disable the servos
243         pth.servo_enable(con.Servo_Pan, False)
244         pth.servo_enable(con.Servo_Tilt, False)
```

Τμήμα κώδικα 4 Η κλάση objCenter

```
1  import imutils
2  import cv2
3
4  class objCenter:
5      def __init__(self,myTracker): ####change the haarPath
6          #load OpenCV's Haar cascade face detector
7          #self.detector = cv2.CascadeClassifier(myTracker) #####also must be changed
8
9
10     def newTracker(self,myTracker):
11         OPENCV_OBJECT_TRACKERS = {
12         "csrt": cv2.TrackerCSRT_create,
13         "kcf": cv2.TrackerKCF_create,
14         "boosting": cv2.TrackerBoosting_create,
15         "mil": cv2.TrackerMIL_create,
16         "tld": cv2.TrackerTLD_create,
17         "medianflow": cv2.TrackerMedianFlow_create,
```

```
18          "mosse": cv2.TrackerMOSSE_create
19          }
20          print(OPENCV_OBJECT_TRACKERS[myTracker])
21          tracker = OPENCV_OBJECT_TRACKERS[myTracker]()
22          return tracker
23
24      def update(self, tracker, frame, frameCenter, initBB):
25          if initBB is not None:
26              (success,box) = tracker.update(frame)
27          if success:
28              (x, y, w, h) = [int(v) for v in box]
29              objX = int(x + (w/2.0))
30              objY = int(y + (h/2.0))
31              return ((objX, objY), (x, y, w, h))
32          return (frameCenter, None)
```

Ο εντοπισμός του κέντρου ενός στιγμιοτύπου του βίντεο γίνεται δημιουργώντας ένα βοηθητικό αρχείο (module) στην Python με κατάλληλο import στη γραμμή 6 του Τμήμα κώδικα 3. Το αρχείο αυτό παρουσιάζεται στο Τμήμα κώδικα 4. Όπως φαίνεται στον κώδικα δημιουργείται μία κλάση objCenter. Ο constructor δέχεται ως παράμετρο τον tracker που θα του δοθεί και θα πρέπει να είναι ένας από τους 7 (γραμμή 12-18) που ορίζονται στη μέθοδο newTracker η οποία βρίσκει τον αντίστοιχο tracker από τη βιβλιοθήκη cv2. Αφού δημιουργηθεί ο tracker (γραμμή 21) επιστρέφεται στο κυρίως πρόγραμμα (γραμμή 22). Η μέθοδος update (γραμμές 24-32) δέχεται ως παραμέτρους τον tracker, το frame του βίντεο, το κέντρο του frame και την δομή δεδομένων initBB που θα είναι είτε κενή είτε θα είναι ένα ορθογώνιο πλαίσιο. Αν δεν είναι κενό το initBB (γραμμή 25) τότε δημιουργείται μία tuple με στοιχεία success και box τα στοιχεία που επιστρέφει η μέθοδος update του tracker (γραμμή 26). Αν το success δεν είναι κενό τότε σε μία tuple τα στοιχεία τα x, y, w, h παίρνουν τις τιμές του ορθογώνιου πλαισίου (γραμμή 28), δηλαδή τις συντεταγμένες του αριστερού άκρου του frame (x,y) και τις διαστάσεις του ορθογώνιου πλαισίου, δηλαδή πλάτος και ύψος (w, h). Δύο μεταβλητές objX, objY παίρνουν τις τιμές του κέντρου του ορθογώνιου πλαισίου (γραμμές 29, 30). Τέλος, επιστρέφεται μία tuple που αποτελείται από 2 tuples: (objX, objY) και (x, y, w, h) (γραμμή 31). Αν το success είναι κενό τότε δεν επιστρέφεται το κέντρο του frame (γραμμή 32).

Τμήμα κώδικα 5 constants.py

1	`Servo_Pan = 1`
2	`Servo_Tilt = 2`
3	`Minimum_Delay = 1`
4	`Servo_Angle_Range = (-45, 45)`
5	`System_Title = "Automated Tank Detector (ATD)"`
6	`Final_Width = 640`

Στο Τμήμα κώδικα 5 ορίζονται οι βασικές παράμετροι του συστήματος σε ξεχωριστό αρχείο με σκοπό ο εύκολος εντοπισμός τους και παραμετροποίησής τους από τον χρήστη. Αναλυτικότερα ορίζεται στη γραμμή 1 η θέση του σερβοκινητήρα υπεύθυνου για την οριζόντια κίνηση στο raspberry (θέση 1). Στη γραμμή 2 ορίζεται η θέση του σερβοκινητήρα υπεύθυνου για την κάθετη κίνηση στο raspberry (θέση 2). Στη γραμμή 3 ορίζεται μία ελάχιστη καθυστέρηση που χρησιμοποιοείται σε διάφορα σημεία του κώδικα και είναι ίση με

1 δευτερόλεπτο. Στη γραμμή 4 ορίζεται το εύρος τιμών κίνησης των σερβοκινητήρων σε γωνία 90ο. Στη γραμμή 5 ορίζεται ο τίτλος του συστήματος με σκοπό τη χρήση του αλφαριθμητικού αυτού σε μηνύματα του συστήματος. Τέλος, στη γραμμή 6 ορίζεται το πλάτος του πλαισίου που θα εμφανίζεται στο χρήστη.

3.2.3 PID έλεγχος

Ο ελεγκτής Proportional-Integral-Derivative (PID controller[26]) χρησιμοποιείται ευρέως επειδή είναι αρκετά κατανοητός και πάρα πολύ χρήσιμος. Ένα πλεονέκτημα του PID controller είναι ότι η εννοιολογική διαφοροποίηση και ολοκλήρωση είναι κατανοητές με τέτοιο τρόπο ώστε να μπορούν να εφαρμοστούν στο σύστημα του ελεγκτή χωρίς να είναι απαραίτητη προϋπόθεση η γνώση της θεωρίας του ελέγχου. Επιπρόσθετα, παρόλο που ο αντισταθμιστής (compensator) είναι απλός, έχει τεράστιες δυνατότητες αποτυπώνοντας την ιστορία του συστήματος μέσω ολοκλήρωσης και αποτυπώνει τη μελλοντική συμπεριφορά του συστήματος (μέσω παραγώγισης differentiation).

3.2.4 Διαδικασία calibration του συστήματος

Ακολουθώντας τη διαδικασία που προτείνεται στο [10] επιτεύχθηκε το καλιμπράρισμα του προτεινόμενου συστήματος στα χαρακτηριστικά του δικού μας υλικού και του περιβάλλοντος ορίζοντας κατάλληλες τιμές για τις κρίσιμες μοιραζόμενες μεταβλητές του Τμήμα κώδικα 6. Συνοπτικά τα βήματα της μεθοδολογίας αυτής φαίνονται παρακάτω

1. Ορίζουμε την παράμετρο kI και kD σε μηδέν.

2. Αυξάνουμε την παράμετρο kP από μηδέν έως την τιμή η οποία θα προσδώσει μία πρώτη μικρή κίνηση στο σερβοκινητήρα και μειώνουμε στο μισό την τιμή του kP.

3. Αυξάνουμε το kI μέχρι να παρατηρήσουμε γρήγορη και σωστή απόκριση του σερβοκινητήρα.

4. Αυξάνουμε το kD μέχρις ότου η κίνηση του σερβοκινητήρα μπορεί να εξυπηρετήσει πολύ γρήγορες κινήσεις του αντικειμένου στόχου.

ΚΕΦΑΛΑΙΟ 4
Αυτόματη ανίχνευση αντικειμένου

4.1 Ορισμός machine learning

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης και της επιστήμης υπολογιστών ή οποία επικεντρώνεται στη χρήση δεδομένων και αλγορίθμων για να μιμηθεί τον τρόπο με τον οποίο μαθαίνουν οι άνθρωποι, βελτιώνοντας σταδιακά την ακρίβειά του. Στην εν λόγω εργασία χρησιμοποιούμε την βιβλιοθήκη ανοιχτού κώδικα εικόνας υπολογιστή (OpenCV) που περιέχει αλγορίθμους μηχανικής μάθησης.

4.2 Movidius

Οι Intel Movidius Vision Processing Units (VPUs) [7](βλ. Εικόνα 9)επιτρέπουν την αποδοτική εκτέλεση απαιτητικών εφαρμογών υπολογιστικής όρασης και τεχνητής νοημοσύνης (AI). Η αποδοτική εκτέλεση επιτυγχάνεται μέσω παράλληλων προγραμματιζόμενων υπολογισμών σε συνδυασμό με κατάλληλη επιτάχυνση υλικού για συγκεκριμένο φόρτο εργασίας σε μια μοναδική αρχιτεκτονική που ελαχιστοποιεί την μετακίνηση των δεδομένων. Οι VPUs της Movidius επιτυγχάνουν ισορροπία απόδοσης ισχύος και υπολογιστικής απόδοσης. Η τεχνολογία VPU επιτρέπει τη διασύνδεση σε έξυπνες κάμερες, τερματικούς διακομιστές και συσκευές τεχνητής νοημοσύνης. Συχνή χρήση VPUs γίνεται σε εφαρμογές βασισμένες σε βαθιά νευρωνικά δίκτυα, υπολογιστική όραση και σε τομείς όπως η αυτόματη μέτρηση κατά την παραγωγή εμπορευμάτων, η ασφάλεια και ο βιομηχανικός αυτοματισμός.

Εικόνα 9 Movidius neural compute stick[35]

4.3 Εκπαίδευση Movidius

Η εκπαίδευση έγινε με την πλατφόρμα caffe-cpu που περιγράφεται αναλυτικά στην ενότητα 5.1.8. Στο σημείο αυτό θα αναφερθεί ότι ένας χρήστης έχει τη δυνατότητα επιλογής δημιουργίας μοντέλου μηχανικής μάθησης μέσω ενός πλήθους διαφορετικών πλατφορμών όπως φαίνεται στον Πίνακας 3 [11]. Στην παρούσα διπλωματική εργασία έχει επιλεγεί η δημιουργία μοντέλου τύπου caffe [12],[13] και έχει αναπτυχθεί κατάλληλος κώδικας σε python για την αναγνώριση των αντικειμένων που επιστρέφει το μοντέλο (βλ. Τμήμα κώδικα 6). Το μοντέλο έχει αποθηκευτεί και εκτελείται στον ειδικό επεξεργαστή movidius ο οποίος έχει συνδεθεί με το Raspberry Pi όπως περιεγράφηκε στην ενότητα 3.1.2.

Πίνακας 3 Συγκριτικός πίνακας εργαλείων μηχανικής μάθησης

Εργαλείο	Λειτουργικό Σύστημα	Γλώσσα	Αλγόριθμοι/ Χαρακτηριστικά
Caffe	Linux, Mac OS, Windows	C++, Python, Matlab, CUDA	• Ταξινόμηση • Ομαδοποίηση • Προεπεξεργασία • Επιλογή Μοντέλου • R-CNN ανίχνευση
Shogun	Windows Linux UNIX Mac OS	C++	• Μέθοδος οπισθοδρόμησης • Ταξινόμηση • Ομαδοποίηση • Υποστήριξη συστοιχίας μηχανημάτων • Μείωση διαστάσεων
Scikit-learn	-//-	Python, Cython, C, C++	• Ταξινόμηση • Μέθοδος οπισθοδρόμησης • Ομαδοποίηση • Προεπεξεργασία • Επιλογή Μοντέλου • Μείωση διαστάσεων
PyTorch	-//-	Python, C++, CUDA	• Autograd Μονάδα • Optim Μονάδα • nn Μονάδα
Tensor-Flow	-//-	-//-	Παροχή βιβλιοθήκης για dataflow programming
Weka	-//-	Java	• Προετοιμασία δεδομένων • Ταξινόμηση • Μέθοδος οπισθοδρόμησης • Ομαδοποίηση

			• Οπτικοποίηση Αποτελεσμάτων • Κανόνες συσχέτισης εξόρυξης
KNIME	-//-	Java	• Υποστήριξη εξόρυξης κειμένου και εικόνων • Αποτελεσματικό με μεγάλο μέγεθος δεδομένων
Colab	Cloud Service	–	Υποστήριξη βιβλιοθηκών PyTorch, Keras, TensorFlow, and OpenCV
Apache Mahout	Cross-platform	Java Scala	• Προεπεξεργαστές • Μέθοδος οπισθοδρόμησης • Ομαδοποίηση • Κατανεμημένη Γραμμική Άλγεβρα
Accors.Net	-//-	C#	• Ταξινόμηση • Μέθοδος οπισθοδρόμησης • Κατανομή • Ομαδοποίηση • Μέθοδοι σε επίπεδο πυρήνα
Keras.io	-//-	Python	• API για νευρωνικά δίκτυα
RapidMiner	-//-	Java	• Προεπεξεργασία δεδομένων και οπτικοποίηση • Φόρτωση και μετασχηματισμός δεδομένων

4.4 Χειρισμός Movidius από raspberry

Στο Τμήμα κώδικα 6 παρουσιάζεται το πρόγραμμα το οποίο επιτυγχάνει την αυτόματη αναγνώριση ενός αντικειμένου με χρήση κατάλληλου εκπαιδευμένου μοντέλου μηχανικής μάθησης και συνδυάζει όλες τις προηγούμενες λειτουργίες που έχουν αναλυτικά περιγραφεί στο κεφάλαιο 3.2.2. Αξίζει να σημειωθεί ότι η διαδικασία της αναγνώρισης εκτελείται σε επεξεργαστή ειδικού σκοπού movidius ο οποίος συνεργάζεται-επικοινωνεί με το raspberry pi το οποίο είναι υπεύθυνο για την οδήγηση του gimbal της κάμερας.

Ο κώδικας που φαίνεται παρακάτω αποτελεί επέκταση του κώδικα που περιγράφηκε στο κεφάλαιο 3.2.2, με τη διαφορά ότι η μεθοδολογία της αναγνώρισης (task detection) εκτελεί τελείως διαφορετικό κώδικα ο οποίος θα αναλυθεί λεπτομερώς δεδομένου ότι η επιλογή του αντικειμένου δεν γίνεται πια από το χρήστη αλλά αναγνωρίζεται αυτόματα. Επιπρόσθετα, για την ορθή εκτέλεση του παρακάτω προγράμματος χρειάζεται ένα τροποποιημένο consants.py αρχείο το οποίο θα περιγραφεί στη συνέχεια, ένα αρχείο MobileNetSSD_deploy.caffemodel [8] και ένα αρχείο MobileNetSSD_deploy.prototxt. Τα δύο αυτά αρχεία στην πραγματικότητα εμπεριέχουν το εκπαιδευμένο μοντέλο.

Η αρχιτεκτονική του προγράμματος που ακολουθεί είναι παρόμοια των προγραμμάτων που παρουσιάστηκαν σε προηγούμενα κεφάλαια, δηλαδή τέσσερις παράλληλες διεργασίες με αυστηρά καθορισμένο διακριτό ρόλο η καθεμία εκτελούνται και συνεργάζονται με σκοπό την επίτευξη του στόχου που είναι υ συνεχής παρακολούθηση ενός αντικειμένου. Στο πρόγραμμα αυτό οι τέσσερις διεργασίες είναι οι: i) Detector_Tracking για τον εντοπισμό του αντικειμένου , ii) και iii) Pid_Parameters_Calculation για τον υπολογισμό των PID παραμέτρων για pan και tlt και iv) Drive_Servos για την οδήγηση των σερβοκινητήρων. Ακολουθεί λεπτομερής περιγραφή του κώδικα.

Αρχικά γίνονται import τα βασικά πακέτα της python που θα μας χρειαστούν με τα πιο βασικά να είναι τα computer vision(cv2) και image utilities(imutils). Η εισαγωγή των πακέτων αυτών γίνονται στις γραμμές 2-18.

Στις γραμμές 20-23 ορίζονται οι μεταβλητές που θα αποθηκεύουν τα PID των διεργασιών που θα δημιουργηθούν στην κύρια συνάρτηση (main). Το σώμα των διεργασιών αυτών περιγράφεται αναλυτικά στη συνέχεια του κεφαλαίου στην αντίστοιχη για κάθε διεργασία συνάρτηση.

Στις γραμμές 25-114 ορίζεται η συνάρτηση Detector_Tracking που θα αποτελέσει σώμα μίας από τις τέσσερις παράλληλες διεργασίες που θα δημιουργήσει το σύστημα και πιο συγκεκριμένα της διεργασίας που είναι υπεύθυνη για τον εντοπισμό του αντικειμένου. Αρχικά στη γραμμή 28 αρχικοποιείται το μοντέλο της μηχανικής μάθησης και στη γραμμή 31 ορίζεται ότι το μοντέλο θα εκτελεστεί σε movidius. Να σημειωθεί ότι στην έκδοση αυτή οι βασικές παράμετροι δεν δίνονται από χρήστη σε γραμμή εντολών αλλά ορίζονται στο αρχείο constants.py. Για παράδειγμα στο αρχείο constants υπάρχει το dictionary `main_args= {'vFile': None, 'prototxt': 'MobileNetSSD_deploy.prototxt', 'model': 'MobileNetSSD_deploy.caffemodel', 'confidence': 0.50, 'movidius': 1}` στο οποίο περιγράφονται βασικά ορίσματα του προγράμματος.

Στις γραμμές 33-40 ελέγχεται η τιμή στον δείκτη "vFile" στο dictionary args το οποίο η συνάρτηση έχει δεχτεί ως πρώτο όρισμα και στην πράξη είναι το main_args. Η

τιμή του δείκτη αυτή μας προσδιορίζει αν γίνεται η εισαγωγή της εικόνας από την PiCamera ή από αποθηκευμένο βίντεο. Αν η τιμή του δείκτη αυτού είναι κενή σημαίνει ότι θα πάρουμε την εικόνα από την PiCamera, για το λόγο αυτό στη γραμμή 35 εκκινούμε το videostreaming από την PiCamera και αποθηκεύουμε τον προσδιοριστή της πηγής εικόνας στη μεταβλητή Video_Source. Η εντολή αυτή συνοδεύεται από μία καθυστέρηση χρόνου ,ο οποίος έχει προσδιοριστεί στο τμήμα κώδικα(constants), (γραμμή 36) για να δοθεί χρόνος στην κάμερα να ενεργοποιηθεί. Αν η τιμή του δείκτη βίντεο δεν είναι κενή (γραμμή 37) σημαίνει ότι μας έχει περάσει ως όρισμα το όνομα αρχείου αποθηκευμένου βίντεο το οποίο εκκινούμε στη γραμμή 39 αρχικοποιώντας αντίστοιχα τον προσδιοριστή της πηγής εικόνας στη μεταβλητή Video_Source. Και στην περίπτωση αυτή υπάρχει αντίστοιχη καθυστέρηση χρόνου, που έχει ομοίως προσδιοριστεί στο τμήμα κώδικα(constants), (γραμμή 40). Σε κάθε περίπτωση εμφανίζεται στο χρήστη αντίστοιχο μήνυμα (γραμμή 34 ή γραμμή 38). Στη συνέχεια ορίζεται μία λίστα (CLASSES) με όλα τα πιθανά αντικείμενα που εν δυνάμει μπορεί να αναγνωρίσει το μοντέλο (γραμμή 42-45). Στο συγκεκριμένο πρόγραμμα για χάριν της επαλήθευσης της ορθότητάς του έχουμε επικεντρωθεί στην αναγνώριση του αντικειμένου τύπου "bottle". Στη συνέχεια αρχικοποιούμε τη μεταβλητή Frames_Per_Second σε κενή (γραμμή 47). Η μεταβλητή Frames_Per_Second θα αποθηκεύει την τρέχουσα τιμή των frames per second τη μέτρηση της οποίας εκκινούμε στη γραμμή 48. Επίσης στη γραμμή 49 ορίζεται μία Boolean μεταβλητή Found_Object η οποία αρχικοποιείται με False και είναι ένδειξη αν βρέθηκε το αντικείμενο ή όχι.

Στη γραμμή 50-112 υλοποιείται μία δομή επανάληψης while η οποία αποτελεί το κύριο σώμα της συνάρτησης object_detector. Η δομή επανάληψης αυτή εκτελείται χωρίς συνθήκη τερματισμού και μπορεί να τερματίσει είτε με εντολή break γραμμή 58 είτε όταν η γονική διεργασία λάβει σήμα SIGINT και εκτελέσει το handler kill_child_processes στη γραμμή 169. Η συνάρτηση kill_child_processes θα αναλυθεί στη συνέχεια. Σε κάθε επανάληψη της δομής while της συνάρτησης object_detector αρχικά διαβάζεται ένα καινούργιο frame (γραμμή 51) είτε από το αρχείο είτε από κάμερα. Στην περίπτωση που το frame διαβάζεται από την κάμερα χρειάζεται μία περιστροφή της κατά 180^{o} (γραμμή 56) εξαιτίας του τρόπου τοποθέτησής της στο gimbal. Η περιστροφή αυτή γίνεται με τη χρήση της εντολής flip η οποία δεχόμενη ως δεύτερο όρισμα την τιμή μηδέν πραγματοποιεί περιστροφή του frame (πρώτο όρισμα) γύρω από τον άξονα των x. Στην περίπτωση που δεν διαβαστεί frame ο βρόχος σπάει (γραμμή 57,58) δεδομένης αδυναμίας του συστήματος να διαβάσει frame είτε από την κάμερα είτε από το αρχείο. Δεδομένης επιτυχούς ανάγνωσης ενός frame γίνεται σε αυτό scaling σε πλάτος 640 (γραμμή 59) και υπολογίζονται οι νέες διαστάσεις του frame που αποθηκεύονται στις μεταβλητές (Height, Weight) ώς tuple (γραμμή 60). Ο χρήστης μπορεί να επιλέξει άλλη τιμή διαφορετική από 640 αν το επιθυμεί τροποποιώντας την αντίστοιχη παράμετρο στο αρχείο constants.py

μεταβάλλοντας τη σταθερά init_width. Στη συνέχεια τα ορίσματα Frame_Center_X.value, Frame_Center_Y.value λαμβάνουν τιμές ίσες με το μισό πλάτος και μισό ύψος αντίστοιχα (γραμμές 61,62). Στη συνέχεια το frame δίνεται ως όρισμα στη συνάρτηση dnn.blobFromImage του πακέτου cv2, η οποία μετατρέπει το frame σε μορφή που μπορεί να επεξεργαστεί το μοντέλο της μηχανικής μάθησης, στην προκειμένη περίπτωση το MobileNetSSD_deploy.caffemodel η δομή που επιστρέφει η συνάρτηση είναι τύπου binary large object (blob) και θα δοθεί ως input στο νευρωνικό δίκτυο (γραμμή 65) το οποίο θα μας επιστρέψει μία δομή δεδομένων (detections) η οποία περιέχει πληροφορίες αναφορικά με τα αντικείμενα που ανιχνεύθηκαν (γραμμή 66). Στη συνέχεια με μία δομή επανάληψης for (γραμμές 68-93) διατρέχουμε τη δομή δεδομένων detections. Σε κάθε επανάληψη της δομής for για κάθε στιγμιότυπο στο detections διαβάζουμε τη βεβαιότητα αναγνώρισης του συγκεκριμένου προτύπου και το αποθηκεύουμε στη μεταβλητή confidence (γραμμή 71). Αν η μεταβλητή confidence είναι μεγαλύτερη από αυτή που μας έχει ορίσει ο χρήστης ως ελάχιστη αποδεκτή βεβαιότητα, λίστα args αρχείου constants.py στον δείκτη "confidence" (γραμμή 74) τότε μπαίνουμε στη δομή απόφασης if των γραμμών 74-93. Στο σημείο αυτό διαβάζουμε από τη δομή detections το είδος αντικειμένου που έχει ανιχνευτεί και πιο συγκεκριμένα έναν δείκτη στη λίστα CLASSES τον οποίο αποθηκεύουμε στη μεταβλητή idx (γραμμή 78). Αν ο τύπος αντικειμένου που ανιχνεύτηκε είναι αυτός που μας ενδιαφέρει, για το συγκεκριμένο παράδειγμα "bottle", τότε μπαίνουμε στη δομή απόφασης γραμμών 79-93. Στη δομή αυτή μεταβάλλουμε την τιμή της μεταβλητής Found_Object σε True για να εμφανίσουμε την κατάλληλη ένδειξη σε μήνυμα οθόνης που θα αναφερθεί παρακάτω (γραμμή 98-103). Έπειτα διαβάζουμε από τη δομή detections τη θέση του αντικειμένου στο frame και την αποθηκεύουμε στη δομή box (γραμμή 82). Η δομή αυτή, αφού μεταβάλλουμε τις τιμές της σε ακέραιη μορφή, αποθηκεύεται σε μία tuple (γραμμή 83). Τα πρώτα δύο μέλη της tuple μας ορίζουν τις συντεταγμένες του άνω αριστερού άκρου του box τις οποίες αποθηκεύουμε στις μεταβλητές objX, objY και είναι μοιραζόμενες με τις άλλες διεργασίες και χρήσιμες για την εκτέλεσή τους (γραμμή 84-85). Ένα αλφαριθμητικό κατασκευάζεται στη συνέχεια με όνομα label το οποίο περιέχει τον τύπο του αντικειμένου που ανιχνεύθηκε και τη βεβαιότητα αναγνώρισής του (γραμμή 87). Το label αυτό σε συνδυασμό με ένα ορθογώνιο πλαίσιο που περιβάλλει το αντικείμενο που εντοπίστηκε προστίθεται στο frame (γραμμές 88-90) με κατάλληλη γραμματοσειρά και χρώμα. Αν το στιγμιότυπο της δομής detections δεν είναι του τύπου που μας ενδιαφέρει , στο συγκεκριμένο παράδειγμα τύπος bottle, το πρόγραμμα οδηγείται στην εντολή else της γραμμής 92 όπου απλά θέτουμε τη μεταβλητή Found_Object ίση με False και βγαίνουμε από τη δομή if της γραμμής 79 και από τη δομή if της γραμμής 74. Στο σημείο αυτό ανανεώνουμε την τιμή της μεταβλητής Frames_Per_Second (γραμμές 95,96). Δημιουργούμε μία λίστα (SCREEN_MESSAGE) από κατάλληλα αλφαριθμητικά τα οποία θα

αποτελέσουν χρήσιμες πληροφορίες για τον χρήστη που θα εμφανιστούν στην κάτω αριστερή γωνία του frame. Πιο συγκεκριμένα ενημερώνεται ο χρήστης αν υπήρχε επιτυχής ή όχι αναγνώρισης του αντικειμένου, ο τύπος του αντικειμένου, η τιμή της μεταβλητής Frames_Per_Second και το όνομα του συστήματος. Η λίστα των αλφαριθμητικών αυτών ενσωματώνεται στο frame με τη χρήση της δομής επανάληψης for των γραμμών 105 έως 108. Η τελική μορφή του frame, δηλαδή η εικόνα μαζί με το label, το ορθογώνιο περίγραμμα του αντικειμένου και το μήνυμα της οθόνης (SCREEN_MESSAGE) εμφανίζεται στην οθόνη του χρήστη με την εντολή imshow του πακέτου cv2 (γραμμές 110,111). Μετά την εμφάνιση του frame τερματίζει μία επανάληψη της δομής while των γραμμών 50-111 και οδηγούμαστε στην επόμενη επανάληψη που θα εκτελέσει ακριβώς την ίδια διαδικασία με νέο όμως frame.

Στις γραμμές 116-132 ορίζεται η συνάρτηση Drive_Servos που θα αποτελέσει σώμα της τέταρτης παράλληλης διεργασίας. Πιο συγκεκριμένα, επαναληπτικά (βρόχος while γραμμές 118-129) υπολογίζονται οι τιμές των γωνιών οριζόντιας και κάθετης μετακίνησης (γραμμές 120-121) δεδομένων των παραμέτρων pan και tlt που δέχεται η συνάρτηση και αν οι γωνίες αυτές είναι εντός ορίων, όπως τα όρια αυτά ορίστηκαν στην tuple Servo_Angle_Range του αρχείου constants.py, μέσω της συνάρτησης Range_Validation καλεί την αντίστοιχη συνάρτηση ενημέρωσης των σερβοκινητήρων (γραμμές 125 και 129) που ανήκουν στο πακέτο pantilthat. Η συνάρτηση Range_Validation επιστρέφει μία Boolean μεταβλητή η οποία είναι true αν το πρώτο όρισμα είναι μεταξύ του δεύτερου και του τρίτου (con. Servo_Angle_Range[0], con. Servo_Angle_Range[1]), διαφορετικά επιστρέφει false. Τα εύρη τιμών λειτουργίας των σερβοκινητήρων μπορεί ο χρήστης να τα μεταβάλλει στο αρχείο constants.py.

Στις γραμμές 134-145 ορίζεται η συνάρτηση Pid_Parameters_Calculation που θα αποτελέσει σώμα της δεύτερης και τρίτης παράλληλης διεργασίας. Πιο συγκεκριμένα δημιουργείται και αρχικοποιείται (γραμμές 136,137) ένα αντικείμενο της κλάσης PID (βλ αρχείο κώδικα pid.py). Το αντικείμενο αυτό έχει ως στόχο τον συνεχόμενο (βρόχος while γραμμές 140-145) υπολογισμό του σφάλματος και των PID παραμέτρων που θα χρησιμοποιηθούν στην οδήγηση των σερβοκινητήρων του gimbal που γίνεται από τις δύο αυτές διεργασίες. Η συνάρτηση αυτή όπως αναφέρθηκε αποτελεί σώμα για δύο διεργασίες η διαφορετική λειτουργία της όμως για κάθε διεργασία προσδιορίζεται μέσω της πρώτης παραμέτρου output όπως φαίνεται στις γραμμές 213, 214 όπου η μία διεργασία καλείται με πρώτο argument pan ενώ η δεύτερη καλείται με πρώτο argument tlt.

Στις γραμμές 158-172 ορίζεται η συνάρτηση kill_child_processes η οποία αρχικά στις γραμμές 162,163 εκτελεί τις συναρτήσεις servo_enable του πακέτου pth που απενεργοποιούν τους δύο σερβοκινητήρες αφού στο δεύτερο όρισμά τους έχουν την Boolean

τιμή false. Στις γραμμές 166-169 έχουμε τον τερματισμό των τεσσάρων διεργασιών και στη γραμμή 170 πραγματοποιείται το κλείσιμο όλων των ενεργών παραθύρων του προγράμματος. Τέλος έχουμε τερματισμό της κύριας διεργασίας αφού περιμένει δύο δευτερόλεπτα για να βεβαιώσει τον τερματισμό των παιδιών της (γραμμές 171,172).

Αφού έχουν δηλωθεί οι προαναφερθέντες συναρτήσεις που θα αποτελέσουν σώμα των τεσσάρων διεργασιών ορίζεται η main στη γραμμή 155-232 που θα δημιουργήσει τις διεργασίες αυτές. Στην main αρχικά ορίζεται (γραμμή 158-172) η συνάρτηση kill_child_processes που αναλύθηκε προηγουμένως και στη συνέχεια (γραμμή 174) ότι ο χειριστής του σήματος SIGINT είναι η συνάρτηση kill_child_processes. Έπειτα ξεκινάει η διαδικασία δημιουργίας παράλληλων διεργασιών (γραμμές 177-227) όπως εξηγήθηκε στην προηγούμενη υποενότητα 3.2.1.1 αφού όμως πρώτα έχει γίνει αρχικοποίηση βασικών μεταβλητών P, I, D και άλλων μοιραζόμενων μεταξύ των διεργασιών μεταβλητών (γραμμές 183-202). Οι τιμές αυτές έχουν προκύψει από τη διαδικασία καλιμπραρίσματος (βλ. ενότητα 3.2.4). Στη main επίσης ενεργοποιούνται (γραμμές 179-180) οι σερβοκινητήρες του gimbal στην αρχή και απενεργοποιούνται (γραμμές 230-231) στο τέλος. Στο Τμήμα κώδικα 6 υπάρχει επίσης πλήθος σχολίων ανά γραμμή προς διευκόλυνση του αναγνώστη. Ο κώδικας αυτός αποτελεί την τελική έκδοση του προς παρουσίαση συστήματος, με την υποσημείωση ότι ο χρήστης μπορεί αλλάζοντας μοντέλο να το χρησιμοποιεί για τις δικές του ανάγκες. Στην παρούσα διπλωματική εργασία το μοντέλο του συστήματος αυτού αντικαταστάθηκε με αυτό της αυτόματης αναγνώρισης αρμάτων που υλοποιήθηκε όπως αναλύεται στην ενότητα 5.1.8.

Τμήμα κώδικα 6 movidius_final.py

```
1  #import necessary packages
2  from multiprocessing import Manager
3  from multiprocessing import Process
4  from imutils.video import VideoStream
5  from imutils.video import FPS
6  from pyimagesearch.objcenter import objCenter
7  from pyimagesearch.pid import PID
8  import pantilthat as pth
9  import argparse
10 import signal
11 import time
12 import sys
13 import cv2
14 import imutils
15 import constants as con
16 import numpy as np
```

```
17  import os
18  import subprocess
19
20  processDetectorTracking = 0
21  processPanning = 0
22  processTilting = 0
23  processSetServos = 0
24
25  def Detector_Tracking(args, objX, objY, Frame_Center_X, Frame_Center_Y, Video_Source):
26
27      print("[ATD MESSAGE] loading model...")
28      net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
29
30      # specify the target device as the Myriad processor on the NCS
31      net.setPreferableTarget(cv2.dnn.DNN_TARGET_MYRIAD)
32
33      if args["vFile"] is None:
34          print("[ATD MESSAGE] Starting Camera Streaming")
```

```
35          Video_Source = VideoStream(usePiCamera=True).start()
36          time.sleep(con.Minimum_Delay)
37      else:
38          print("[ATD MESSAGE] Starting video Streaming from file")
39          Video_Source = cv2.VideoCapture(args["vFile"])
40          time.sleep(con.Minimum_Delay)
41
42      CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
43      "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
44      "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
45      "sofa", "train", "tvmonitor"]
46
47      Frames_Per_Second = None
48      Frames_Per_Second = FPS().start()
49      Found_Object = False
50      while (True):
51          frame = Video_Source.read()
52
```

```
53          if args["vFile"] != None:
54              frame = frame[1]
55          else:
56              frame = cv2.flip(frame, 0)
57          if frame is None:
58              break
59          frame = imutils.resize(frame, width=con.init_width)
60          (Height, Width) = frame.shape[:2]
61          Frame_Center_X.value = Width//2
62          Frame_Center_Y.value = Height//2
63
64          blob = cv2.dnn.blobFromImage(frame, 0.007843, (300, 300), 127.5)
65          net.setInput(blob)
66          detections = net.forward()
67
68          for i in np.arange(0, detections.shape[2]):
69              # extract the confidence (i.e., probability) associated with
70              # the prediction
```

```
71              confidence = detections[0, 0, i, 2]
72              # filter out weak detections by ensuring the `confidence` is
73              # greater than the minimum confidence
74              if confidence > args["confidence"]:
75                  # extract the index of the class label from the
76                  # `detections`, then compute the (x, y)-coordinates of
77                  # the bounding box for the object
78                  idx = int(detections[0, 0, i, 1])
79                  if CLASSES[idx]=="bottle":
80
81                      Found_Object = True
82                      box = detections[0, 0, i, 3:7] * np.array([Width, Height, Width, Height])
83                      (startX, startY, endX, endY) = box.astype("int")
84                      objX.value = startX
85                      objY.value = startY
86                      # draw the prediction on the frame
87                      label = "{}: {:.2f}%".format(CLASSES[idx],confidence * 100)
88                      cv2.rectangle(frame, (startX, startY), (endX, endY),(0, 0, 255), 2)
```

```
89                      y = startY - 15 if startY - 15 > 15 else startY + 15
90                      cv2.putText(frame, label, (startX, y),cv2.FONT_HERSHEY_SIMPLEX,0.5, (0,0,255),2)
91
92                  else:
93                      Found_Object = False
94
95          Frames_Per_Second.update()
96          Frames_Per_Second.stop()
97
98          SCREEN_MESSAGE = [
99              ("Success", "Yes" if Found_Object==True else "No"),
100             ("Detected Object",CLASSES[idx] if Found_Object==True else "-"),
101             ("Frames per second", "{:.2f}".format(Frames_Per_Second.fps())),
102             ("System", con.System_Title)
103         ]
104
105         for (i, (Message_Title, Message_Text)) in enumerate(SCREEN_MESSAGE):
106             text = "{}: {}".format(Message_Title, Message_Text)
```

```
107             cv2.putText(frame, text, (10, Height - ((i * 20) + 20)),
108             cv2.FONT_HERSHEY_TRIPLEX, 0.6, (0, 255, 0), 2)
109
110         cv2.imshow(con.System_Title, frame)
111         Key_Selection = cv2.waitKey(con.Minimum_Delay) & 0xFF
112     #end of while
113     #loop indefinitely
114 #end of function Detector_Tracking
115
116 def Drive_Servos (pan, tlt):
117     #loop indefinitely
118     while True:
119         #the pan and tilt angles are reversed
120         panAngle = -1 * pan.value
121         tiltAngle = -1 * tlt.value
122
123         #if the pan angle is within the range, pan
124         if Range_Validation(panAngle, con.Servo_Angle_Range[0], con.Servo_Angle_Range[1]):
```

```
125             pth.pan(panAngle)
126
127         #if the tilt angle is within the range, tilt
128         if Range_Validation(tiltAngle, con.Servo_Angle_Range[0], con.Servo_Angle_Range[1]):
129             pth.tilt(tiltAngle)
130     #end of while True
131 #end of function Drive_Servos
132 #function to handle keyboard interrupt
133
134 def Pid_Parameters_Calculation(output, p, i, d, objCoord, centerCoord):
135     #create a PID and initialize it
136     p = PID(p.value, i.value, d.value)
137     p.initialize()
138
139     #loop indefinitely
140     while True:
141         #calculate the error
142         error = centerCoord.value - objCoord.value
```

```
143
144         #update the value
145         output.value = p.update(error)
146     #end of loop
147     #end of function Pid_Parameters_Calculation
148
149 def Range_Validation(val, start, end):
150     #determine the input value is in the supplied range
151     return (val >= start and val <= end)
152     #end of function Range_Validation
153
154 #check to see if this is the main body of execution
155 if __name__=="__main__":
156
157     #start a manager for managing process-safe variable
158     def kill_child_processes(*args):
159
160         #time.sleep(5)
```

```
161
162         pth.servo_enable(con.Servo_Pan, False)
163         pth.servo_enable(con.Servo_Tilt, False)
164
165         #Video_Source.stop()
166         processPanning.kill()
167         processTilting.kill()
168         processSetServos.kill()
169         processDetectorTracking.kill()
170         cv2.destroyAllWindows()
171         time.sleep(2)
172         sys.exit()
173
174     signal.signal(signal.SIGINT, kill_child_processes)
175
176
177     with Manager() as manager:
178         #enable the servos
```

```
179         pth.servo_enable(con.Servo_Pan, True)
180         pth.servo_enable(con.Servo_Tilt, True)
181
182         #set integer values for the object center (x, y)-coordinates
183         Frame_Center_X = manager.Value("i", 0)
184         Frame_Center_Y = manager.Value("i", 0)
185
186         #set integer values for the object's (x, y)-coordinates
187         objX = manager.Value("i", 0)
188         objY = manager.Value("i", 0)
189
190         #pan and tilt values will be managed by independed PIDs
191         pan = manager.Value("i", 0)
192         tlt = manager.Value("i", 0)
193
194         #set PID values for panning
195         panP = manager.Value("f", 0.025)
196         panI = manager.Value("f", 0.000008)
```

```
197         panD = manager.Value("f", 0.0000009)
198
199         #set PID values for tilting
200         tiltP = manager.Value("f", 0.022)
201         tiltI = manager.Value("f", 0.000005)
202         tiltD = manager.Value("f", 0.0000002)
203
204         Video_Source = manager.Value("i",0)
205         #we have 4 independent processes
206         #1. Detector_Tracking - finds/localizes the object
207         #2. panning           - PID control loop determines panning angle
208         #3. tilting           - PID control loop determines tilting angle
209         #4. setServos         - drives the servos to proper angles based
210         #                       on PID feedback to keep object in center
211
212         processDetectorTracking = Process(target=Detector_Tracking, args=(con.main_args, objX, objY,
Frame_Center_X, Frame_Center_Y, Video_Source))
213         processPanning = Process(target=Pid_Parameters_Calculation, args=(pan, panP, panI, panD,
objX, Frame_Center_X))
```

```
214         processTilting = Process(target=Pid_Parameters_Calculation, args=(tlt, tiltP, tiltI, tiltD, objY, Frame_Center_Y))
215         processSetServos = Process(target=Drive_Servos, args=(pan, tlt))
216
217         #start all 4 processes
218         processDetectorTracking.start()
219         processPanning.start()
220         processTilting.start()
221         processSetServos.start()
222
223         #join all 4 processes
224         processDetectorTracking.join()
225         processPanning.join()
226         processTilting.join()
227         processSetServos.join()
228         #disable the servos
229         pth.servo_enable(con.Servo_Pan, False)
230         pth.servo_enable(con.Servo_Tilt, False)
```

Το αρχείο constants.py για την τελική έκδοση του συστήματος έχει ανανεωθεί σε σχέση με αυτό του Τμήμα κώδικα 5 μετά την προσθήκη της γραμμής 7 η οποία ορίζει σε dictionary με όνομα main_args όλα τα βασικά ορίσματα του κυρίως προγράμματος όπως το path του αρχείου βίντεο, τα path των αρχείων του μοντέλου η ελάχιστη αποδεκτή βεβαιότητα και η επιλογή εκτέλεσης σε επεξεργαστή movidius. Να σημειωθεί ότι σε προηγούμενες εκδόσεις του προγράμματος που αναλύθηκαν σε προηγούμενα κεφάλαια το dictionary αυτό αρχικοποιούταν από τα ορίσματα που πέρναγε ο χρήστης στη γραμμή εντολών για την εκτέλεση του προγράμματος. Στην έκδοση αυτή με σκοπό τη διευκόλυνση του χρήστη για την εκτέλεση του προγράμματος αρκεί να εκτελεστεί το βασικό αρχείο σε περιβάλλον python χωρίς να απαιτείται κάποιο επιπρόσθετο όρισμα.

Τμήμα κώδικα 7 Ανανεωμένο constants.py αρχείο

```
1  Servo_Pan = 1
2  Servo_Tilt = 2
3  Minimum_Delay = 1
4  Servo_Angle_Range = (-45, 45)
5  System_Title = "Automated Tank Detector (ATD)"
6  Final_Width = 640
7  main_args= {'vFile': None, 'prototxt': 'MobileNetSSD_deploy.prototxt',
   'model': 'MobileNetSSD_deploy.caffemodel', 'confidence': 0.50, 'movidi-
   us': 1}
```

ΚΕΦΑΛΑΙΟ 5
Παρουσίαση συστήματος αυτόματης παρακολούθησης αντικειμένου

5.1 Εγχειρίδιο συστήματος

5.1.1 Έλεγχος λειτουργίας εξαρτημάτων και συναρμολόγηση

5.1.1.1 Λειτουργικό σύστημα

Δεδομένης της αρχιτεκτονικής του προτεινόμενου συστήματος που βασίζεται σε Raspberry Pi, το λειτουργικό του συστήματος έχει επιλεγεί να είναι το Raspbian. Το λειτουργικό αυτό σύστημα εγκαθίσταται σε μία SD Card η οποία εισάγεται στην ειδική για αυτό το σκοπό θύρα του Raspberry. Το προτεινόμενο σύστημα χρησιμοποιεί μία SD Card χωρητικότητας 64 GB, αλλά για λόγους οικονομίας ο χρήστης μπορεί να χρησιμοποιήσει μικρότερη η οποία όμως θα είναι χωρητικότητας τουλάχιστον 32 GB. Από την επίσημη ιστοσελίδα https://www.raspberrypi.com/software/ ο χρήστης μπορεί να κατεβάσει το αρχείο raspberry pi imager που θα εκτελέσει με σκοπό την εγκατάσταση του λειτουργικού συστήματος για raspberry pi στην SD Card. Για να καταστεί αυτό εφικτό η κάρτα συνδέεται με τον υπολογιστή που έχει το προαναφερθέν αρχείο για την εγκατάσταση, με ένα Card Reader , που είναι σε μορφή USB flash drive ή μία μικρή συσκευή που συνδέεται με USB θύρα. Κατά την εκτέλεση του imager επιλέγουμε αρχικά «CHOOSE OS», Raspberry Pi OS(32-bit). Στη συνέχεια στην επιλογή «CHOOSE STORAGE» προσδιορίζουμε την SD Card που έχουμε συνδέσει για το σκοπό αυτό στον υπολογιστή μας. Μετά την εμφάνιση της επιλογής «WRITE», όταν πατηθεί εκκινεί η διαδικασία εγγραφής στην SD Card η οποία διαρκεί μερικά λεπτά. Όταν ολοκληρωθεί η εγκατάσταση εκτελείται η ασφαλής κατάργηση της SD Card από τον υπολογιστή και στη συνέχεια αυτή τοποθετείται στο Raspberry Pi. Στις θύρες USB του Raspberry Pi μπορούν να συνδεθούν πληκτρολόγιο και ποντίκι, ενώ στη θύρα HDMI εξωτερική οθόνη. Όταν τροφοδοτηθεί το Raspberry Pi με 5V, 2.5A θα εκκινήσει το λειτουργικό σύστημα Raspbian και ο χρήστης έχει τη δυνατότητα να λειτουργήσει το Raspberry ως έναν υπολογιστή με λειτουργικό Raspbian.

5.1.1.2 Έλεγχος Pantilt hat

Μετά την αρχική εγκατάσταση του λειτουργικού συστήματος στο Raspberry βασικό εξάρτημα που πρέπει να ελεγχθεί η λειτουργία του είναι το gimbal της κάμερας το οποίο στο προτεινόμενο σύστημα υλοποιείται με ένα pantilt hat της pimoroni (βλ. Εικόνα 6). Για το σκοπό αυτό συνδέουμε το pantilt hat με την πλακέτα του raspberry με τη μεθοδολογία που περιγράφεται στην ενότητα 5.1.1.4. Αρχικά, πρέπει να γίνει η εγκατάσταση του πακέτου `pantilthat` με κατάλληλη pip install εντολή όπως φαίνεται παρακάτω

```
$pip install pantilthat
```

Στη συνέχεια αναπτύχθηκε σε περιβάλλον python και editor thonny το Τμήμα κώδικα 8 που είναι αποκλειστικά για τον έλεγχο της ορθής λειτουργίας των σερβοκινητήρων του pantilt hat. Η ορθή λειτουργία των σερβοκινητήρων πιστοποιείται αν κατά την εκτέλεση του παρακάτω κώδικα διαπιστωθεί κίνηση αρχικά στον οριζόντιο άξονα κατά 20^o κάθε ένα δευτερόλεπτο ξεκινώντας από γωνία -90 έως γωνία +90,και στη συνέχεια αφού επανέλθει σε γωνία 0 στον οριζόντιο άξονα πραγματοποιείται κίνηση στον κάθετο άξονα κατά 20^o κάθε ένα δευτερόλεπτο ξεκινώντας από γωνία -90 έως γωνία +90. Μετά την ολοκλήρωση της διαδικασίας αυτής αφαιρείται το pantilt hat με σκοπό να ελεγχθούν τα υπόλοιπα module όπως αυτό θα αναλυθεί στις επόμενες ενότητες. Το Τμήμα κώδικα 8 αναλύεται λεπτομερώς παρακάτω.

Τμήμα κώδικα 8 Τεστ λειτουργίας σερβοκινητήρων

	import pantilthat as pth
	import time
	pth.servo_enable(1, True)
	pth.servo_enable(2, True)
	panAngle=-90
	for i in range(9):
1	pth.pan(panAngle)

```
1          panAngle+=20
1          time.sleep(1)
1       pth.pan(0)
1       time.sleep(2)
1       tiltAngle=-90
1       for i in range(9):
1          pth.tilt(tiltAngle)
1          tiltAngle +=20
1          time.sleep(1)
2       pth.tilt(0)
2       pth.servo_enable(1, False)
2       pth.servo_enable(2, False)
```

Στις γραμμές 1, 2 γίνεται import των απαραίτητων πακέτων pantilthat, time. Στις γραμμές 6, 7 ενεργοποιούνται οι σερβοκινητήρες, επειδή είναι απενεργοποιημένοι. Στο πρόγραμμα αυτό υπάρχουν δύο βασικές μεταβλητές οι panAngle και tiltAngle στις οποίες αποθηκεύεται η γωνία των κινητήρων σε κάθε άξονα. Στις γραμμές 8, 15 αρχικοποιούνται οι μεταβλητές αυτές στην τιμή -90 για να τεθεί ο κινητήρας οριζόντιας κίνησης και ο κινητήρας κάθετης κίνησης αντίστοιχα στο ένα άκρο της εμβέλειάς τους. Στις γραμμές 9-11 και 16-18 ορίζονται δύο δομές επανάληψης που κινούν τον αντίστοιχο κινητήρα ανά 20 μοίρες. Στην πράξη αυξάνεται η τιμή των μεταβλητών panAngle και tiltAngle κατά 20^o και καλούνται οι μέθοδοι pth.pan και pth.tilt με παράμετρο την αντίστοιχη μεταβλητή. Στις γραμμές 12,19 δημιουργείται καθυστέρηση ίση με ένα δευτερόλεπτο μεταξύ των μεταβολών γωνίας των κινητήρων. Στις γραμμές 13 και 20 η θέση των σερβοκινητήρων τίθεται σε γωνία μηδέν. Τέλος στις γραμμές 21,22 απενεργοποιούνται οι σερβοκινητήρες και το πρόγραμμα τερματίζει.

5.1.1.3 Έλεγχος Pi Camera

Επόμενο module του οποίου πρέπει να ελεγχθεί η ορθή λειτουργία είναι αυτό της Pi camera. Για το σκοπό αυτό η κάμερα συνδέεται στο raspberry με την καλωδιοταινία (κατά προτίμηση μεγαλύτερη από 20 cm, βλ Εικόνα 5) στην θύρα υποδοχής καλωδιοταινίας με

την ένδειξη camera που έχει το raspberry Pi, η οποία είναι περίπου στο κέντρο της πλακέτας. Σε ένα `terminal` (Ctrl+Alt+T) πληκτρολογείται η εντολή `$sudo raspi-config` που οδηγεί σε ρυθμίσεις του firmware του raspberry. στις επιλογές διεπαφής (`interface options`) επιλέγεται η κάμερα και η ενεργοποίηση αυτής. Ενδεχομένως να απαιτηθεί επανεκκίνηση του συστήματος. Έπειτα εκκινείται ένα terminal και εκτελείται η εντολή `$raspistill -o test.jpg` για να ελεγχθεί εάν λειτουργεί σωστά η κάμερα. Η δημιουργία μίας φωτογραφίας με όνομα `test.jpg` στο working directory του terminal(αν δεν είναι γνωστό το directory, μπορεί να βρεθεί με την εντολή `pwd`) πιστοποιεί την ορθή λειτουργία της κάμερας και την επικοινωνία αυτής με το raspberry. Η κάμερα μπορεί να αφαιρεθεί.

5.1.1.4 Συναρμολόγηση

Η σειρά των βημάτων συναρμολόγησης του υλικού του συστήματος αναλύεται παρακάτω:

Βήμα 1: Τοποθέτηση κάρτας SIM με εγκατεστημένο Raspbian στην ειδική υποδοχή του Raspberry Pi(βλ. Εικόνα 2).

Βήμα 2: Τοποθέτηση του ενός άκρου της καλωδιοταινίας (βλ. Εικόνα 5) στην ειδική υποδοχή του Raspberry Pi.

Βήμα 3: Τοποθέτηση του armor case with dual fans (βλ. Εικόνα 7) στο Raspberry Pi. ΠΡΟΣΟΧΗ το άκρο της καλωδιοταινίας που δεν είναι συνδεδεμένο με το Raspberry Pi πρέπει να περάσει εντός της οπής του armor case ώστε να μείνει ελεύθερο το άκρο του στο πάνω μέρος του armor case. Στο armor case τοποθετούνται ειδικές μικρές επιφάνειες που είναι κατάλληλες για τη μόνωση του υλικού της πλακέτας με την θήκη και τοποθετούνται πάνω από τον επεξεργαστή και τη μνήμη RAM.

Βήμα 4: Επιμήκυνση των 40 pins του Raspberry Pi που προεξέχουν μετά την τοποθέτηση του armor case με χρήση header pins 20x2 (βλ. Εικόνα 8).

Βήμα 5: Τοποθέτηση του pantilt HAT (βλ. Εικόνα 6) προσέχοντας τα καλώδια των σερβοκινητήρων να συνδεθούν στα κατάλληλα pins τροφοδοσίας, γείωσης και σήματος του Raspberry Pi. Επίσης το ελεύθερο άκρο της καλωδιοταινίας περνάει μέσα από την οπή του pantilt HAT.

Βήμα 6: Τοποθέτηση του ελεύθερου άκρου της καλωδιοταινίας στην κάμερα(βλ. Εικόνα 4).

Βήμα 7: Τοποθέτηση της κάμερας στην κατάλληλη θήκη του pantilt HAT.

Βήμα 8: Εισαγωγή του movidius(βλ. Εικόνα 9) σε USB 3.0 θύρα του Raspberry Pi.

5.1.2 Εγκατάσταση OpenCV dependencies

Μετά τη συναρμολόγηση του υλικού και δεδομένου ότι στο Raspberry Pi έχει εγκατασταθεί μόνο το λειτουργικό του σύστημα είναι αναγκαία η εγκατάσταση περαιτέρω χρήσιμων πακέτων και λογισμικού όπως αυτό περιγράφεται παρακάτω από terminal:

Βήμα 1: Update και upgrade του λειτουργικού συστήματος

```
$sudo apt-get update && $sudo apt-get upgrade.
```

Βήμα 2: Εγκατάσταση εργαλείων περιβάλλοντος ανάπτυξης (development environment) συμπεριλαμβάνοντας το cmake.

```
$sudo apt-get install build-essential cmake unzip pkg-config.
```

Βήμα 3: Εγκατάσταση βιβλιοθηκών χειρισμού εικόνας και βίντεο.

```
$sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
```

```
$sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

```
$sudo apt-get install libxvidcore-dev libx264-dev
```

Βήμα 4: Εγκαθιστούμε το Gnome Tool Kit (GTK) για παραγωγή GUI περιβάλλοντος και μια βιβλιοθήκη η οποία μειώνει τα ενοχλητικά προειδοποιητικά μηνύματα της GTK.

```
$sudo apt-get install libgtk-3-dev
```

```
$sudo apt-get install libcanberra-gtk*
```

Βήμα 5: Εγκατάσταση δύο πακέτων που περιέχουν αριθμητικές βελτιστοποιήσεις για το OpenCV από βιβλιοθήκες fortran

```
$ sudo apt-get install libatlas-base-dev gfortran
```

Βήμα 6: Εγκατάσταση python3 development package

```
$ sudo apt-get install python3-dev
```

5.1.3 Λήψη του OpenCV

Αφού έχουν εγκατασταθεί επιτυχώς τα πακέτα και οι βιβλιοθήκες που αναφέρθηκαν παραπάνω, σειρά έχει η λήψη και προετοιμασία των αρχείων του OpenCV για εγκατάσταση με χρήση των κάτωθι εντολών:

```
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.0.zip
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.0.0.zip
$ unzip opencv.zip
$ unzip opencv_contrib.zip
$ mv opencv-4.0.0 opencv
$ mv opencv_contrib-4.0.0 opencv_contrib
```

5.1.4 Εγκατάσταση virtual environment

Προαιρετικά ο χρήστης μπορεί να δημιουργήσει ένα εικονικό περιβάλλον για τις παραπάνω εγκατεστημένες εφαρμογές και πακέτα αν έχει σκοπό να χρησιμοποιήσει το ίδιο σύστημα για άλλες εφαρμογές που χρησιμοποιούν άλλες εκδόσεις των ίδιων πακέτων. Αυτό είναι εφικτό με τη χρήση των κάτωθι εντολών:

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
$ sudo pip3 install virtualenv virtualenvwrapper
$ sudo rm -rf ~/get-pip.py ~/.cache/pip
$ export WORKON_HOME=$HOME/.virtualenvs
$ export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
$ source /usr/local/bin/virtualenvwrapper.sh
```

```
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
$ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
$ source ~/.profile
$ mkvirtualenv cv -p python3
```

Με την παρακάτω εντολή επιβεβαιώνουμε αν το virtual environment λειτουργεί σωστά.

```
$ workon cv
```

5.1.5 OpenCV compile and build

Δεδομένου ότι ολοκληρώθηκε το βήμα λήψης των αρχείων του OpenCV όπως αυτό περιγράφηκε στην ενότητα 5.1.3 ακολουθεί η εγκατάσταση του OpenCV σε δύο διακριτά βήματα του compile και του build όπως αυτά περιγράφονται στις δύο επόμενες υποενότητες.

5.1.5.1 Build OpenCV

Η εγκατάσταση πακέτου numpy πρέπει να προηγηθεί της διαδικασίας build του OpenCV με χρήση της παρακάτω εντολής:

```
$ pip3 install numpy
```

Στη συνέχεια γίνεται build του πακέτου OpenCV με τις παρακάτω εντολές:

```
$ cd ~/opencv
$ mkdir build
$ cd build
$ cmake \
    -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
    -D ENABLE_NEON=ON \
    -D ENABLE_VFPV3=ON \
```

```
-D BUILD_TESTS=OFF \
-D OPENCV_ENABLE_NONFREE=ON \
-D INSTALL_PYTHON_EXAMPLES=OFF \
-D BUILD_EXAMPLES=OFF ..
```

Επιβεβαίωση ότι η μεταβλητή `OPENCV_EXTRA_MODULES_PATH` δείχνει στη σωστή διαδρομή.

5.1.5.2 Compile OpenCV

Ακολουθεί η διαδικασία μεταγλώτισσης του πακέτου OpenCV ακολουθώντας τα παρακάτω βήματα:

Βήμα 1: Αλλαγή του χώρου SWAP στο Raspberry Pi. Αυτή η μέθοδος θα βοηθήσει στο να γίνει compile του OpenCV με όλους τους πυρήνες επεξεργαστή.

```
$ sudo nano /etc/dphys-swapfile
# CONF_SWAPSIZE=100
CONF_SWAPSIZE=2048
```

Αύξηση από 100 MB σε 2048 MB. Αν δεν εκτελεστεί αυτή η μέθοδος το πιο πιθανό είναι το λειτουργικό σύστημα Raspbian να σταματήσει να λειτουργεί.

```
$ sudo /etc/init.d/dphys-swapfile stop
$ sudo /etc/init.d/dphys-swapfile start
```

Βήμα 2: Compile OpenCV

```
$ make -j4
$ sudo make install
$ sudo ldconfig
```

Βήμα 3: Επαναφορά του swap space στα 100 MB με την ίδια μέθοδο.

5.1.5.3 Σύνδεση του OpenCV με το εικονικό περιβάλλον της Python

Κατόπιν της επιτυχούς εγκατάστασης του OpenCV είναι εφικτή η σύνδεσή του με το εικονικό περιβάλλον της Python με χρήση των παρακάτω εντολών:

```
$ sudo make install
$ sudo ldconfig
```

Τέλος απαιτείται η εγκατάσταση των πρόσθετων αυτών πακέτων

```
$pip3 install dropbox
$pip3 install imutils
$pip3 install "picamera[array]"
```

Στο σημείο αυτό έχει γίνει η εγκατάσταση όλων των απαραίτητων πακέτων για να μπορεί να εκτελεστεί πρόγραμμα σε python3 που χρησιμοποιεί OpenCV στο Raspberry Pi. Ακολουθεί στην επόμενη ενότητα η εγκατάσταση πακέτων για τη λειτουργική επικοινωνία του Raspberry με το movidius.

5.1.6 Εγκατάσταση OpenVINO dependencies

Το λογισμικό OpenVINO θα εγκατασταθεί με σκοπό να εξυπηρετήσει την επικοινωνία του Raspberry με το movidius. Για το σκοπό αυτό θα εγκατασταθούν αρχικά βασικά dependencies. Τα βήματα 1-5 της παρούσας ενότητας μπορούν να παραληφθούν αν η εγκατάσταση του OpenVINO εκτελεστεί σειριακά, όπως και προτείνεται, μετά την εγκατάσταση του OpenCV.

Βήμα 1: Κάνουμε ενημέρωση και αναβάθμιση λειτουργικού συστήματος

```
$ sudo apt-get update && sudo apt-get upgrade
```

Βήμα 2: Εγκαθιστούμε εργαλεία προγραμματιστή

```
$ sudo apt-get install build-essential cmake unzip pkg-config
```

Βήμα 3: Εγκατάσταση βιβλιοθηκών απαραίτητες για την αναγνώριση εικόνων και βίντεο

```
$sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
$sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

Βήμα 4: Εγκατάσταση GTK

```
$ sudo apt-get install libgtk-3-dev
$ sudo apt-get install libcanberra-gtk*
```

Βήμα 5: Εγκατάσταση πακέτων αριθμητικών βελτιστοποιήσεων

```
$ sudo apt-get install libatlas-base-dev gfortran
```

Βήμα 6: Εγκατάσταση κεφαλίδων python3

```
$ sudo apt-get install python3-dev
```

5.1.7 Εγκατάσταση OpenVINO

Μετά την εγκατάσταση των απαραίτητων dependencies ακολουθεί η εγκατάσταση του OpenVINO με χρήση των παρακάτω βημάτων:

Βήμα 1: Δρομολογούμε την κονσόλα να δείχνει στον φάκελο home

```
$ cd ~
```

Βήμα 2: Λήψη της εργαλειοθήκης OpenVINO

```
$ wget https://download.01.org/opencv/2020/openvinotoolkit/2020.1/
/l_openvino_toolkit_runtime_raspbian_p_2020.1.023.tgz
```

Βήμα 3: Αποσυμπίεση και μετονομασία του ληφθέντος αρχείου

```
$ tar -xf l_openvino_toolkit_runtime_raspbian_p_2020.1.023.tgz
$ mv l_openvino_toolkit_runtime_raspbian_p_2020.1.023 openvino
```

Βήμα 4: Διαμόρφωση ρυθμίσεων ώστε να ενεργοποιούνται οι μεταβλητές του OpenVINO με κάθε άνοιγμα κονσόλας.

```
$ sudo nano ~/.bashrc
```

Στο τέλος του αρχείου γράφουμε αυτή τη γραμμή

```
source ~/openvino/bin/setupvars.sh
```

Αποθήκευση του bashrc και στην κονσόλα εκτελούμε την εντολή

```
$ source ~/.bashrc
```

Βήμα 5: Ρύθμιση κανόνων για την αναγνώριση του Movidius από το Raspbian.

```
$ sudo usermod -a -G users "$(whoami)"
$ cd ~
$ sh openvino/install_dependencies/install_NCS_udev_rules.sh
```

Βήμα 6: Δημιουργία εικονικού περιβάλλοντος OpenVINO

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/get-pip.py ~/.cache/pip
$ sudo nano ~/.bashrc
```

Στο αρχείο bashrc προσθέτουμε τις παρακάτω γραμμές

```
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
VIRTUALENVWRAPPER_ENV_BIN_DIR=bin
```

Το αποθηκεύουμε και μετά εκτελούμε την εντολή

```
$ source ~/.bashrc
```

Δημιουργούμε εικονικό περιβάλλον με την εντολή

```
$ mkvirtualenv openvino -p python3
```

Βήμα 7: Εγκατάσταση πακέτων σε περιβάλλον OpenVINO

```
$ workon openvino
$ pip3 install numpy
$ pip3 install "picamera[array]"
$ pip3 install imutils
```

Βήμα 8: Δημιουργία script για την έναρξη εικονικού περιβάλλοντος OpenVINO

Δημιουργία ενός νέου αρχείου με το όνομα `start_openvino.sh` και τοποθέτηση αυτού στον φάκελο `~/`.

Συμπλήρωση των παρακάτω γραμμών

```
#!/bin/bash
echo "Starting Python 3.7 with OpenCV-OpenVINO 4.2.0 bindings..."
source ~/openvino/bin/setupvars.sh
workon openvino
```

Βήμα 9: Αποθήκευση και κλείσιμο του αρχείου `start_openvino.sh`, και εκτέλεση της παρακάτω εντολής

```
$ source ~/start_openvino.sh
```

Στο σημείο αυτό έχει επιτευχθεί η εγκατάσταση όλου του απαραίτητου λογισμικού στο Raspberry Pi για την εκτέλεση του κώδικα που αναπτύχθηκε στη διπλωματική εργασία και η επικοινωνία του με το movidius. Εκκρεμεί όμως η δημιουργία μετά από εκπαίδευση του μοντέλου το οποίο θα εγκατασταθεί και θα εκτελεστεί στον συνεπεξεργαστή movidius το οποίο αναλύεται στην παρακάτω ενότητα.

5.1.8 Ανάπτυξη μοντέλου μηχανικής μάθησης

5.1.8.1 Επιλογή περιβάλλοντος και λειτουργικού συστήματος

Όπως έχει προαναφερθεί στην ενότητα 4.3 έχει επιλεχθεί η πλατφόρμα caffe-cpu. Η εγκατάσταση παρόλο που μπορεί να γίνει σε οποιοδήποτε λειτουργικό σύστημα χρήζει ιδιαίτερης προσοχής γιατί δεν είναι απλή και οποιοδήποτε λάθος μπορεί να οδηγήσει σε επανεγκατάσταση ολόκληρης της πλατφόρμας. Οπότε έχει επιλεχθεί να εγκατασταθεί σε virtual environment ώστε να υπάρχει η δυνατότητα εύκολης επαναφοράς σε προηγούμενη κατάσταση του λειτουργικού συστήματος εφόσον ο χρήστης έχει ήδη δημιουργήσει το σημείο επαναφοράς. Η τεχνολογία αυτή ονομάζεται snapshot. Η πλατφόρμα που

προτείνεται για την εγκατάσταση της εικονικής μηχανής (virtual machine) είναι το virtual box και το λειτουργικό σύστημα είναι το Ubuntu 20.04 LTS[25].

5.1.8.2 Εγκατάσταση βιβλιοθηκών του caffe-cpu

Όλες οι χρήσιμες βιβλιοθήκες για τη δημιουργία του μοντέλου μηχανικής μάθησης μπορούν να εγκατασταθούν ακολουθώντας τα παρακάτω βήματα σε terminal

Βήμα 1: Εκτέλεση της εντολής `NUMBER_OF_CORES=nproc`. Η εντολή nproc ανιχνεύει πόσους πυρήνες έχει ο επεξεργαστής του λειτουργικού συστήματος. Στη συνέχεια ο αριθμός των πυρήνων αποθηκεύεται στη μεταβλητή `NUMBER_OF_CORES` η οποία θα χρησιμοποιηθεί στη συνέχεια.

Βήμα 2: Ενημέρωση του λειτουργικού συστήματος.

```
$sudo apt-get update
```

Βήμα 3: Η εντολή `noninteractive` χρησιμοποιείται για να μην υπάρχει διαδραστικότητα του συστήματος με τον χρήστη πληκτρολογόντας αυτόματα όλες τις προεπιλεγμένες επιλογές, δηλαδή να μην σταματάει την εγκατάσταση των βιβλιοθηκών αναμένοντας από τον χρήστη να πατήσει κάποιο πλήκτρο. Η επιλογή `Dpkg::Options::="--force-confold` είναι για να αναγκάσει το σύστημα στην περίπτωση που η βιβλιοθήκη ή το πακέτο είναι ήδη εγκατεστημένα να μην αλλαχθούν οι εκδόσεις τους.

```
$sudo DEBIAN_FRONTEND=noninteractive apt-get upgrade -y -q -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold"
```

Βήμα 4:
```
$sudo apt-get install -y libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev
```

Βήμα 5:
```
$sudo apt-get install -y --no-install-recommends libboost-all-dev
```

Βήμα 6:
```
$sudo apt-get install -y libatlas-base-dev
```

Βήμα 7:
```
$sudo apt-get install -y python-dev
```

Βήμα 8:
```
$sudo apt-get install -y python3-pip git
```

Βήμα 9: Εγκατάσταση του πακέτου lmdb

```
$git clone https://github.com/LMDB/lmdb.git
$cd lmdb/libraries/liblmdb
$sudo make
$sudo make install
```

Βήμα 10:
```
sudo apt-get install -y cmake unzip doxygen
```

Βήμα 11:
```
sudo apt-get install -y protobuf-compiler
```

Βήμα 12: `sudo apt-get install -y libffi-dev python-dev build-essential`

Βήμα 13: `sudo pip3 install lmdb`

Βήμα 14: `sudo pip3 install numpy`

Βήμα 15: `sudo apt-get install -y python-numpy`

Βήμα 16: `sudo apt-get install -y gfortran # required by scipy`

Βήμα 17: `sudo pip3 install scipy`

`Εναλλακτικά $sudo apt-get install -y python3-scipy`

Βήμα 18: `sudo apt-get install -y python-nose`

Βήμα 19: `sudo pip3 install scikit-image`

5.1.8.3 Εγκατάσταση caffe

Βήμα 1: Λήψη των χρήσιμων αρχείων και προετοιμασία αυτών.

```
$cd ~
$mkdir caffe
$cd caffe
$wget https://github.com/BVLC/caffe/archive/master.zip
$unzip -o master.zip
$cd caffe-master
```

Βήμα 2: Προετοιμασία σύνδεσης με python

```
$cd python
$for req in $(cat requirements.txt); do sudo pip install $req; done
$echo "export PYTHONPATH=$(pwd):$PYTHONPATH " >> ~/.bash_profile #
$source ~/.bash_profile
$cd ..
```

Βήμα 3: Μερικές επιπλέον παραμετροποιήσεις

```
$cp Makefile.config.example Makefile.config
$sudo gedit MakeFile.config
```

Στη γραμμή που βρίσκεται το `#CPU_ONLY := 1` σβήνουμε το `#`

Στη γραμμή που βρίσκεται το `BLAS := atlas` σβήνουμε το atlas και γράφουμε open, δηλαδή `BLAS := open`.

```
$echo "export OPENBLAS_NUM_THREADS=($NUMBER_OF_CORES)" >> ~/.bash_profile
```

Βήμα 4: Compile των caffe και pycaffe

```
$mkdir build
$cd build
$cmake ..
$cd ..
$make all -j$NUMBER_OF_CORES
$make pycaffe -j$NUMBER_OF_CORES
$make test
$make runtest
```

Στην περίπτωση που αποτυχημένης εκτέλεσης του build και χρειαστεί να γίνει εκ νέου εγκατάσταση πακέτων ή αλλαγή ρυθμίσεων κτλ, πραγματοποιούνται οι ρυθμίσεις και ξαναεκτελείται το βήμα από την αρχή αφού πρώτα μετά από μετάβαση στον κατάλογο **caffe-master directory** εκτελεστεί **make clean.**

Βήμα 5: Εγκατάσταση επιπλέον βιβλιοθηκών αν κριθεί απαραίτητο

```
$sudo pip3 install pydot
$sudo apt-get install -y graphviz
$sudo pip3 install scikit-learn
```

Βήμα 6: Ο φάκελος στον οποίο είναι εγκατεστημένο το caffe πρέπει να προσαρμοστεί στις παρακάτω διαδρομές:

```
$export OPENBLAS_NUM_THREADS=$(nproc)

$export CAFFE_ROOT=$HOME/caffe

$export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH

$export PYTHONPATH=$HOME/caffe/python:$PYTHONPATH
```

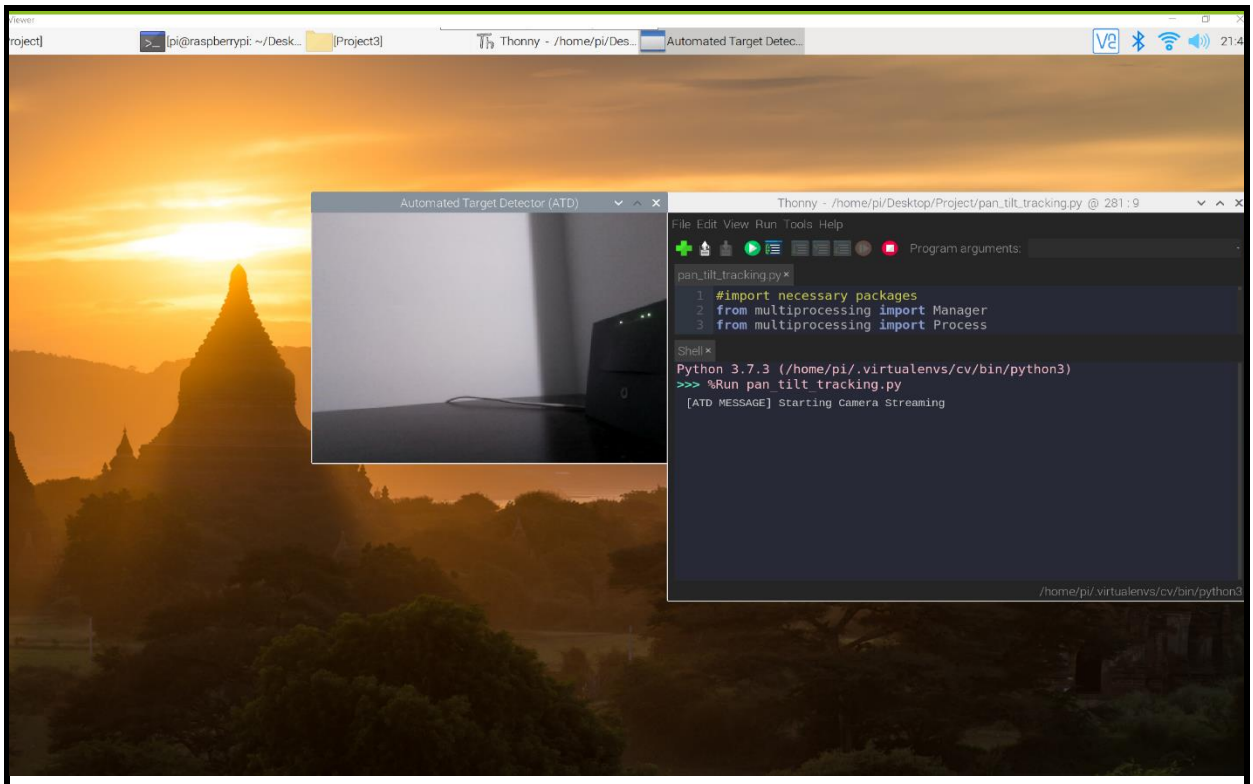
ΚΕΦΑΛΑΙΟ 6
Επίδειξη συστήματος

Στο κεφάλαιο αυτό γίνεται μία γραφική επίδειξη των συστημάτων ιχνηλάτησης αντικειμένου (βλ. Πίνακας 4) και αναγνώρισης και παρακολούθησης αντικειμένου με μηχανική μάθηση (βλ. Αφού ο χρήστης διαλέξει το αντικείμενο και πατήσει "ENTER" ένα πράσινο περίγραμμα (bounding box) εμφανίζεται γύρω από το αντικείμενο που επιλέχθηκε (βλ. Πίνακας 4δ). Το bounding box ακολουθεί το αντικείμενο καθ' όλη τη διάρκεια των κινήσεών του σε διαφορετικά σημεία του frame είτε είναι στη δεξιά πιο σκοτεινή γωνία (βλ. Πίνακας 4θ-Πίνακας 4ια) είτε είναι στην αριστερή φωτεινή γωνία (βλ. Πίνακας 4ε - Πίνακας 4η). Αξίζει να σημειωθεί ότι την κίνηση αυτή ακολουθεί και η κάμερα μέσω του gimbal. Τις μικρές κινήσεις του στόχου διορθώνει η κάμερα και αυτό είναι ορατό από τον χώρο που υπάρχει πίσω από το μαύρο αντικείμενο δεξιά που επίτηδες τοποθετήθηκε για να αναδεικνύει αυτές τις μετακινήσεις. Για παράδειγμα ο χώρος πίσω από το αντικείμενο αυτό είναι πολύ μικρότερος στο στιγμιότυπο του Πίνακας 4γ σε σχέση με το χώρο πίσω από το αντικείμενο στο στιγμιότυπο Πίνακας 4ια. Τέλος, από το σημείο που άρχισε η ιχνηλάτηση εμφανίζονται στην κάτω αριστερή γωνία του frame πληροφορίες αναφορικά με το FPS, την επιτυχή αναγνώριση ή όχι, του αλγόριθμου ιχνηλάτησης κ.ά.

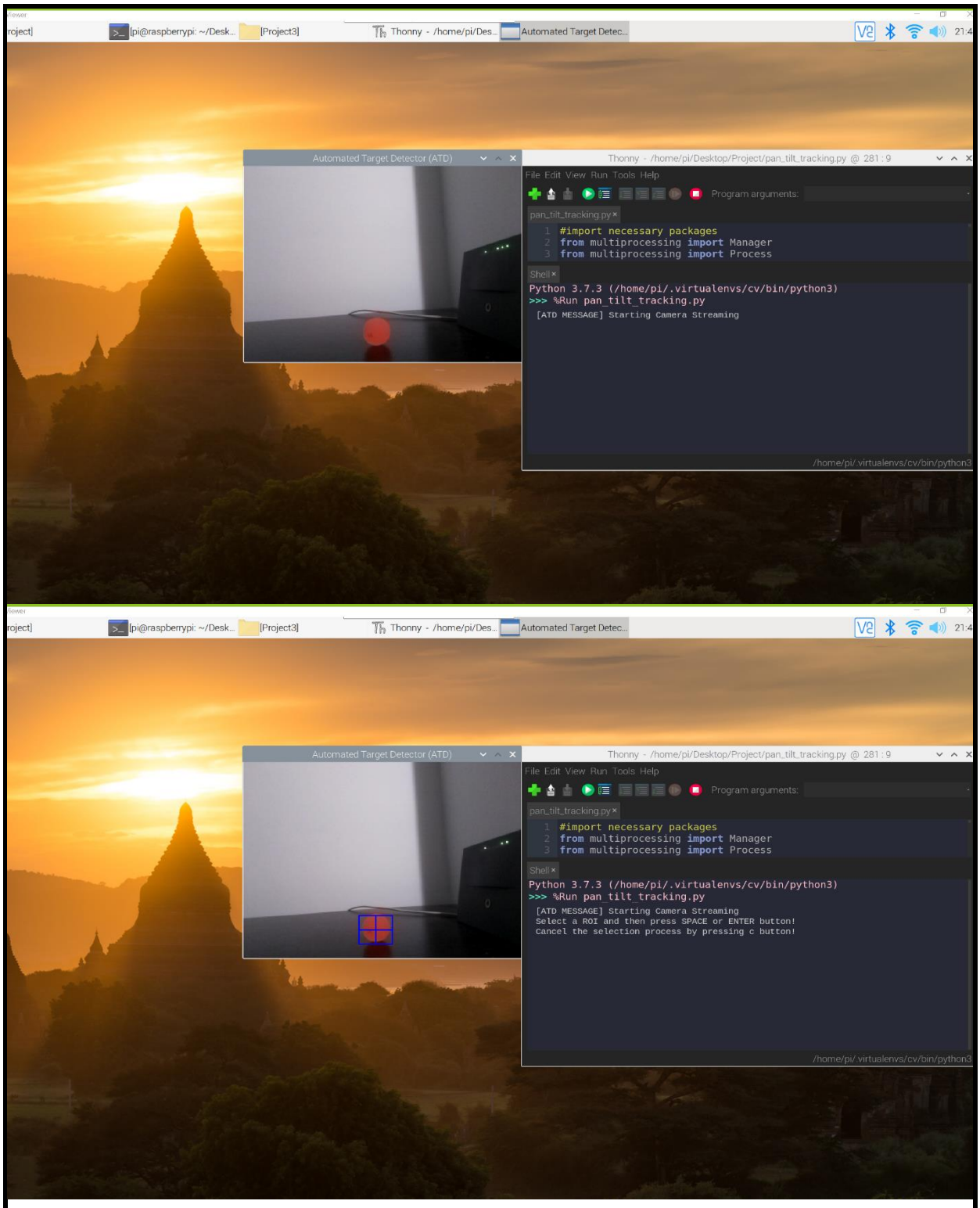
Στον Πίνακας 5 παρουσιάζονται τα στιγμιότυπα αναγνώρισης και ιχνηλάτησης στόχου με μηχανική μάθηση. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο για χάριν επίδειξης θα χρησιμοποιηθεί ένα μοντέλο που αυτόματα αναγνωρίζει μπουκάλια.

Πίνακας 5). Το πρώτο σύστημα κατά την εκκίνησή του εμφανίζει ένα παράθυρο με το τρέχον frame που διαβάζει η κάμερα του Raspberry Pi και αντίστοιχα εμφανίζεται μήνυμα στο terminal "Starting Camera Streaming"(Πίνακας 4α) . Το σύστημα συνεχίζει αυτή τη διαδικασία μέχρις ότου ο χρήστης πληκτρολογήσει "s" (Πίνακας 4γ). Όταν συμβεί αυτό ο χρήστης μπορεί να δημιουργήσει μία μπλε θηλειά γύρω από το αντικείμενο που τον ενδιαφέρει, στην προκειμένη περίπτωση για τις ανάγκες της επίδειξης το πορτοκαλί μπαλάκι που πρωτοεμφανίστηκε στον Πίνακας 4β.

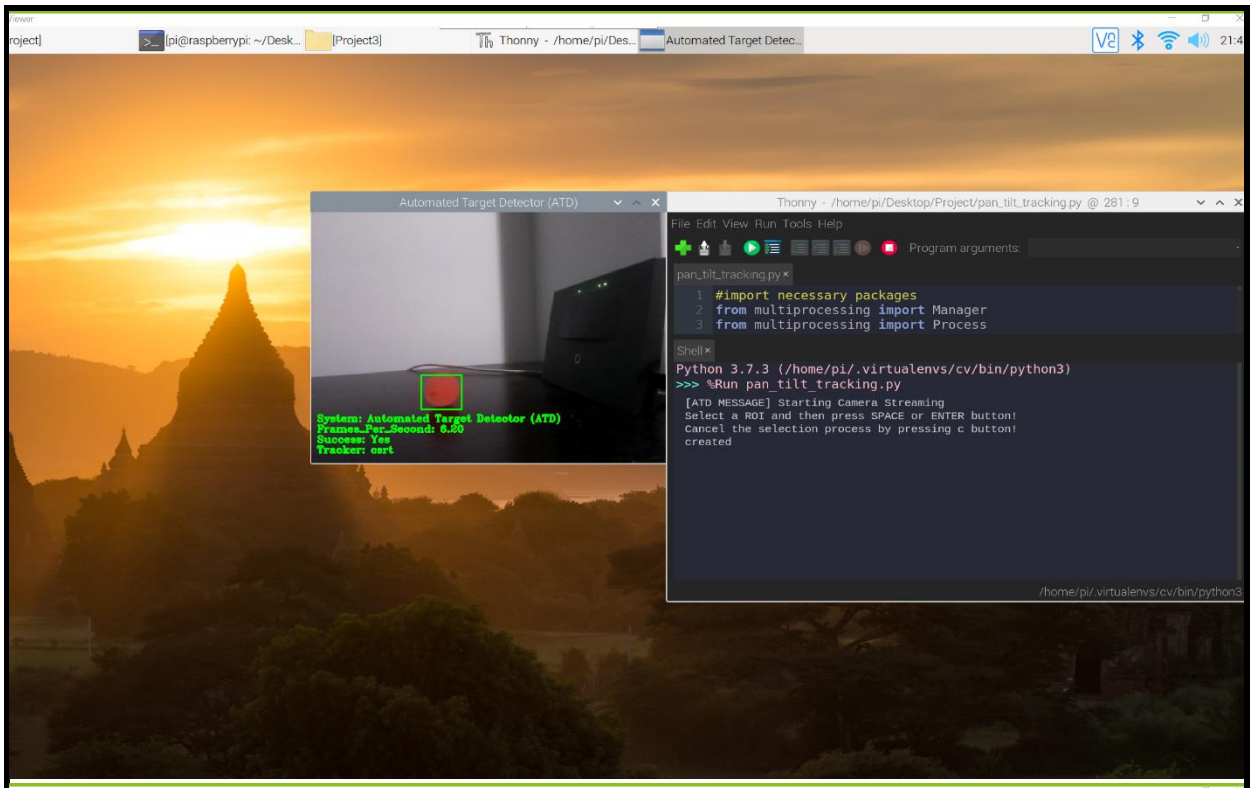
Πίνακας 4 Στιγμιότυπα ιχνηλάτησης αντικειμένου

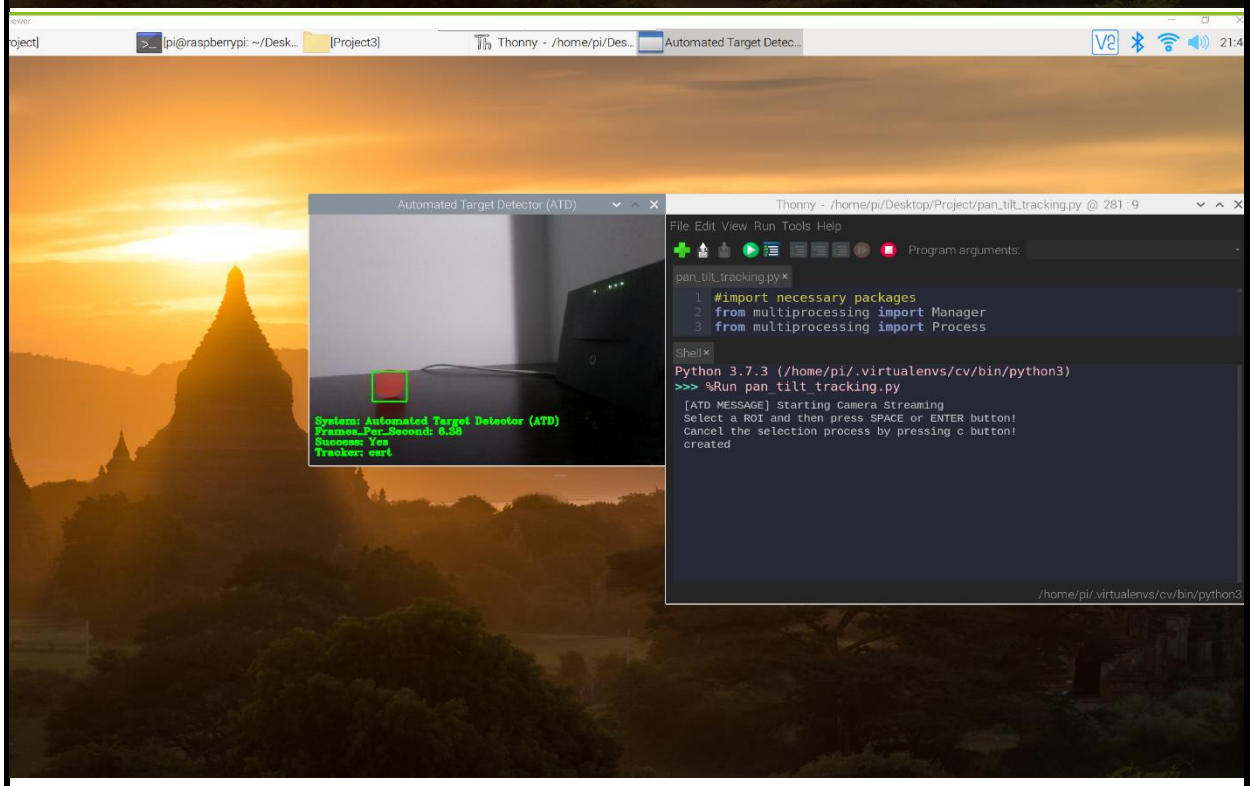

Automated Target Detector (ATD)
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming

α.

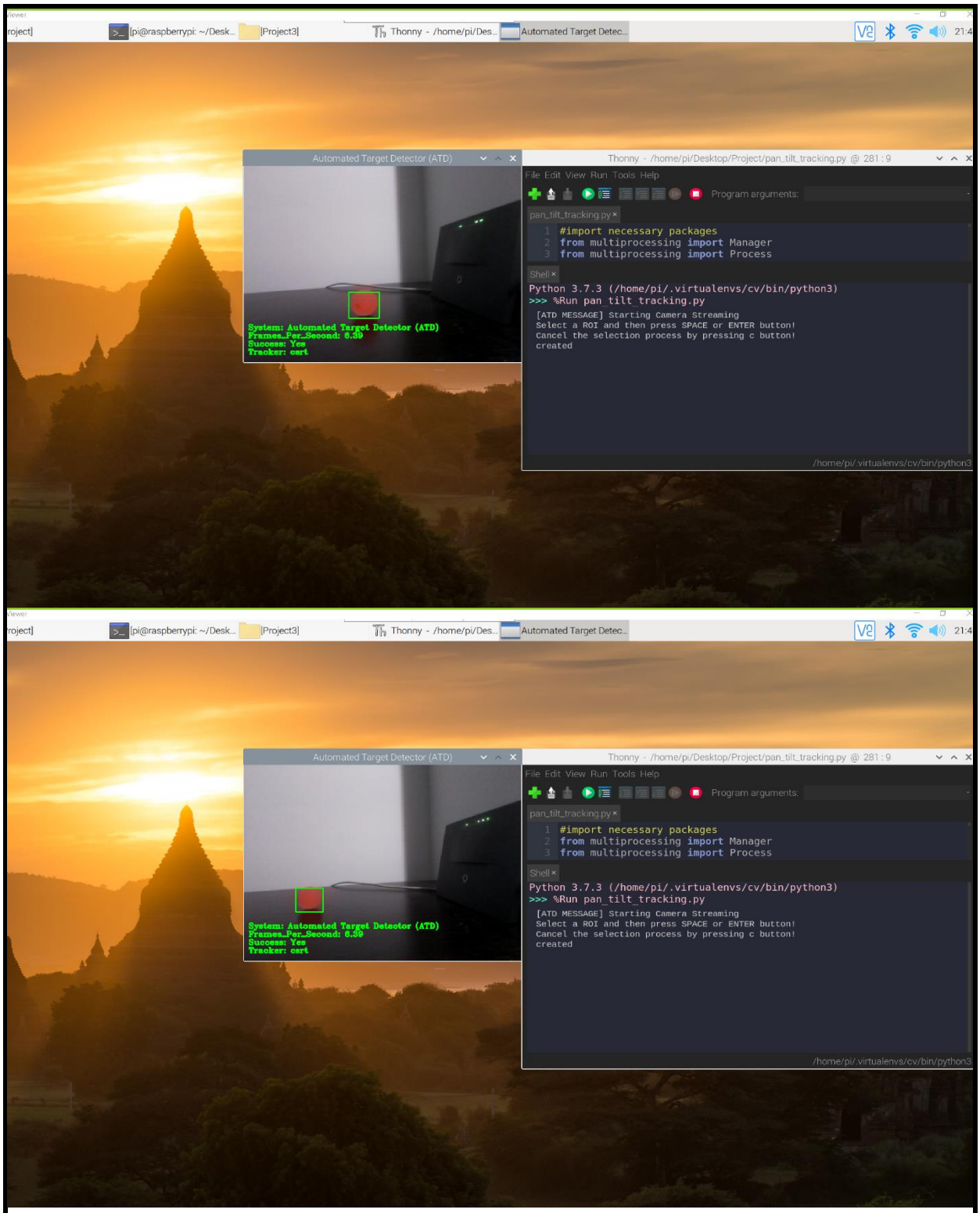

Automated Target Detector (ATD)
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
/home/pi/.virtualenvs/cv/bin/python3
Automated Target Detector (ATD)
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
/home/pi/.virtualenvs/cv/bin/python3

β,γ.

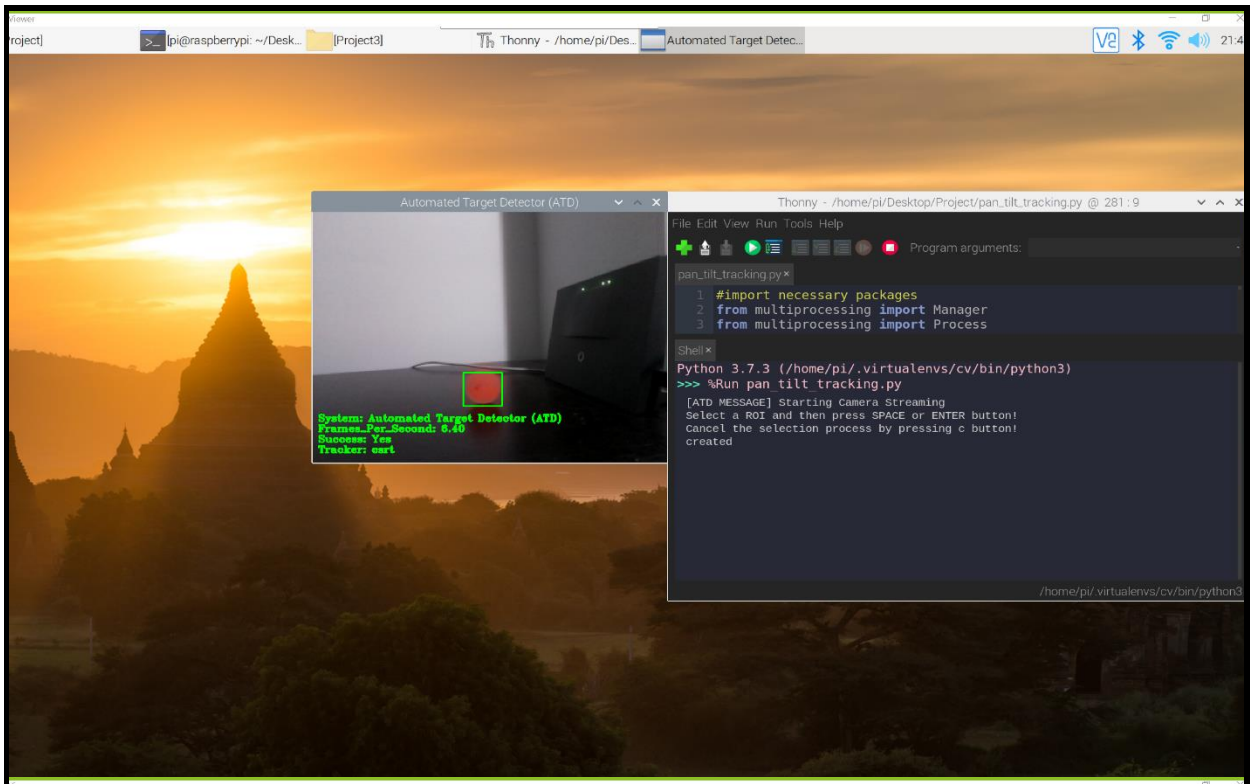

Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 8.20
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

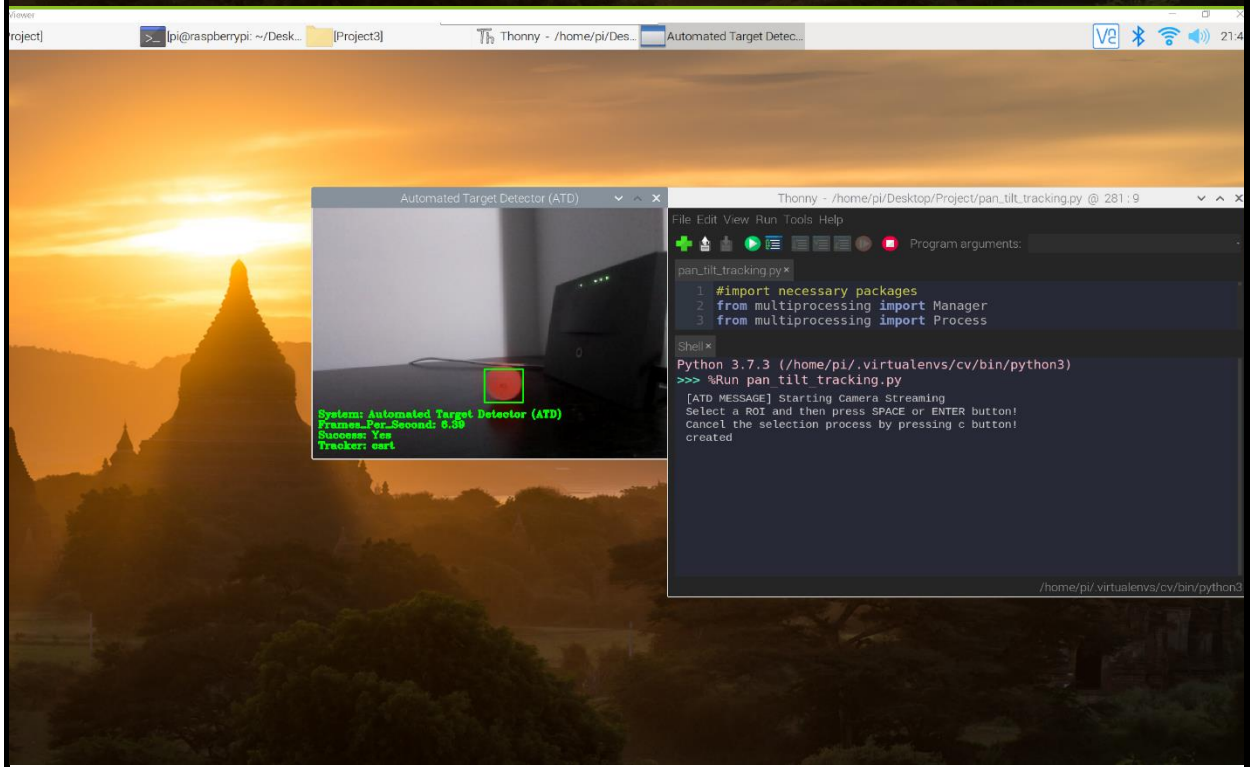

Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 8.30
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

δ,ε

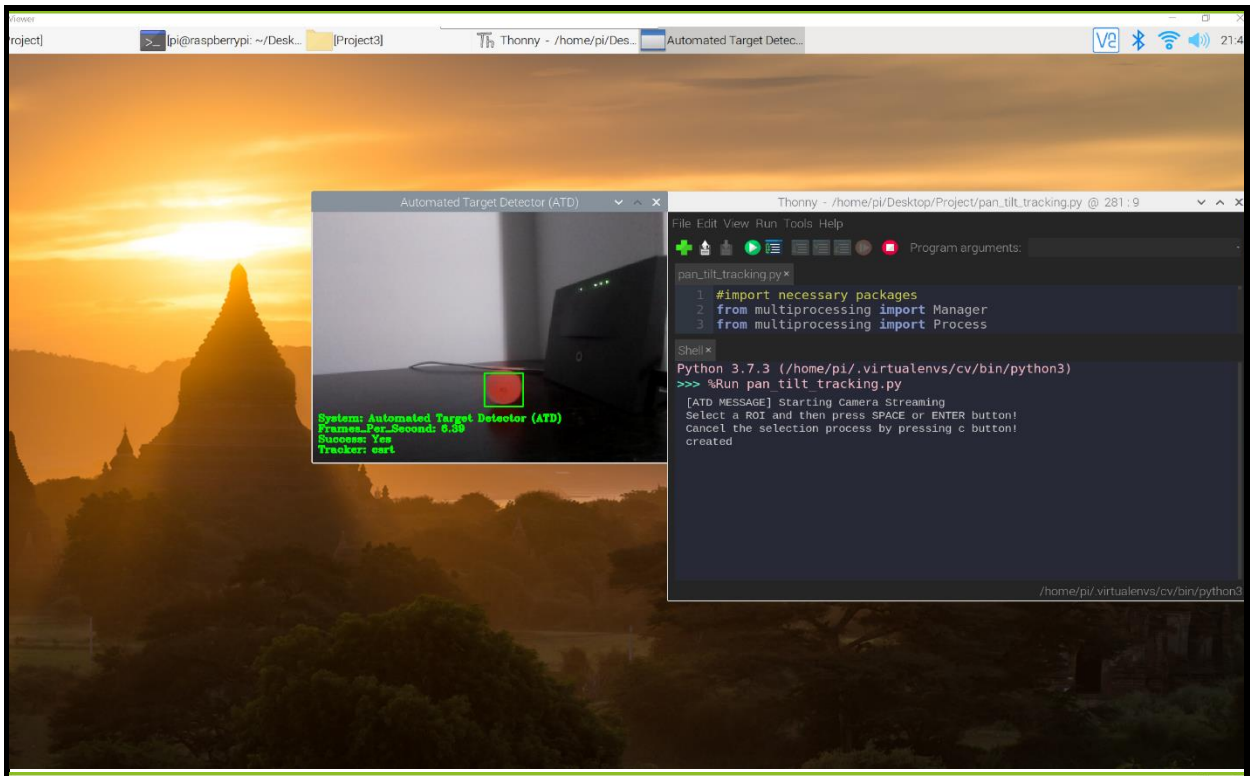

[pi@raspberrypi: ~/Desk...
[Project3]
Thonny - /home/pi/Des...
Automated Target Detec...
Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 8.39
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

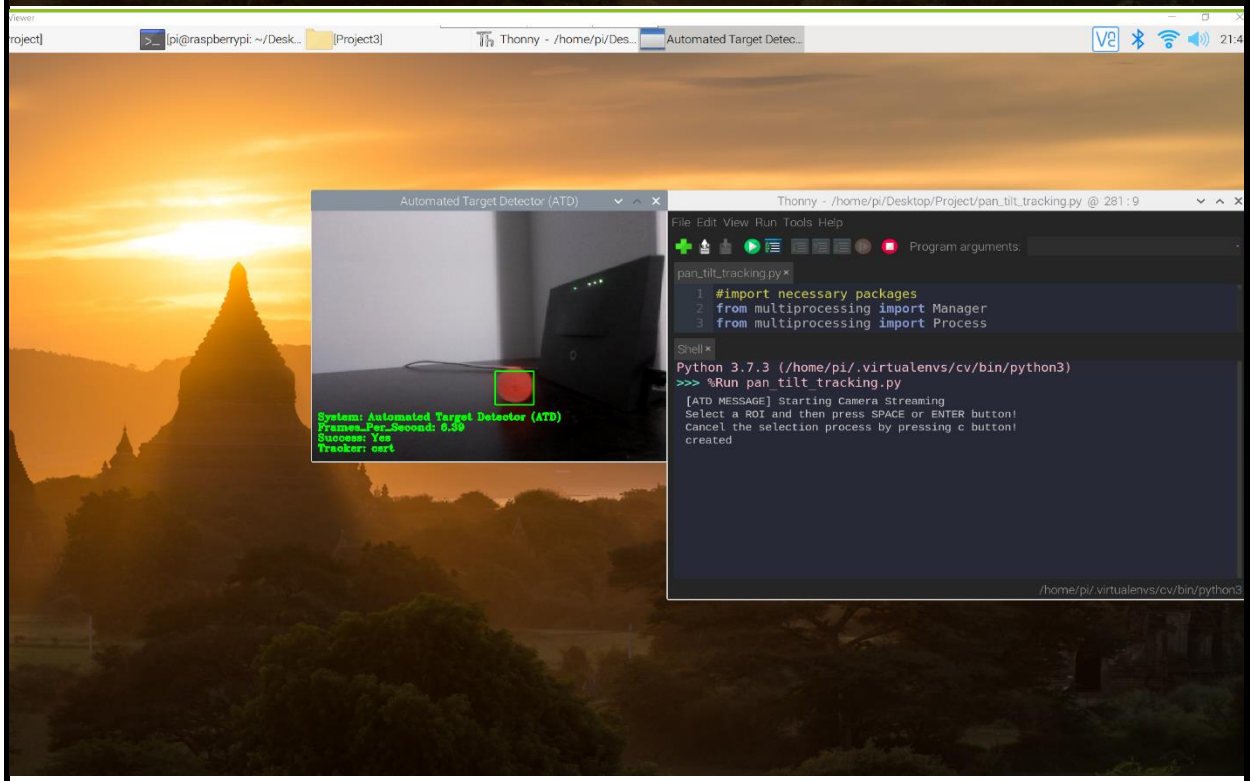
στ,ζ


Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 8.40
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3


Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 8.80
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

η,θ


Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 6.30
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

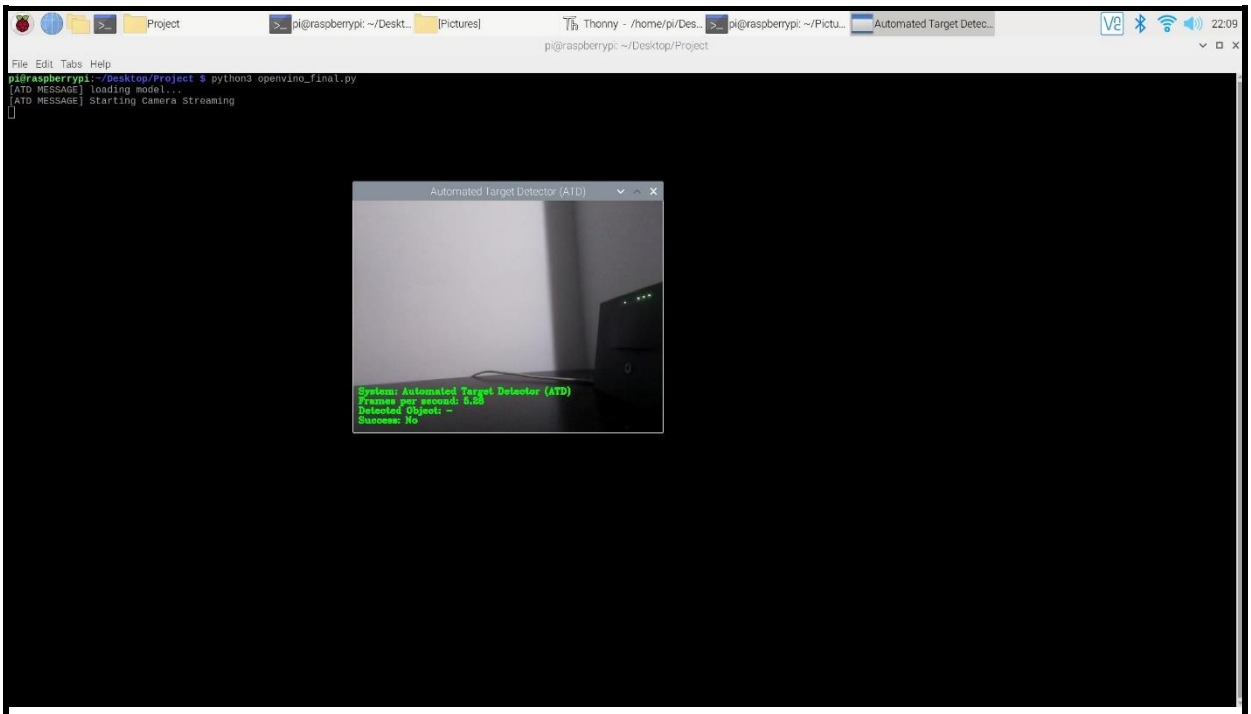

Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames_Per_Second: 6.30
Success: Yes
Tracker: csrt
Thonny - /home/pi/Desktop/Project/pan_tilt_tracking.py @ 281 : 9
File Edit View Run Tools Help
Program arguments:
pan_tilt_tracking.py
#import necessary packages
from multiprocessing import Manager
from multiprocessing import Process
Shell
Python 3.7.3 (/home/pi/.virtualenvs/cv/bin/python3)
>>> %Run pan_tilt_tracking.py
[ATD MESSAGE] Starting Camera Streaming
Select a ROI and then press SPACE or ENTER button!
Cancel the selection process by pressing c button!
created
/home/pi/.virtualenvs/cv/bin/python3

ι,ια

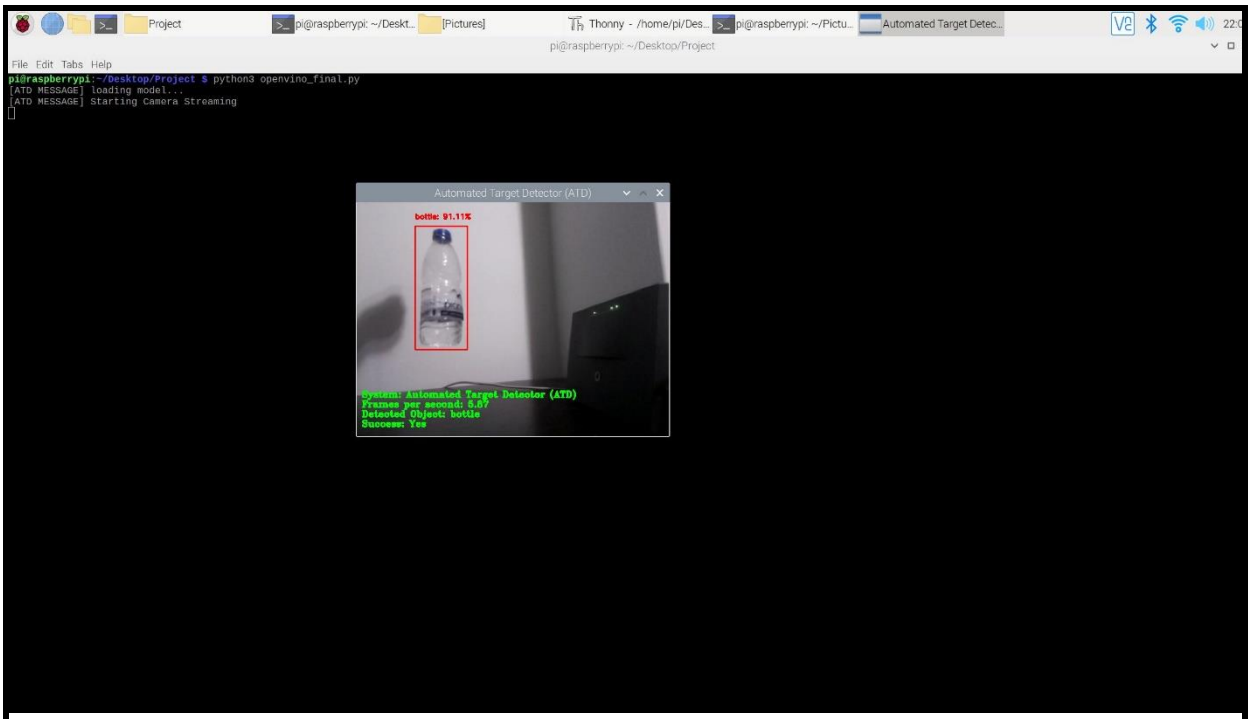
Αφού ο χρήστης διαλέξει το αντικείμενο και πατήσει "ENTER" ένα πράσινο περίγραμμα (bounding box) εμφανίζεται γύρω από το αντικείμενο που επιλέχθηκε (βλ. Πίνακας 4δ). Το bounding box ακολουθεί το αντικείμενο καθ' όλη τη διάρκεια των κινήσεών του σε διαφορετικά σημεία του frame είτε είναι στη δεξιά πιο σκοτεινή γωνία (βλ. Πίνακας 4θ-Πίνακας 4ια) είτε είναι στην αριστερή φωτεινή γωνία (βλ. Πίνακας 4ε - Πίνακας 4η). Αξίζει να σημειωθεί ότι την κίνηση αυτή ακολουθεί και η κάμερα μέσω του gimbal. Τις μικρές κινήσεις του στόχου διορθώνει η κάμερα και αυτό είναι ορατό από τον χώρο που υπάρχει πίσω από το μαύρο αντικείμενο δεξιά που επίτηδες τοποθετήθηκε για να αναδεικνύει αυτές τις μετακινήσεις. Για παράδειγμα ο χώρος πίσω από το αντικείμενο αυτό είναι πολύ μικρότερος στο στιγμιότυπο του Πίνακας 4γ σε σχέση με το χώρο πίσω από το αντικείμενο στο στιγμιότυπο Πίνακας 4ια. Τέλος, από το σημείο που άρχισε η ιχνηλάτηση εμφανίζονται στην κάτω αριστερή γωνία του frame πληροφορίες αναφορικά με το FPS, την επιτυχή αναγνώριση ή όχι, του αλγόριθμου ιχνηλάτησης κ.ά.

Στον Πίνακας 5 παρουσιάζονται τα στιγμιότυπα αναγνώρισης και ιχνηλάτησης στόχου με μηχανική μάθηση. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο για χάριν επίδειξης θα χρησιμοποιηθεί ένα μοντέλο που αυτόματα αναγνωρίζει μπουκάλια.

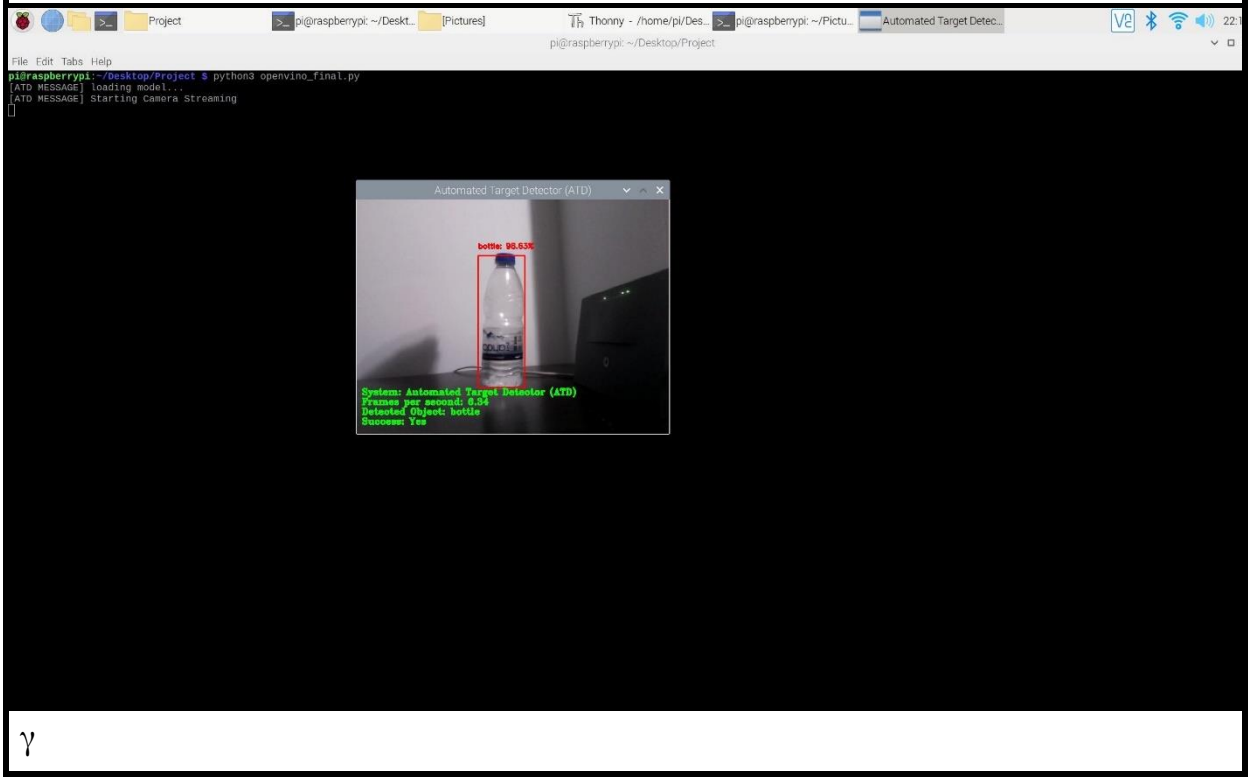
Πίνακας 5 Στιγμιότυπα αναγνώρισης και ιχνηλάτησης στόχου με μηχανική μάθηση



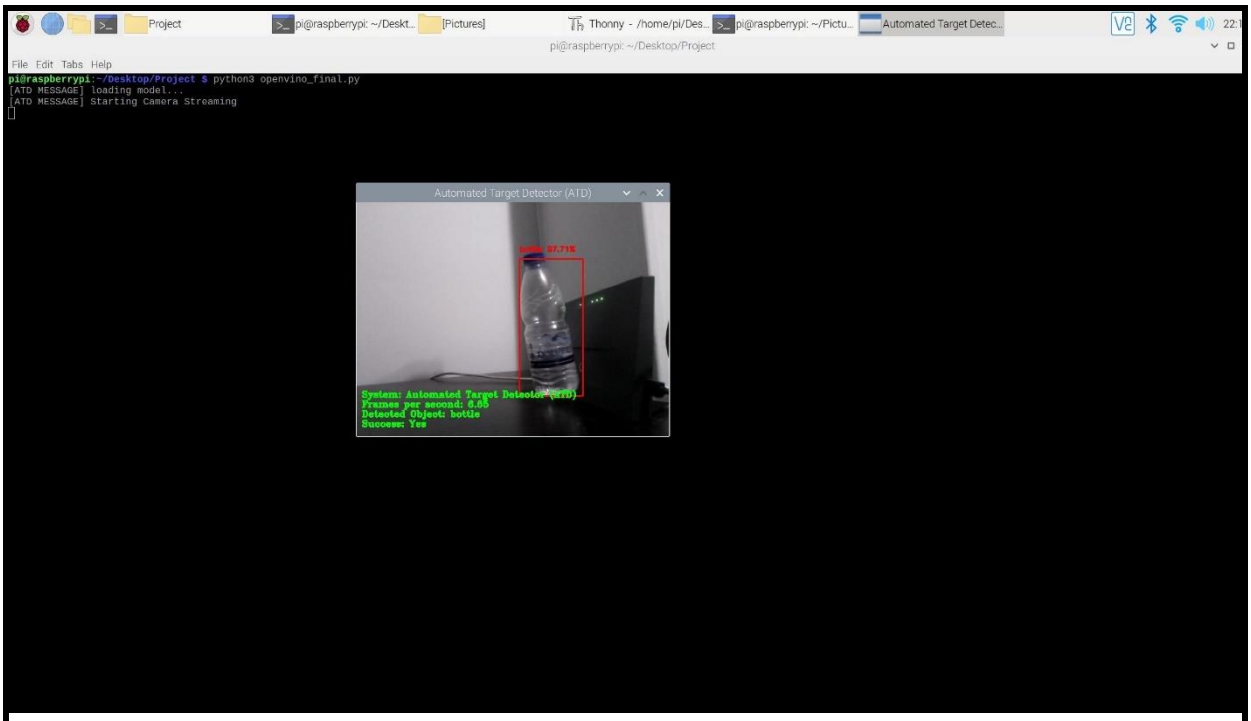
α.


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 91.11%
System: Automated Target Detector (ATD)
Frames per second: 5.67
Detected Object: bottle
Success: Yes

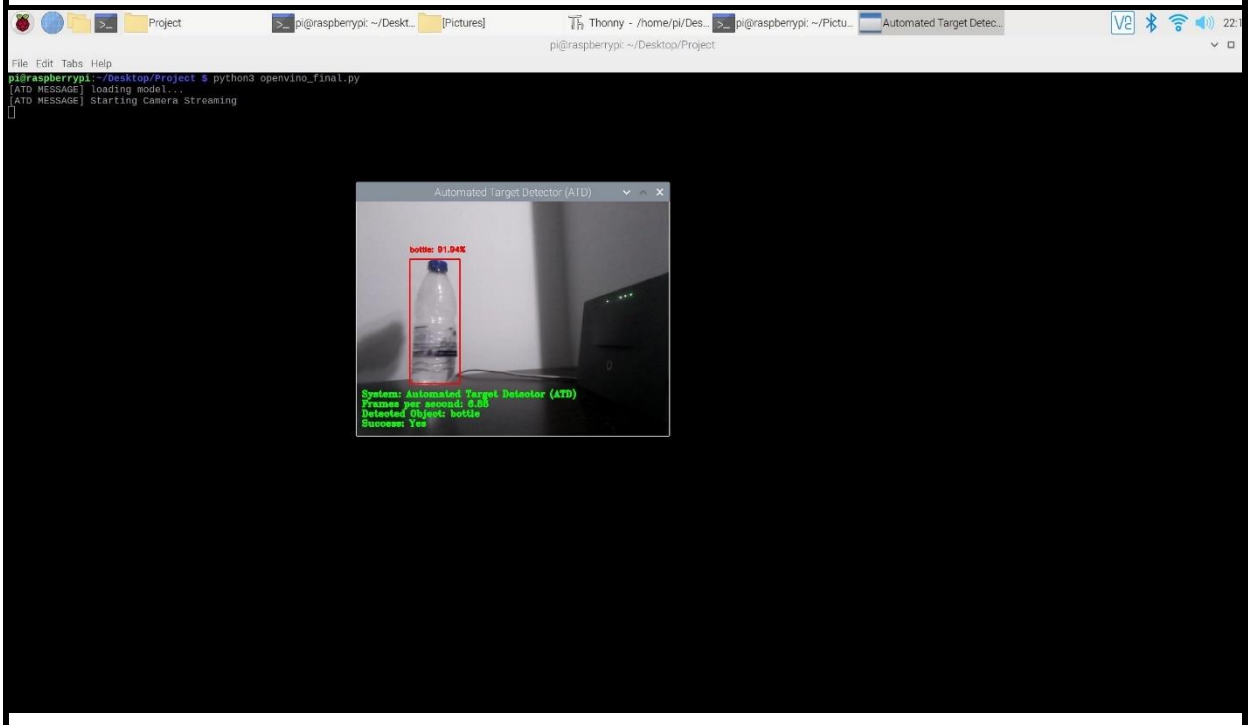
β


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 95.63%
System: Automated Target Detector (ATD)
Frames per second: 6.34
Detected Object: bottle
Success: Yes

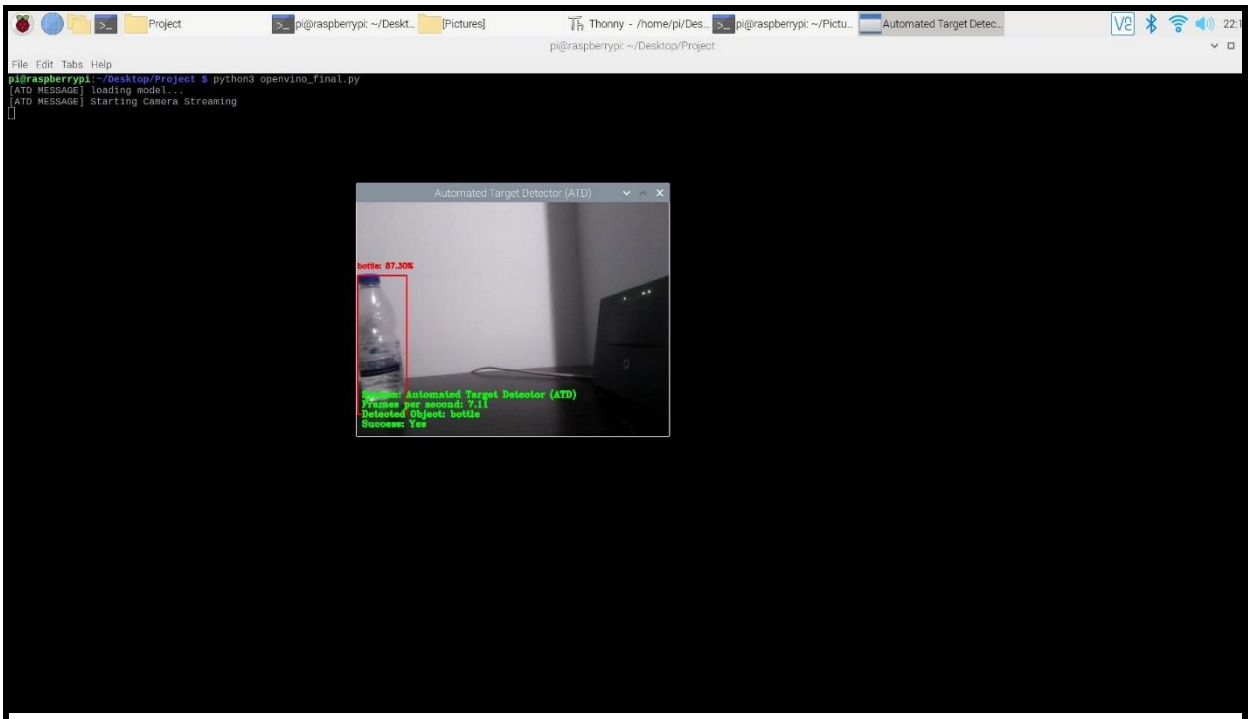
γ


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Frames per second: 0.05
Detected Object: bottle
Success: Yes

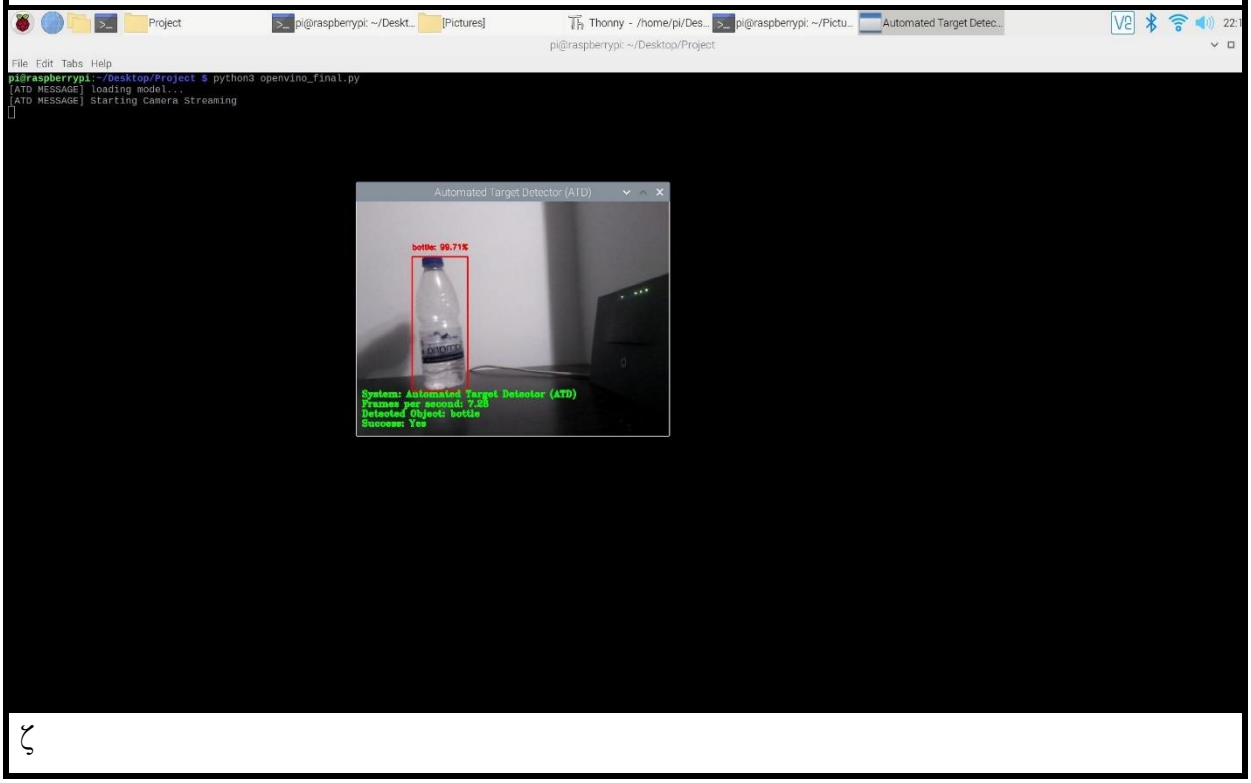
δ.


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 91.04%
System: Automated Target Detector (ATD)
Frames per second: 0.05
Detected Object: bottle
Success: Yes

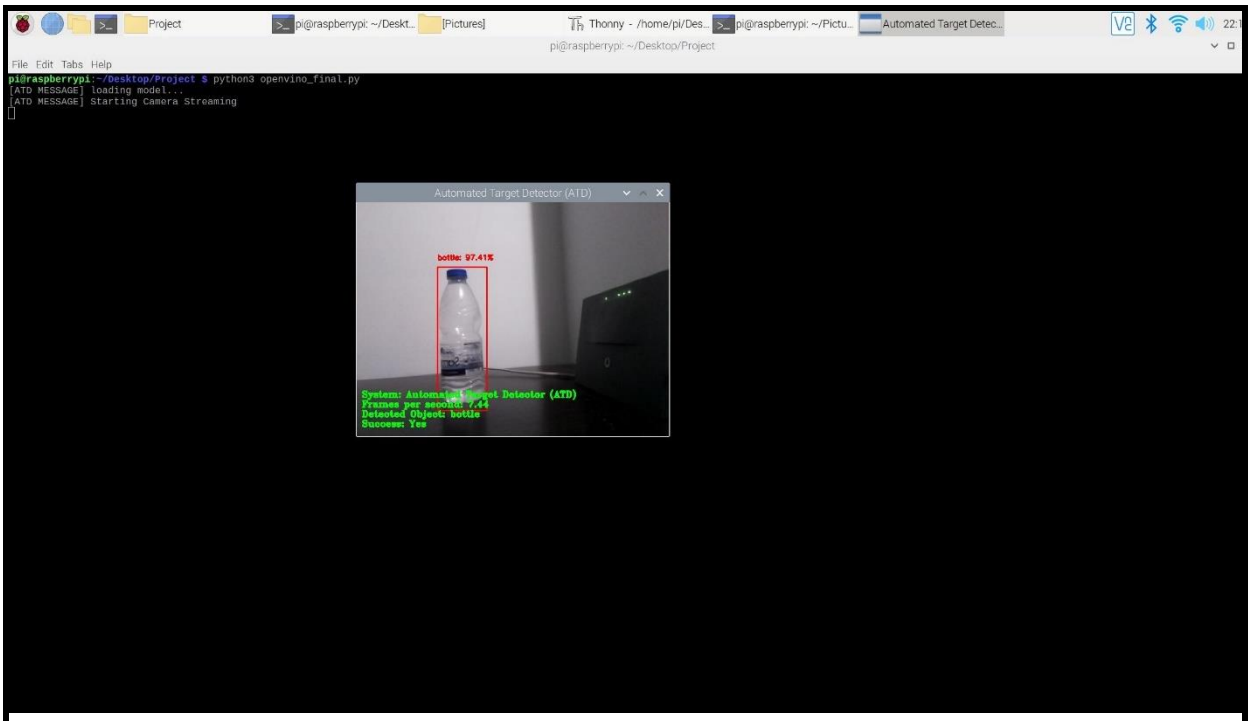
ε.


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 87.30%
System: Automated Target Detector (ATD)
Frames per second: 7.11
Detected Object: bottle
Success: Yes

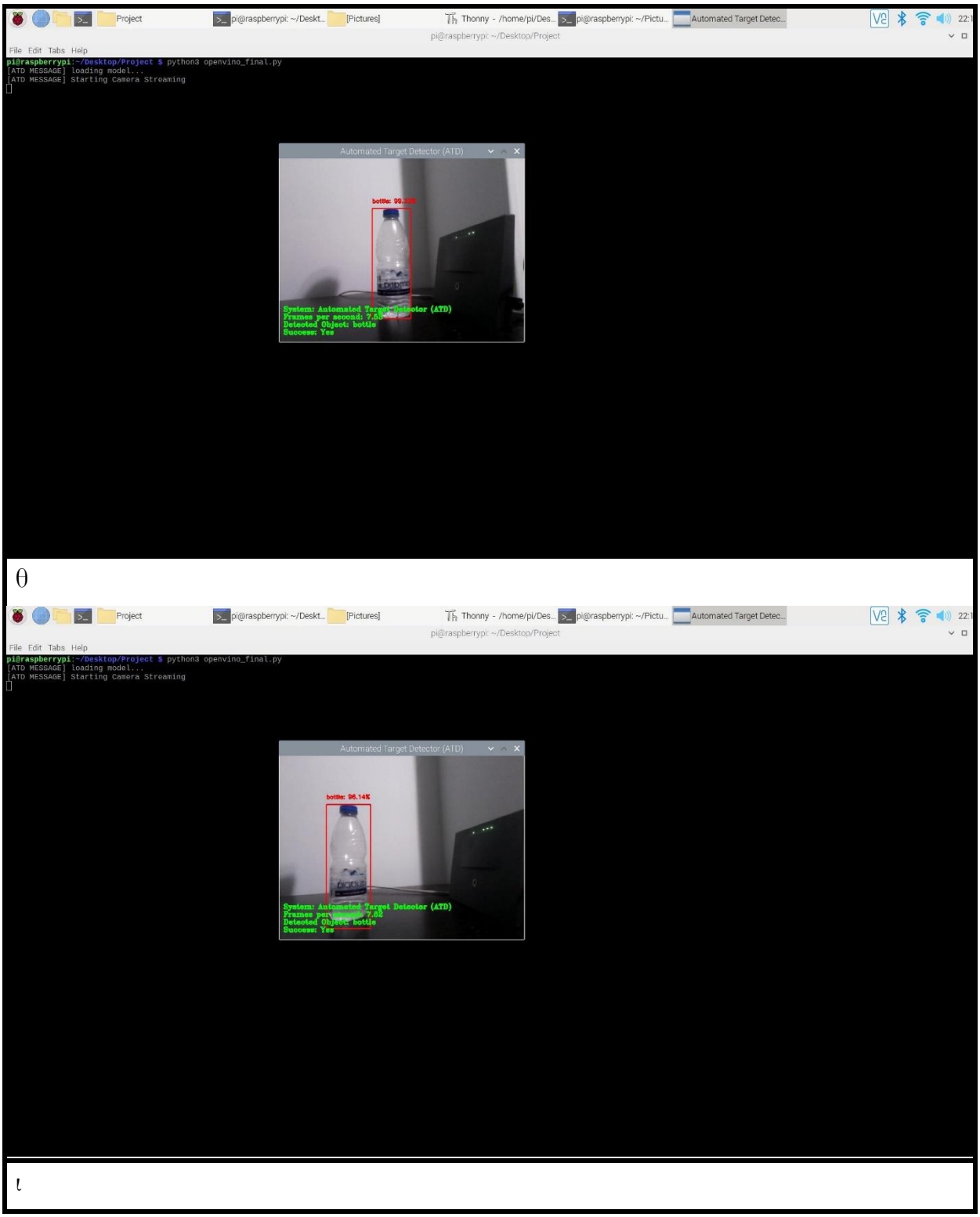
στ.


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 99.71%
System: Automated Target Detector (ATD)
Frames per second: 7.28
Detected Object: bottle
Success: Yes

ζ


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 97.41%
Frames per second: 7.44
Detected Object: bottle
Success: Yes

η


Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
System: Automated Target Detector (ATD)
Detected Object: bottle
Success: Yes
θ
Project
pi@raspberrypi: ~/Deskt...
[Pictures]
Thonny - /home/pi/Des...
pi@raspberrypi: ~/Pictu...
Automated Target Detec...
pi@raspberrypi: ~/Desktop/Project
File Edit Tabs Help
pi@raspberrypi:~/Desktop/Project $ python3 openvino_final.py
[ATD MESSAGE] loading model...
[ATD MESSAGE] Starting Camera Streaming
Automated Target Detector (ATD)
bottle: 96.14%
System: Automated Target Detector (ATD)
Detected Object: bottle
Success: Yes
ι

Το σύστημα αυτό δεν χρειάζεται εξωτερική παρέμβαση του χρήστη δηλαδή πληκτρολόγηση του "s" και δημιουργία πλαισίου γύρω από το αντικείμενο ενδιαφέροντος. Σε αντίθεση, το σύστημα είναι ικανό από μόνο του να αναγνωρίσει ένα αντικείμενο του τύπου που έχει εκπαιδευτεί και να δημιουργήσει αυτόματα ένα κόκκινο bounding box γύρω του (βλ. Πίνακας 5β). Όσο δεν υπάρχει μπουκάλι στο τρέχον frame το σύστημα δεν αναγνωρίζει τίποτα (βλ. Πίνακας 5α). Όσο το αντικείμενο εξακολουθεί να υπάρχει στα επόμενα frames το σύστημα συνεχώς το αναγνωρίζει είτε αυτό είναι στην αριστερή γωνία μακριά από το μαύρο αντικείμενο (βλ. Πίνακας 5ε, Πίνακας 5ζ), είτε είναι στο κέντρο (βλ. Πίνακας 5γ, Πίνακας 5η, Πίνακας 5θ), είτε είναι υψωμένο (βλ. Πίνακας 5β), είτε είναι υπό κλίση (βλ. Πίνακας 5δ, Πίνακας 5στ), είτε είναι μερικώς εκτός frame(bl. Πίνακας 5στ). Κατά την αναγνώριση του αντικειμένου εκτός από τη δημιουργία του κόκκινου πλαισίου εμφανίζεται και το ποσοστό βεβαιότητας με το οποίο το έχουμε αναγνωρίσει, για παράδειγμα στον Πίνακας 5ι το ποσοστό είναι 96,14%. Αξίζει να σημειωθεί ότι την κίνηση αυτή ακολουθεί και η κάμερα μέσω του gimbal. Τις μικρές κινήσεις του στόχου διορθώνει η κάμερα και αυτό είναι ορατό από τον χώρο που υπάρχει πίσω από το μαύρο αντικείμενο δεξιά που επίτηδες τοποθετήθηκε για να αναδεικνύει αυτές τις μετακινήσεις. Για παράδειγμα ο χώρος πίσω από το αντικείμενο αυτό είναι πολύ μικρότερος στο στιγμιότυπο του Πίνακας 5στ σε σχέση με το χώρο πίσω από το αντικείμενο στο στιγμιότυπο Πίνακας 5δ. Τέλος, από το σημείο που άρχισε η αναγνώριση εμφανίζονται στην κάτω αριστερή γωνία του frame πληροφορίες αναφορικά με το FPS, την επιτυχή αναγνώριση ή όχι, τύπο αντικειμένου που αναγνωρίστηκε κ.ά.

ΚΕΦΑΛΑΙΟ 7
Συμπεράσματα και μελλοντικές επεκτάσεις

Σκοπός της διπλωματικής εργασίας ήταν η σχεδίαση μίας αρχιτεκτονικής στηριζόμενη σε Raspberry Pi και movidius coprocessor ικανή να εντοπίζει αντικείμενα για τα οποία έχει εκπαιδευτεί το σύστημα και να ιχνηλατεί την κίνησή τους μέσω μίας κάμερας που κινείται σε δύο άξονες. Για την αρχιτεκτονική αυτή αναπτύχθηκε κατάλληλος κώδικας σε γλώσσα προγραμματισμού python εμπλουτισμένη με το πακέτο OpenCV. Το σύστημα που αναπτύχθηκε εντοπίζει με ικανοποιητική ακρίβεια και ιχνηλατεί αποδοτικά τον προσδιορισμένο στόχο. Στόχος του συγγραφέα της διπλωματικής αυτής εργασίας είναι αποτελέσματα της αναγνώρισης και ιχνηλάτησης να μην χρησιμοποιούνται μόνο στην καθοδήγηση της κάμερας, αλλά να μεταφέρονται με συγκεκριμένο πρωτόκολλο σε ελεγκτή πτήσης (flight controller) Pixhawk Cube ενός μη επανδρωμένου αεροσκάφους με σκοπό την προσθήκη τεχνητής νοημοσύνης και υπολογιστικής όρασης σε αυτό. Το γεγονός αυτό θα προσδώσει δυνατότητα κίνησης της κάμερας και στον τρίτο άξονα.

Βιβλιογραφία - Πηγές

1 Gangopadhyay I., Chatterjee A., Das I. (2019) Face Detection and Expression Recognition Using Haar Cascade Classifier and Fisherface Algorithm. In: Bhattacharyya S., Pal S., Pan I., Das A. (eds) Recent Trends in Signal and Image Processing. Advances in Intelligent Systems and Computing, vol 922. Springer, Singapore. https://doi.org/10.1007/978-981-13-6783-0_1

2 B. Babenko, M. Yang and S. Belongie, "Visual tracking with online Multiple Instance Learning," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 983-990, doi: 10.1109/CVPR.2009.5206737.

3 https://opencv.org/

4 https://www.raspberrypi.org/

5 https://shop.pimoroni.com/products/pan-tilt-hat?variant=22408353287

6 https://grobotronics.com/armor-case-for-raspberry-pi-4-b-with-dual-fan.html

7 https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html

8 https://github.com/PINTO0309/MobileNet-SSD-RealSense/tree/master/caffemodel/MobileNetSSD

9 https://www.pyimagesearch.com/

10 https://en.wikipedia.org/wiki/PID_controller#Manual_tuning

11 https://www.spaceotechnologies.com/machine-learning-platforms/

12 https://caffe.berkeleyvision.org/

13 Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional Architecture for Fast Feature Embedding. ArXiv Preprint ArXiv:1408.5093.

14 Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2010

15 Geron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

16 Jones, M. J., & Viola, P. (2006). U.S. Patent No. 7,099,510. Washington, DC: U.S. Patent and Trademark Office.

17 Challa, S., Morelande, M., Mušicki, D., & Evans, R. (2011). Fundamentals of Object Tracking. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511975837

18 Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press. ISBN: 9780262035613

19 Finn, A., & Scheding, S. (2010). Developments and Challenges for Autonomous Unmanned Vehicles: A Compendium. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

20 Howse, J., & Minichino, J. (2020). Learning OpenCV 4 computer vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning

21 Molloy, D. (2016). Exploring Raspberry Pi: interfacing to the real world with embedded Linux. John Wiley & Sons.

22 M. H. Ionica and D. Gregg, "The Movidius Myriad Architecture's Potential for Scientific Computing," in IEEE Micro, vol. 35, no. 1, pp. 6-14, Jan.-Feb. 2015, doi: 10.1109/MM.2015.4.

23 H. El-Rewini, H. H. Ali and T. Lewis, "Task scheduling in multiprocessing systems," in Computer, vol. 28, no. 12, pp. 27-37, Dec. 1995, doi: 10.1109/2.476197.

24 Gay, W. (2018). Advanced Raspberry Pi: Raspbian Linux and GPIO Integration. Apress

25 Clinton, D.; Negus, C. (2020). Ubuntu Linux Bible (10th ed.). Wiley. Retrieved from https://www.perlego.com/book/1978898/ubuntu-linux-bible-pdf (Original work published 2020)

26 S. Bennett, "Development of the PID controller," in IEEE Control Systems Magazine, vol. 13, no. 6, pp. 58-62, Dec. 1993, doi: 10.1109/37.248006.

27 https://www.avionews.com/item/127876-usaf-un-solo-uomo-per-quattro-un-manned-aerial-vehicle.html

28 https://grobotronics.com/raspberry-pi-4-model-b-8gb.html#group_1461861ed51dde24b7-3

29 https://sisale.2021outletshop.ru/content?c=gpio%20pins%20raspberry%20pi%203%20b&id=13

30 https://sc04.alicdn.com/kf/HTB1me8lFnJYBeNjy1zeq6yhzVXax.jpg

31 https://m.media-amazon.com/images/I/51z+vmCQB4L._SL1000_.jpg

32 https://grobotronics.com/pimoroni-pan-tilt-hat-full-kit.html#group_1357361ed5e1eb0cf1-1

33 https://m.media-amazon.com/images/I/61EV8F7XpJL._AC_SL1000_.jpg

34 https://imgaz.staticbg.com/thumb/large/2014/xiemei-juan/11/SKU180411/IMG_4503.jpg.webp

35 https://m.media-amazon.com/images/I/41QCcp7a16L._AC_.jpg