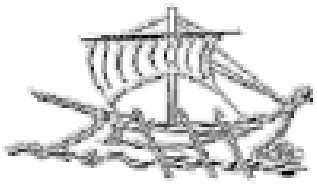


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση διέλευσης ατόμων και ανάλυση της κίνησής τους

Χρήστος Γαμβρουλάς

Εισηγητής: Ιωάννης Ν. Έλληνας, Καθηγητής

ΑΘΗΝΑ – 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση διέλευσης ατόμων και ανάλυση τηε κίνησής τους

Χρήστος Γαμβρουλάς
Α.Μ. 43566

Εισηγητής:

Ιωάννης Ν. Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

-

Ημερομηνία εξέτασης -/-/2017

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η Χρήστος Γαμβρουλάς, του Σπυρίδων, με αριθμό μητρώου 43566 φοιτητής/τρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

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

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

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

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπαιρέωση της παρούσας Πτυχιακής εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου, Ιωάννη Ν. Έλληνα, για τη συνεργασία του και τις πολύτιμες συμβουλές του που ήταν καθοριστικές για την ολοκλήρωσή της.
Επίσης θα ήθελα να ευχαριστήσω τους φίλους μου και πρωτίστως την οικογένειά μου που μου στάθηκαν καθ'όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε με την αναγνώριση διέλευσης ατόμων και την ανάλυση της κίνησης τους.
Αρχικά θα εξετάσουμε τα υλικά που χρησιμοποιήθηκαν για την εκτέλεση της εργασίας, όπως το Raspberry Pi και η Raspberry Pi Camera, αλλά και τα λογισμικά.
Θα αναφερθούμε στην τεχνητή όραση και σε εφαρμογές της και στη συνέχεια θα αναλύσουμε κάποιες μεθόδους αναγνώρισης κίνησης, όπως η μέθοδος διαφοράς διαδοχικών καρές, η μέθοδος εύρεσης φόντου κατά προσέγγιση μέσης τιμής και η μέθοδος ανίχνευσης κίνησης με χρήση στατιστικών μεθόδων.
Αφού αναφέρουμε τις μεθόδους, θα τις χρησιμοποιήσουμε σε αρχείο βίντεο και θα αναφέρουμε τα αποτελέσματα και τα συμπεράσματα.
Στη συνέχεια θα χρησιμοποίσουμε την τελευταία μέθοδο και θα την εφαρμόσουμε σε ζωντανή ροή βίντεο με σκοπό την καταμέτρηση των ατόμων που περνούν μπροστά από την κάμερα.
Τέλος, θα μεταφορτώσουμε τα δεδομένα στην πλατφόρμα ThingSpeak και θα βγάλουμε τα αποτελέσματά μας σε μορφή διαγραμμάτων.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή όραση, Επεξεργασία εικόνας, Αναγνώριση ανθρώπινης κίνησης.
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Raspberry Pi, Python, Μέθοδοι αναγνώρισης κίνησης.

ABSTRACT

In this thesis we will deal with the identification of people passing through and analysis of their movement.

First we will look at the materials used to perform the work, such as Raspberry Pi and Raspberry Pi Camera, as well as the software.

We will refer to artificial vision and its applications, and then we will analyze some motion detection methods, such as the method of successive frames, the background finding method by averaging approach and the method of detecting motion using statistical methods.

Once we report the methods, we will use them in a video file and report the results and conclusions.

Then we will use the latest method and apply it to live video streams to count people who pass in front of the camera.

Finally, we will upload the data to the ThingSpeak platform and make our results in chart formats.

SCIENTIFIC AREA: Artificial vision, Image processing, Human movement detection

KEYWORDS: Raspberry Pi, Python, motion detection methods

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Δημόσιοι χώροι όπως τα εμπορικά κέντρα και τα αεροδρόμια παρακολουθούνται με τη χρήση κλειστού κυκλώματος τηλεόρασης (closed circuit television ή CCTV) ώστε να εξασφαλιστεί η ομαλή λειτουργία. Για την εκτέλεση του έργου αυτού προσλαμβάνεται ένας αριθμός ανθρώπινου δυναμικού, ωστόσο καθώς τα κλειστά κυκλώματα τηλεόρασης γίνονται πιο συχνά είναι απίθανο να παρακαλοθούνται όλες οι πλευρές λόγω του αριθμού των εγκατεστημένων καμερών. Τα τελευταία χρόνια, οι ερευνητές έχουν στραφεί στην τεχνητή όραση προκειμένου να παρακολουθούνται τα πλήθη αυτόματα.

Σε μεγάλους δημόσιους χώρους είναι συχνά αδύνατο να παρακολουθήσει κανείς την ατομική συμπεριφορά κάθε ατόμου λόγω του μεγέθους του πλήθους. Αντιθέτως, οι ιδιότητες ενός πλήθους μπορούν να παρακολοθούνται όπως η κατανομή του πλήθους στον χώρο και ο συνολικός αριθμός των ανθρώπων στο χώρο. Το μέγεθος του πλήθους μπορεί να έιναι δείκτης απειλών για την ασφάλεια όπως κάποια αναταραχή, βίαιη διαμαρτυρία και μαζικό πανικό. Ακόμα και σε ειρινικά πλήθη το μέγεθος μπορεί να είναι δείκτης συμφόρησης, καθυστέρησης ή άλλης ανωμαλίας.

Πληροφορίες σχετικά με το πλήθος μπορούν επίσης να χρησιμοποιηθούν για την παροχή σημαντικών δεδομένων για τα συστήματα επιχειρηματικής ευφυΐας. Για παράδειγμα, η διανομή πλήθους σε ένα συγκρότημα μαγαζιών ή ένα μεγάλο κατάτημα λιανικής πώλησης μπορεί αν χρησιμοποιηθεί για την ανάλυση των καταναλωτικών προτύπων αγορών, ενώ το συνολικό μέγεθος του πλήθους μπορεί να παρακολουθείται για να εκτιμηθεί η απόδοση του καταστήματος στο πέρασμα του χρόνου. Για αυτούς του λόγους η εκτίμηση του μεγέθους του πλήθους είναι ένα θεμελιώδες καθήκον στον τομέα της ανάλυσης της κίνησης του πλήθους.

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία ασχολείται με την αναγνώριση της διέλευσης ατόμων και την ανάλυση της κίνησης τους. Η ανάγκη για την ιδέα αυτή προέκυψε από το πρόβλημα που αναφέρθηκε και τα δεδομένα που προκύπτουν από ένα τέτοιο σύστημα μπορούν να θωρηθούν σημαντικά από διάφορους τομείς.

Για την υλοποίηση της πτυχιακής χρησιμοποιήσαμε το Raspberry Pi ως υπολογιστή για την επεξεργασία εικόνας τις οποίες εικόνες λαμβάνουμε σε μορφή βίντεο με τη βοήθεια της Raspberry Pi Camera. Στη συνέχεια στέλνουμε δεδομένα στην πλατφόρμα ThingSpeak, της εταιρίας Mathworks, στην οποία γίνεται και η ανάλυση κίνησης.

Για αρχή θα χρησιμοποιήσουμε 3 μεθόδους για την αναγνώριση κίνησης και θα αναφέρουμε τα αποτελέσματα. Για την υλοποίηση της πτυχιακής θα χρησιμοποιήσουμε την τελευταία μέθοδο, δηλαδή την αναγνώριση κίνησης με την βοήθεια του αλγορίθμου GMM (Gaussian Mixture Model).

Στη συνέχεια θα χρησιμοποιήσουμε την πλατφόρμα ThingSpeak και με τη χρήση των εργαλείων που προσφέρει θα αναλύσουμε την κίνηση των ατόμων και θα βγάλουμε κάποια αποτελέσματα σε μορφή διαγραμμάτων και τέλος θα συγκρίνουμε 3 συναρτήσεις της βιβλιοθήκης OpenCV (χρησιμοποιείται για την αναγνώριση κίνησης).

1.2 Δομή πτυχιακής εργασίας

Στο *Κεφάλαιο 1* γίνεται μια εισαγωγή στο αντικέιμενο της πτυχιακής εργασίας και εξηγείται η πορεία της.

Στο *Κεφάλαιο 2* γίνεται μια αναφορά στο υλικό που θα χρησιμοποιηθεί.

Στο *Κεφάλαιο 3* γίνεται μια αναφορά στα λογισμικά που θα χρησιμοποιήσουμε για την εκτέλεση της εργασία και εξηγούνται οι ορισμοί τους.

Στο *Κεφάλαιο 4* γίνεται μια αναφορά στην τεχνητή όραση και στι εφαρμογές της.

Στο *Κεφάλαιο 5* γίνεται μια αναφορά στις μεθόδους ανίχνευσης κίνησης που θα χρησιμοποιήσουμε.

Στο *Κεφάλαιο 6* παραπέμπονται τα αποτελέσματα από την εφαρμογή αυτών των μεθόδων.

Στο *Παράρτημα Α* θα δούμε την πορεία της εργασίας μαζί με οδηγίες για την εγκατάσταση των απαραίτητων λογισμικών και προγραμμάτων.

Στο *Παράρτημα Β* θα δούμε τον κώδικα στην τελική του μορφή παραλλαγμένος έτσι ώστε να γίνεται η καταμέτρηση των ατόμων και η μεταφόρτωση των δεδομένων για ανάλυση.

ΚΕΑΛΑΙΟ 2: ΥΛΙΚΟ ΕΡΓΑΣΙΑΣ

2.1 Το Raspberry Pi

2.1.1 Επισκόπηση

Το 2006 μια ομάδα εργαζομένων στο πανεπιστήμιο του Cambridge προσπαθούσε να βρει μια λύση για την κίνηση του ενδιαφέροντος των φοιτητών στην πληροφορική. Σκέφτηκαν πως η λύση θα ήταν ένας μικρός και προσιτός υπολογιστής . Η ομάδα περιλάμβανε μετξύ άλλων τους Rob Mullins, Eben Upton, Alan Mycroft και Jack Lang.

Με την οικονομία σε αυτήν την κατάσταση η προσιτότητα του Raspberry θα βοηθούσε στο να διδάσκεται πρακτικά η πληροφορική στα σχολεία και να κινήσει το ενδιαφέρον των μαθητών.

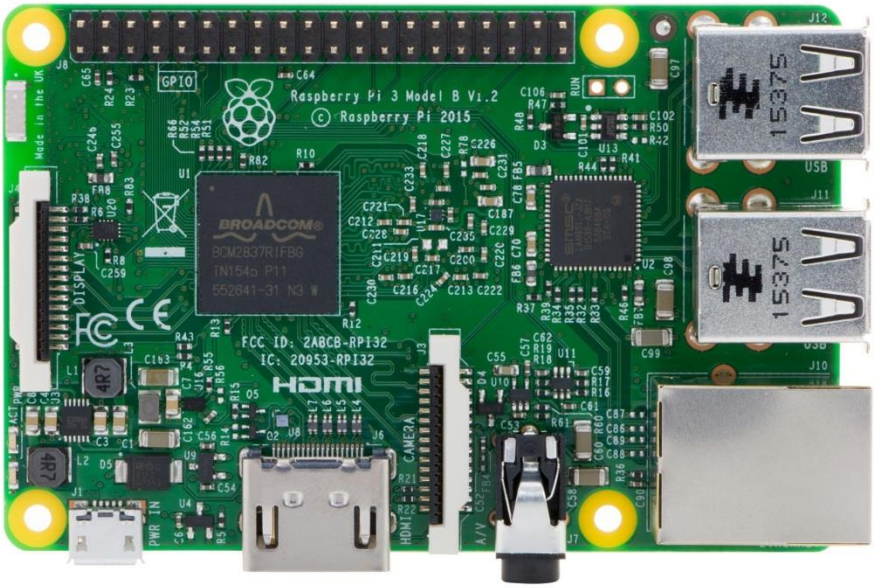
Το 2006 όμως υπήρχαν σημαντικοί περιορισμοί από πλευράς τεχνολογίας. Οι επεξεργαστές για mobile συσκευές είχαν υψηλό κόστος αλλά και χαμηλή ισχύ. Βαθμιαία, με την επέλαση των smartphones το 2007, άρχισε να μειώνεται το κόστος της τεχνολογίας ωστέ να γίνει εφικτή η υλοποίηση του Raspberry.
Το 2008 δημιουργήθηκε το ίδρυμα Raspberry Pi Foundation και τρία χρόνια αργότερα δημιουργήθηκε το πρώτο Raspberry Pi και ακολούθησαν το Model A, Model A+ και Model B.

Μόνο το Model B, πούλησε πάνω απο δύο εκατομμύρια κομμάτια από όταν βγήκε στην μαζική παραγωγή.

Ακολούθησαν οι εκδόσεις Model B rev 2 και Model B+ οι οποίες διέθεταν 512MB RAM. Τον Οκτώβρη του 2014 οι συνολικές πωλήσεις προσέγγιζαν τα τέσσερα εκατομμύρια.

Το Raspberry συνέχισε να αναπτύσσεται και τον Νοέμβρη του 2015 κυκλοφόρησε το Raspberry Pi Zero το οποίο ήταν το μισό σε μέγεθος σε σύγκριση με το Raspberry Pi. Επομενο στη σειρά έρχεται το Raspberry Pi Generation 3 Model B το οποίο πρωτοκυκλοφόρησε το Φεβρουάριο του 2016. Το συγκεκριμένο μοντέλο διαθέτει

ταχύτερο επεξεργαστή από τα προηγούμενα (ARM Cortex-A53 στα 1200MHz), 1GB RAM και κάρτα γραφικών Broadcom VideoCore IV με την υψηλότερη συχνότητα σε σχέση με προηγούμενες γενιές και μοιάζει με αυτό της εικόνας 2.1.



Εικόνα 1: Raspberry Pi 3 Model B

2.1.2 Το υλικό του Raspberry Pi

Το Raspberry Pi έχει συναντήσει σημαντικές αλλαγές στο υλικό του από έκδοση σε έκδοση και συνεχίζει να αναπτύσσεται. Στην εικόνα που ακολουθεί (εικόνα 2.2) παρουσιάζεται ένας πίνακας με τις αλλαγές στο υλικό από έκδοση σε έκδοση:

Raspberry Pi Boards Feature Summary

	Model B	Model A	Compute Module	Model B+	Model A+	2 Model B	Zero	3 Model B
Release	Apr. 2012	Feb. 2013	Apr. 2014	Jul. 2014	Nov. 2014	Feb. 2015	Nov. 2015	Feb. 2016
RAM	512MB	256MB	512MB			1GB	512MB	1GB
System on a Chip	Broadcom BCM2835					Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2837
Processor	ARMv6 (32-bit)					ARMv7 (32-bit)	ARMv6 (32-bit)	ARMv8 (64-bit)
Cores	1					4	1	4
Clock	700Mhz	700Mhz	700Mhz	700Mhz	700Mhz	900MHz	1GHz	1.2GHz
USB 2.0	2	1		4	1	4	1	4
Ethernet	10/100Mbps	None	10/100Mbps		None	10/100Mbps	None	10/100Mbps
Wi-Fi 802.11n	No							Yes
Bluetooth 4.1	No							Yes
On board storage	SD, MMC, SDIO card slot		4 GB eMMC flash	MicroSDHC slot				
GPIO	8		46	17			40	17
HAT ID bus support	No			Yes			No	yes
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 and VC-1 , H.264/MPEG-4 AVC							
1080p video	30fps							60fps

Εικόνα 2: Πίνακας χαρακτηριστικών Raspberry

Παρατηρούμε πως η κατασκευή του βασίζεται στον Broadcom BCM2835 ο οποίος είναι ένα SoC (System on a Chip) , δηλαδή ένα ολοκληρωμένο σύστημα το οποίο περιλαμβάνει διάφορα ηλεκτρονικά στοιχεία τα οποία λειτουργούν μαζί.

Στην ουσία είναι ένας μικρός υπολογιστής συμπιεσμένος τόσο για να χωρέσει σε ένα chip. Θα μπορούσαμε να συγκρίνουμε ένα SoC με ένα μικρό σύστημα που περιλαμβάνει μητρική, επεξεργαστή, κάρτα γραφικών, ακόμα και κάρτα δικτύου. Σήμερα τα συστήματα αυτά χρησιμοποιούνται σε διάφορα είδη συσκευών όπως σε smartphones, σε tablets, φωτογραφικές μηχανές κ.α.

Ο κεντρικός επεξεργαστής του Raspberry αποτελεί μέλος της σειράς επεξεργαστών ARM 11 της εταιρείας ARM. Συγκεκριμένα, πρόκειται για το μοντέλο ARM1176. Οι

συγκεκριμένοι επεξεργαστές προορίζονταν αρχικά για κινητές συσκευές και αυτό βοήθησε στην μείωση της τιμής για την κατασκευή των Raspberry.

Επίσης, το Raspberry, διαθέτει θύρες USB 2.0 των οποίων ο αριθμός διαφέρει ανα έκδοση και μπορούν να χρησιμοποιηθούν για την σύνδεση περιφερειακών όπως ποντίκι και πληκτρολόγιο.

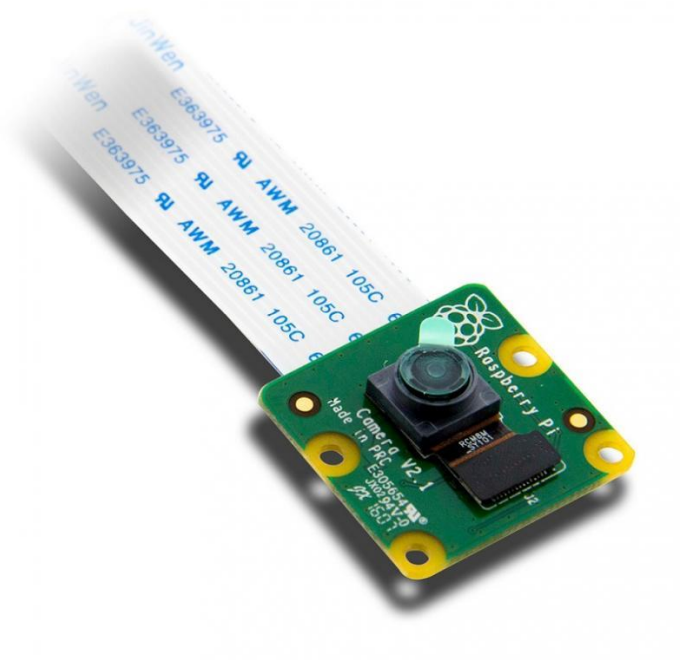
Το Raspberry Pi 3 Model b είναι το πρώτο μοντέλο που έρχεται με ενσωματωμένο WiFi και Bluetooth πράγμα το οποίο το κάνει πιο εύχρηστο και καλύτερο ως συσκευή διασύνδεσης αισθητηρίων με ασύρματη συνδεσιμότητα. Η σύνδεση στο διαδίκτυο στα προηγούμενα μοντέλα επιτυγχάνεται μέσω μιας θύρας Ethernet. Ωστόσο τα μοντέλα model A, model A+ και το Zero δεν διαθέτουν θύρα Ethernet.
Υπάρχει ακόμα αναλογική θύρα ήχου 3.5mm για την κάλυψη των αναγκών σε εφαρμογές πολυμέσων η οποία επιτρέπει τη σύνδεση ακουστικών αλλά και μικροφώνου.

Για τη σύνδεση του συστήματος με οθόνη υπάρχει θύρα HDMI (High Definition Multimedia Interface) η οποία μπορεί να χρησιμοποιηθεί για την αναπαραγωγή εικόνας και ήχου.

Για την αποθήκευση δεδομένων στο Raspberry χρησιμοποιείται κάρτα SD στην ανάλογη θύρα. Η κάρτα αυτή αποτελεί τον σκληρό δίσκο του υπολογιστή και είναι και ο χώρος που θα εγκατασταθεί το λειτουργικό σύστημα. Βέβαια λόφω της περιορισμένης χωριτικότητας μπορεί να χρημοποιηθεί εξωτερικός αποθηκευτικός χώρος σε σύνδεση με θύρα USB.

Από πλευράς μνήμης το model A διαθέτει SDRAM 256MB η οποία δεν ήταν αρκετή για να καλύψει τις ανάγκες των καταναλωτών έτσι το model B βγήκε στην κυκλοφορία με προεγκατεστημένη μνήμη SDRAM 512MB η οποία είναι παραπάνω από αρκετή για την κάλυψη των αναγκών ενός μεγάλου εύρους εφαρμογών. Με την εξέλιξη της τεχνολογιάς και συνεπώς και των εφαρμογών η μνήμη έφτασε στο 1GB στα μοντέλα Raspberry Pi Model B και Raspberry Pi 3 Model B.

Το Raspberry που χρησιμοποιείται στη συγκεκριμένη εφαρμογή διαθέτει και μια διεπαφή (Camera Serial Interface) για τη σύνδεση του Raspberry Pi Camera Module το οποίο χρειαζόμαστε για την άντληση δεδομένων τα οποία στη συνέχεια θα επεξεργαστούμε.



Εικόνα 3: Raspberry Pi Camera Module

Επιπλέον, όλα τα μοντέλα Raspberry φέρουν ενσωματωμένη κάρτα γραφικών Broadcom VideoCore IV η οποία βοηθάει στη διαχείριση εικόνων καθιστόντας την ιδιαίτερα σημαντική για εφαρμογές γραφικών αλλά και παιχνιδιών.
Τέλος, το Raspberry διαθέτει και έναν αριθμό (διαφορετικό ανα έκδοση) από GPIO Pins (General Purpose Inpout/Output Pins) τα οποία χρησιμοποιούνται για τη σύνδεση περιφερειακών στο Raspberry όπως διάφορα αισθητήρια. Προγραμματίζοντάς τα μπορούμε να αντλήσουμε ή να στείλουμε δεδομένα στα περιφερειακά για την κάλυψη των αναγκών της εφαρμογής.

2.1.3 Το λογισμικό του Raspberry

Το Raspberry Pi Foundation για την διευκόλυνση των χρηστών στον τομέα του λογισμικού έχει δημιουργήσει το NOOBS (New Out Of Box System) για την εγκατάσταση του λειτουργικό συστήματος (Το Raspberry έρχεται χωρίς λειτουργικό σύστημα). Ο χρήστης πρέπει να κατεβάσει ένα αρχείο ZIP και να το περάσει στην

κάρτα SD του η οποία πρέπει να είναι 4GB ή μεγαλύτερη. Στη συνέχεια την πρώτη φορά που θα ενεργοποιήσει το Raspberry θα παρατηρήσει ενα μενού από το οποίο θα διαλέξει το λειτουργικό σύστημα που επιθυμέι.

Τα λειτουργικά συστήματα που προτείνει το Raspberry Pi Foundation είναι βασισμένα στον πυρήνα Linux και αυτό γιατί το Raspberry είναι σχεδιασμενο για ελαφριά λειτουργικά συστήματα.

Ανάμεσα στα διάφορα λειτουργικά συστήματα το επίσημο λειτουργικό είναι το Raspbian και γι'αυτο είναι και το πιο δημοφιλές. Πρόκειται για μια εκδοχη του Linux η οποία δημιουργήθηκε για το Raspberry Pi και είναι εφοδιασμένη με όλα τα λογισμικά που μπορέι να χρειαστεί κάποιος για τις καθημερινά του καθήκοντα.

Επίσης, έχει δημιουργηθεί το Rpi Beginners Wiki το οποίο περιέχει πληροφορίες για χρήστες οι οποίοι δεν είναι εξεικοιωμένοι με το Linux.

2.1.4 Εφαρμογές με το Raspberry Pi

Το Raspberry μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός μεγάλου εύρους εφαρμογών αλλά και ως απλός υπολογιστής ικανός να καλύψει τις ανάγκες του μέσου καταναλωτή.

Μπορεί επίσης να χρησιμοποιηθεί σαν παιχνιδοκονσόλα με τη μορφη emulator (εξομοιωτή). Ο χρήστης μπορεί απλά να φορτώσει τα πακέτα emulators που υπάρχουν στο διαδίκτυο και να συνδέσει το Raspberry με οθόνη, πληκτρολόγιο και ποντίκι.

Είναι επίσης ικανό να χρησιμοποιηθεί σαν εγκέφαλος σε συστήματα οικιακού αυτοματισμού, σαν Media Center PC για την αναπαραγωγή ταινιών στην τηλεόραση. Με τη βοήθεια διάφορων αισθητηρίων μπορούμε να αντλήσουμε δεδομένα, να τα αποθηκεύσουμε αλλά και να τα επεξεργαστούμε βγάζοντας τα πορίσματα που χρειαζόμαστε για την επίτευξη ενός στόχου.

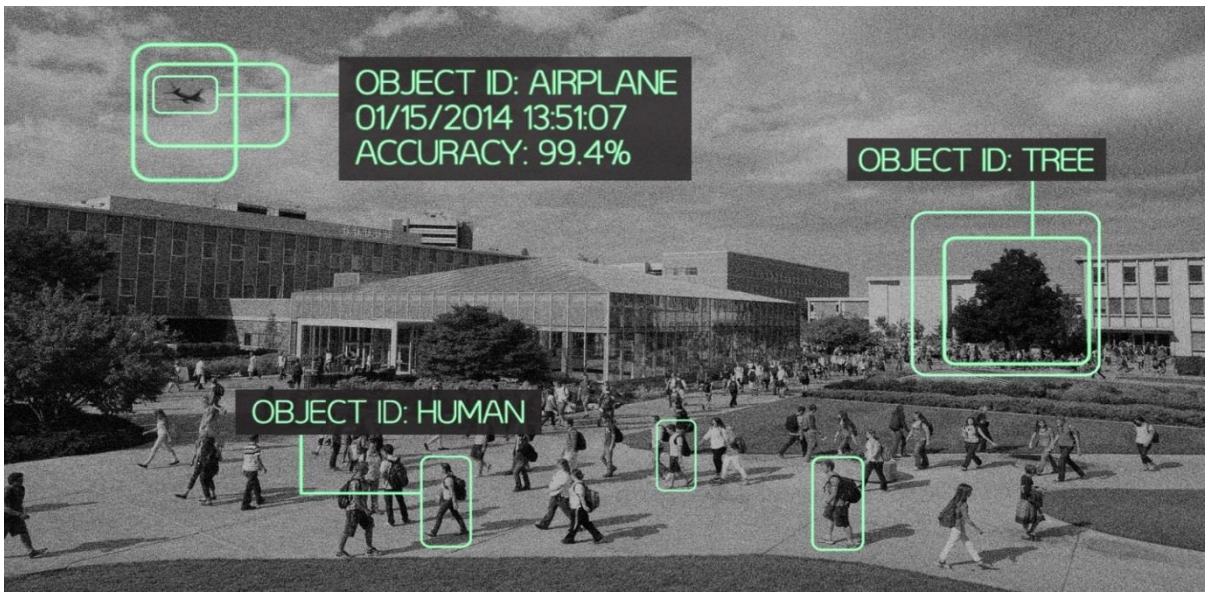
ΚΕΦΑΛΑΙΟ 3: ΛΟΓΙΣΜΙΚΟ ΕΡΓΑΣΙΑΣ

3.1 Η βιβλιοθήκη OpenCV

3.1.1 Επισκόπηση

Η OpenCV (Open Source Computer Vision Library) είναι μια βιβλιοθήκη ελευθέρου λογισμικού γραμμένη σε C/C++ και μπορεί να τρέξει σε όλα τα γνωστά λειτουργικά συστήματα όπως σε Linux, Windows, Macintosh αλλά και σε κινητές συσκευές Android και iOS. Η βιβλιοθήκη δημιουργήθηκε αρχικά από την intel για την επεξεργασία εικόνας σε πραγματικό χρόνο.

Η βιβλιοθήκη OpenCV διαθέτει πάνω απο 2500 βελτιστοποιημένους αλγόριθμους στους τομείς της επεξεργασίας εικόνας αλλά και του machine learning (εκμάθηση μηχανής, αναφέρεται στην απόκτηση νοημοσύνης των μηχανών) οι οποίοι μπορουν να χρησιμοποιηθούν αναγνώριση αντικειμένων αλλά και προσώπων, κινούμενων ή όχι, την αναγνώριση κίνησης κ.α.



Εικόνα 4: Αναγνώριση αντικειμένων με τη χρήση της OpenCV

3.1.2 Ιστορική αναδρομή

Η πρώτη έκδοση της OpenCV έκανε την εμφάνιση της το 2000 στο IEEE Conference on Computer Vision and Pattern Recognition το οποίο θεωρείται το καλύτερο ετήσιο συνέδριο στον τομέα του computer vision (τεχνητή όραση). Η βιβλιοθήκη τότε βρισκόταν σε alpha φάση (alpha θεωρείται η φάση του κύκλου έκδοσης ενός λογισμικού κατά την οποία το υλικό δοκιμάζεται και είναι γενικά ασταθές). Ακολούθησαν 5 εκδόσεις σε φάση beta μεταξύ 2001 και 2005 (beta θεωρείται η φάση στην οποία το λογισμικό περιλαμβάνει, αν όχι όλες, τις περισσότερες λειτουργίες και είναι στη τελική ευθεία).

Η πρώτη ολοκληρωμένη έκδοση κυκλοφόρησε το 2006 και στα μέσα του 2008 η βιβλιοθήκη έλαβε εταιρική υποστήριξη από τον Willow Garage. Η OpenCV βρίσκεται ακόμα και σήμερα κάτω από συνεχή ανάπτυξη.

Στα τέλη 2009 κυκλοφόρησε η δεύτερη έκδοση της βιβλιοθήκης η οποία έφερε σημαντικές αλλαγές στη διεπαφή της γλώσσας C++.

Το 2012 ο μη κερδοσκοπικός οργανισμός OpenCV.org ανέλαβε την υποστήριξη της βιβλιοθήκης. Ο οργανισμός παρέχει ενα διαδικτυακό τόπο για τους χρήστες της OpenCV και άλλον ένα για τους προγραμματιστές.

3.1.3 Πλεονεκτήματα και μειονεκτήματα της Opencv

Η ταχύτητα κατά την εκτέλεση των προγραμματων είναι ξεκάθαρα ένα από τα βασικότερα πλεονεκτήματα της βιβλιοθήκης καθώς και οι μικρές απαιτήσεις σε προγραμματιστικές γνώσεις καθώς ο χρήστης μπορεί να χρησιμοποιήσει τους αλγόριθμους της βιβλιοθήκης με ελάχιστη γνώση πάνω σε αυτούς. Αυτό φυσικά οδηγεί και σε κάποια μειονεκτήματα όπως για παράδειγμα η ανασφάλεια και η αλλοίωση του αποτελέσματος αφού στις περισσότερες περιπτώσεις ο χρήστης δεν ξέρει εις βάθος τον αλγόριθμο που χρησιμοποιεί.

3.2 Η γλώσσα προγραμματισμού Python

3.2.1 Επισκόπηση

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία δημιουργήθηκε το 1990 από τον Γκβίντο βαν Ρόσουμ (Guido van Rossum) και πήρε το όνομα της από την ομάδα άγγλων κωμικών Μόντυ Πάιθον (Monty Python). Κύριος στόχος της γλώσσας είναι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της καθώς το συντακτικό της επιτρέπει στους προγραμματιστές να γράψουν λιγότερες γραμμές κώδικα για τη λύση ενός προβλήματος απ’ότι σε άλλες γλώσσες προγραμματισμού όπως η Java ή η C++ για το ίδιο πρόβλημα.

Η γλώσσα διακρίνεται λόγω των βιβλιοθηκών της οι οποίες διευκολύνουν αρκετές συνιθισμένες εργασίες αλλά και για την ταχύτητα εκμάθησής της.
Επίσης, αξίζει να σημειωθεί πως η γλώσσα Python αναπτύσσεται ως ανοικτό λογισμικό (open source) και διαχειρίζεται από τον μη κερδοσκοπικό οργανισμό Python Software Foundation. Ο κώδικάς της διανέμεται με τη άδεια Python Software Foundation License.

3.2.2 Ιστορική αναδρομή

Η Python, αρχικά, ήταν γλώσσα σεναρίων και χρησιμοποιούνταν στο λειτουργικό σύστημα Amoeba. Στις 16 Οκτωβρίου του 2000 κυκλοφόρησε η Python 2.0 και αργότερα, συγκεκριμένα το Δεκέμβρη του 2008, κυκλοφόρησε η Python 3.0. Πολλά από τα χαρακτηριστικά της νέας έκδοσης μεταφέρθηκαν στις εκδόσεις 2.6 και 2.7 οι οποίες είναι προς τα πίσω συμβατές.

Ιστορικά, η Python 3, είναι η πρώτη γλώσσα προγραμματισμού που σπάει την προς τα πίσω συμβατότητα με προηγούμενες εκδόσεις με σκοπό να διορθωθούν λάθη που υπήρχαν σε προηγούμενες εκδόσεις.

ΚΕΦΑΛΑΙΟ 4: ΤΕΧΝΗΤΗ ΟΡΑΣΗ

4.1 Εισαγωγή

Η μηχανική όραση, υπολογιστική όραση ή αλλιώς τεχνητή όραση, αποτελεί επιστημονικό πεδίο της τεχνητής νοημοσύνης το οποίο επιχειρεί να αναπαράγει αλγοριθμικά την αίσθηση της όρασης, συνήθως σε κάποιο ηλεκτρονικό υπολογιστή. Το πεδίο αυτό σχετίζεται με την σχεδίαση και την κατασκευή συστημάτων τα οποία λαμβάνουν και επεξεργάζονται ψηφιακές εικόνες, παράγοντας δεδομένα. Τα εν λόγω δεδομένα μπορούν να είναι για παράδειγμα φωτογραφίες, βίντεο, όψεις από πολλαπλές κάμερες ή ακόμη και πολυδιάστατες εικόνες από ιατρικό σαρωτή.

Στόχος της τεχνητής όρασης είναι η κατασκευή μηχανικών συστημάτων με δυνατότητα όρασης με την εφαρμογή μοντέλων και θεωριών. Παραδείγματα εφαρμογών τέτοιων συστημάτων είναι τα εξής:

1. Έλεγχος διαδικασιών (π.χ. αυτόνομο όχημα)
2. Ανίχνευση συμβάντων (π.χ. οπτική επιτήρηση)
3. Οργάνωση πληροφοριών (π.χ. ευρετηριοποίηση βάσεων δεδομένωνν και ακολουθιών εικόνων)
4. Εξομοίωση αντικειμένων και περιβάλλοντος (π.χ. ιατρική ανάλυση εικόνας)
5. Αλληλεπίδραση χρηστών με υπολογιστικά συστήματα (π.χ. ως είσοδος σε μια συσκευή επικοινωνίας ανθρώπου –μηχανής)

Όλες αυτές οι διαδικασίες λαμβάνουν κάποια δεδομένα εισόδου τα οποία μπορεί να περιλαμβάνουν κάποιες κωδικοποιημένες πληροφορίες. Μέσα από την απεικόνηση ή την επεξεργασία των δεδομένων, υπάρχει η δυνατότητα της αυτόματης λήψης απόφασης από το σύστημα ή της απόφασης του χρήστη σχετικά με τα αποτελέσματα που έχουν δημιουργηθεί. Η απόφαση μπορεί να είναι κάτι απλό, όπως “υπάρχει ένα άτομο σε αυτή τη σκηνη” ή να αφορά μια πιο σύνθετη διαδικασία, για παράδειγμα “υπάρχουν 14 καρκινικά κύτταρα στην εικόνα».

Με στόχο τη λήψη και την επεξεργασία πληροφοριών, ο τομέας της τεχνητής όρασης χωρίζεται ιεραρχικά σε τρία επίπεδα:

- Όραση χαμηλού επιπέδου: σε αυτό το στάδιο γίνεται η επεξεργασία της εικόνας που έχει ληφθεί και η εξαγωγή των χαρακτηριστικών της, όπως ακμές, γωνίες, χρώματα και άλλα.
- Όραση μεσαίου επιπέδου: μετάτην εξαγωγή των χαρακτηριστικών, λαμβάνονται τα στοιχεία από το προηγούμενο επίπεδο και αναλύεται η κίνηση που υπάρχει στην εικόνα. Εάν είναι απαραίτητο, στο στάδιο αυτό, δημιουργείται τρισδιάστατη κατασκευή της εικόνας αυτής.
- Όραση υψηλού επιπέδου: το τελευταίο αυτό στάδιο, λαμβάνει τις πληροφορίες από το προηγούμενο με σκοπό την ταξινόμηση και την εκκίνηση των διεργασιών που πρέπει να γίνουν σύμφωνα με το αποτέλεσμα που έχει παραχθεί.

4.2 Ιστορική αναδρομή

Τα πρώτα βήματα για την ανάτυξη του τομέα της τεχνητής όρασης έγιναν στη δεκαετία του ’70 απο ερευνητές του τομέα της τεχνητής νοημοσύνης, οι οποίοι είχαν ως στόχο την ανάπτυξη των υπολογιστικών συστημάτων με τη δυνατότητα της όρασης. Ο τομέας της τεχνητής όρασης ανατάσσεται και εξελίσσεται συνεχώς, καθώς συμβαδίζει με τη ραγδαία ανάπτυξη της τεχνολογίας και τις απαιτήσεις τιης εποχής. Τα επιστημονικά πεδία που συμβάλλουν και σήμερα στην ανάπτυξη του τομέα αυτού είναι η τεχνητή νοημοσύνη, η επεξεργασία εικόνας και η αναγνώριση προτύπων.

Η τεχνητή νοημοσύνη αναπτύσσει αλγορίθμους με στόχο την επέκτασή τους ώστε να χρησιμοποιηθούν και από τον τομέα της τεχνητής όρασης.

Η επεξεργασία εικόνας επικεντρώνεται κυρίως στη λήψη πληροφοριών μέσα από το χειρισμό της εκάστοτε εικόνας. Στο πεδίο αυτό συμπεριλαμβάνεται και το πεδίο βελτιστοποίησης της ποιότητας των εικόνων, το οποίο βρίσκεται ακόμη σε

πρωταρχικό στάδιο, καθώς δεν έχει γίνει ακόμη πλήρης αφαίρεση των διάφορων θορύβων που εισάγονται στην εικόνα.

Επιπροσθέτως, η επεξεργασία εικόνας, παρέχει διάφορους μηχανισμούς που αφορούν τη μετατροπή δισδιάστατων εικόνων σε τρισδιάστατες μέσω της βαθμονόμησης της κάμερας, την απομόνωση του θορύβου, την εύρεση χαρακτηριστικών όπως οι ακμές ενός αντικειμένου ακόμη και τη μετατόπιση, την περιστροφή και την ανάκτιση χαμένων πληροφοριών μέσα από μια εικόνα. Οι περισσότεροι αλγόριθμοι χρησιμοποιούν σε ένα μεγάλο ποσοστό την επεξεργασία εικόνας, καθώς μέσα από το στάδιο αυτό παρέχονται χρήσιμα στοιχεία.

Η αναγνώριση πρωτύπων με τη χρήση διαφορετικών τεχνικών, όπως οι στατιστικές προσεγγίσεις και τα νευρωνικά δίκτυα, εξάγει πληροφορίες από καθέ εικόνα μέσω σημάτων και τεχνικών εκμάθησης για την αναγνώριση και την ταξινόμηση προτύπων. Οι τεχνικές αναγνώρισης πρωτύπων χρησιμοποιούνται ευρέως τα τελευταία χρόνια σε εφαρμογές που αφορούν τη μηχανική όραση και έχουν βελτιστοποιηθεί σε μεγάλο βαθμό, ώστε να θεωρούνται αξιόπιστες.

4.3 Εφαρμογές της τεχνητής όρασης

Οι εφαρμογές της τεχνητής όρασης είναι ποικίλες και αφορούν διάφορους τεχνολογικούς τομείς. Οι βασικές λειτουργίες των εφαρμογών αυτών είναι κοινές σε όλους τους τομείς που χρησιμοποιούνται.

Οι πρώτες εφαρμογές που αναπτύχθηκαν αφορούσαν κυρίως στρατιωτικούς σκοπούς. Ένα από τα πιο γνωστά παραδείγματα είναι η καθοδήγση πυραύλων, η ανίχνευση στρατευμάτων καθώς και η δημιουργία οπλικών συστημάτων που ανιχνεύουν πρότυπα και ενεργούν κατά βούληση. Η τεχνητή όραση χρησιμοποιείται στον τομέα αυτο για να αποεφευχθούν τυχόν λάθη από το χειρισμό τέτοιου είδους μηχανημάτων από ανθρώπους. Για την αύξηση της αξιοπιστίας και τη μείωση της πολυπλοκότητας χρησιμοποιείται αυτόματη επεξεργασία δεδομένων τα οποία λαμβάνονται από πολλαπλούς αισθητήρες.

Ένας ακόμη τομέας στον οποίο έχει εφαρμογή η τεχνητή όραση είναι η ιατρική και η επεξεργασία ιατρικών δεδομένων. Οι ιατρικές εφαρμογές που βασίζονται στον τομέα της τεχνητής όρασης έχουν ως κύριο άξονα την επεξεργασία εικόνων. Οι εικόνες αυτές μπορούν να προέρχονται από κάποιο μικροσκόπιο, από υπέρηχους, τομογράφους ή κάποιο άλλο μέσο καταγραφής εικόνων. Κάποια ενδεικτικά παραδείγματα χρήσης του τομέα αυτού στην ιατρική είναι η έυρεση όγκων, η ανίχνευση κάποιας κυτταρικής ή χρωμοσωμικής αντίθεσης και η κατηγοριοποίηση των κυττάρων. Όπως είναι φυσικό τα αποτελέσματα που εξάγονται εξετάζονται και από τους γιατρούς καθώς στην ιατρική όσο αξιόπιστο κι αν είναι ένα μηχάνημα δεν μπορεί να «αποφασίσει», σε ορισμένες περιπτώσεις, μόνο του για την κατάσταση ενός όγκου ή ενός κυττάρου.

Μεγάλη άνθηση, γνωρίζει ο τομέας αυτός και στη ρομποτική. Όταν ένα ρομπότ κινείται σε άγνωστο χώρο ή σε χώρο που μπορεί να αλλάξει η διάταξή του, τότε εμφανίζεται η ανάγκη της όρασης μηχανής. Η τεχνητή όραση μέσα από τις διεργασίες που χρησιμοποιεί δίνει τη δυνατότητα στο ρομπότ να μπορεί να κινηθεί με άνεση σε ένα χώρο, να αποφεύγει τα εμπόδια και να αλλάζει πορεία ανάλογα με τα ερεθίσματα που λαμβάνει από το περιβάλλον. Σε εφαρμογές πραγματικού χρόνου (Real Time Applications) το ρομπότ απαιτεί να έχει στη διάθεσή του πολλές εικόνες το δευτερόλεπτο ειδικά όταν κινείται με μεγάλη ταχύτητα. Συνήθως χρησιμοποιούνται επαναδιαμορφωμένες διατάξεις από ολοκληρωμένα κυκλώματα και αισθητήρες οι οποίοι προγραμματίζονται με τη χρήση αλγορίθμων τεχνητής όρασης.

Οι αισθητήρες που χρησιμοποιούνται για να αντιλαμβάνεται ένα ρομπότ τη διάταξη του χώρου πυ βρίσκεται μπορεί να είναι:

- Αισθητήρες αφής.
- Διατάξεις πομποδέκτων υπερύθρων.
- Διατάξεις πομποδέκτων υπερήχων (σόναρ).
- Διατάξεις με κάμερες.

Αξίζει να σημειωθεί πως η ρομποτική αντιμετωπίζει πολλά και διαφορετικά προβλήματα στον τομέα της όρασης μηχανής. Το μεγαλύτερο από αυτά είναι η

παρεμβολή που δημιουργείται στις εικόνες λόγω της φτεινότητας. Οι διεργασίες που αφορούν την εξαγωγή χαρακτηριστικών προκειμένου να γίνει ανίχνευση κάποιου αντικειμένου ή η αναγνώριση ενός ατόμου, έρχονται αντιμέτωπες με την αυξημένη πιθανότητα σφάλματος κατά την επεξεργασία των εικόνων, λόγω φωτός.

Άλλο ένα πρόβλημα είναι η αναγνώριση αντικειμένων. Για να μπορέσει να χαρακτηριστέι ένα αντικείμενο ως γνωστό ή άγνωστο θα πρέπει να έχει προηγηθεί η εκπαίδευση του συστήματος σχετικά με το ποιά αντικείμενα θα πρέπει να θεωρούνται γνωστά και ποιά άγνωστα. Η εκπαίδευση αυτή γίνεται μέσα από ορισμένα δείγματα φωτογραφιών με τη βοήθεια διάφορων τρόπων και αλγορίθμων ωστέ να γίνει η σωστή αναγνώριση. Το πρόβλημα στο κομμάτι αυτό αφορά τον αριθμό των φωτογραφιών που απαιτούνται για την αναγώριση. Επειδή, ο αριθμός αυτός δεν είναι σταθερός και κάθε εφαρμογή έχει διαφορετικές ανάγκες, απαιτείται να γίνεται ενδελεχής έρευνα του αριθμού των εικόνων καθώς και της ποιότητας αυτών, ωστέ να βρεθεί το δείγμα αυτό το οποίο θα δίνει το βέλτιστο αποτέλεσμα αναγνώρισης.

ΚΕΦΑΛΑΙΟ 5: ΜΕΘΟΔΟΙ ΑΝΙΧΝΕΥΣΗΣ ΚΙΝΗΣΗΣ

5.1 Εισαγωγή

Σχεδόν κάθε σύστημα που έχει ως στόχο την ανάλυση της ανθρώπινης κίνησης ξεκινά με την ανίχνευση ή αλλιώς κατάτμησης της κίνησης σε μια ακολουθία βίντεο. Αυτή η ανίχνευση αποσκοπέι στον διαχωρισμό των περιοχών που περιέχουν κινούμενες οντότητες από την υπόλοιπη εικόνα. Είναι ένα πολύ σημαντικό βήμα για να προχωρήσει κάποιος σε περαιτέρω ανάλυση της ανθρώπινης δραστηριότητας καθώς οι διαδικασίες που ακολουθούν εξαρτώνται σε μεγάλο βαθμό από τα αποτελέσματά του.

Μία από τις δημοφιλέστρες προσεγγίσεις είναι η μοντελοποίηση του φόντου. Η τρέχουσα εικόνα συγκρίνεται με ένα μοντέλο του φόντου το οποίο δεν περιέχει τα κινούμενα αντικείμενα. Συνήθως, αυτό το μοντέλο ανακτάται με την πάροδο του χρόνου. Το πλεονέκτημα τέτοιων προσαρμόσιμων μεθόδων είναι οτι εξάγουν το φόντο από τις εικόνες. Συνήθως, το φόντο ορίζεται ως το πιο συχνό χρώμα στην πάροδο του χρόνου. Αυτό σημαίνει ότι τα αντικείμενα που ήταν στην σκηνή κατά την την εξαγωγή του φόντου βαθμιαία θα αντικατααστηθούν από το φόντο. Αυτό εξαλείφει την ανάγκη της αρχικοποίησης ενός άδειου φόντου. Επίσης, παρέχει ένα μοντέλο φόντου που προσαρμόζεται αυτόματα στις αλλαγές. Για παράδειγμα, όταν αλλάζει ο καιρός ή όταν ένα παρκαρισμένο αυτοκίνητο φεύγει από τη σκηνή.

Ένα σημαντιό μειονέκτημα είναι η αλληλεξάρτηση μεταξύ δύο αντιτιθέμενων απαιτήσεων. Από τη μία, η ενημέρωση του χρόνου θα πρέπει να γίνεται γρήγορα ώστε να να μπορεί να αντιμετωπίσει τις αλλαγές φωτισμού και τις αλλαγές στο φόντο, όπως αλλαγές λόγω ανθρώπων που περπατούν ή λόγω αντικειμένων που φεύγουν από τη σκηνή. Από την άλλη, η ενημέρωση θα πρέπει να γίνεται αργά, ώστε τα αντικείμενα που κινούνται αργά να μην περιλαμβάνονται στο φόντο, όπως ένας άνθρωπος που κοιμάται. Η ταχύτητα λοιπόν, της ενημέρωσης εξαρτάται από την εφαρμογή.

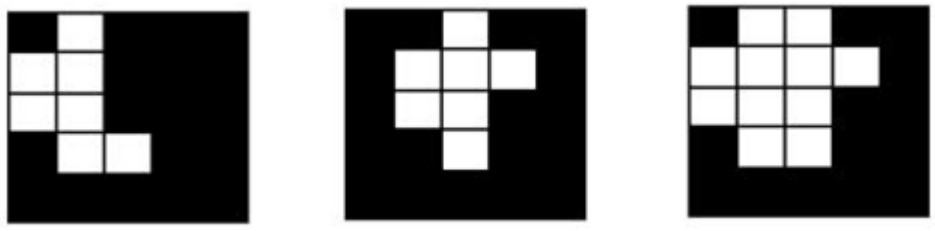
5.2 Μέθοδος διαφοράς διαδοχικών καρέ

Η εύρεση διαφοράς διαδοχικών καρέ, δηλαδή η σύγκριση ανά δύο καρέ και η άντληση της διαφοράς τους αποτελεί τον πιο απλό τρόπο ανίχνευσης κίνησης σε ασπρόμαυρο βίντεο και γίνεται ως εξής:

1. Ορίζουμε έναν πίνακα με διαστάσεις ίδιες με αυτές των καρέ τον οποίο αρχικοποιούμε.
2. Για κάθε δύο διαδοχικά καρέ παίρνουμε κάθε εικονοστοιχείο και βρίσκουμε τη διαφορά της τιμής του μεταξύ των δύο καρέ.
3. Ορίζουμε ένα κατώφλι (threshold) (0<threshold<1)
4. Αν η διαφορά των τιμών του εικονοστοιχείου είναι μικρότερη της τιμής του κατωφλίου, τότε βάζουμε στην αντίστοιχη θέση του πίνακα την τιμή 1, ενώ αν η διαφορά είναι μεγαλύτερη του κατωφλίου, τότε στην αντίστοιχη θέση βάζουμε την τιμή 0.

Έτσι, από κάθε δύο διαδοχικά καρέ προκύπτει η μάσκα, δηλαδή, μία ασπρόμαυρη εικόνα όπου τα μαύρα εικονοστοιχεία αντιστοιχούν στα εικονοστοιχεία που δεν μετακινήθηκαν, δηλαδή στο φόντο, ενώ τα άσπρα εικονοστοιχεία αντιστοιχούν στα εικονοστοιχεία που άλλαξαν, δηλαδή στο κινούμενο αντικείμενο.

Στην παρακάτω εικόνα φαίνεται ένα παράδειγμα, όπως αυτό περιγράφτηκε παραπάνω με κάποια υποτιθέμενα διαδοχικά frames, ώστε να οπτικοποιηθεί η περιγραφή.

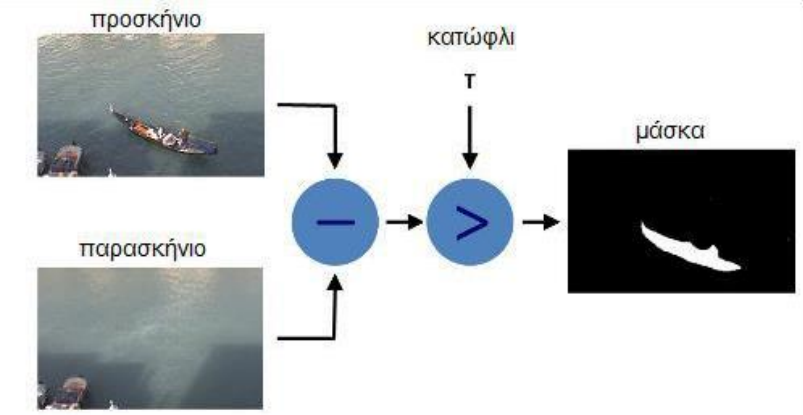


(α) **(β)** **(γ)**

Εικόνα 5: Pixels διαδοχικών καρέ

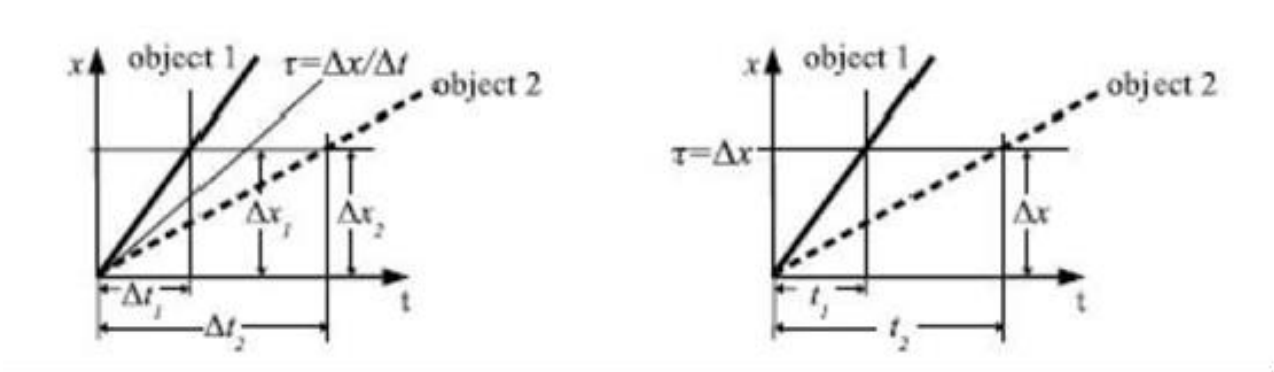
Στα τρία αυτά διαδοχικά καρέ α,β,γ τα μαύρα εικονοστοιχεία αντιστοιχούν στα εικονοστοιχεία που δεν μετακινήθηκαν, δηλαδή στο φόντο, ενώ τα άσπρα εικονοστοιχεία αντιστοιχούν στα εικονοστοιχεία που άλλαξαν, δηλαδή στο κινούμενο αντικείμενο.

Με αυτόν τον τρόπο όμως, προκύπτουν διάφορα προβλήματα. Καταρχήν, χρειάζεται μεγάλη προσοχή στην τιμή που θα βάλουμε στο κατώφλι. Αν η τιμή είναι πολύ μικρή, τότε η παραμικρή κίνηση θα ανιχνευθεί, πράγμα που μπορεί να οδηγήσει σε εσφαλμένες εκτιμήσεις, διότι μπορεί να έχουμε μικρές αλλαγές στο φόντο, οι οποίες δε θέλουμε να θεωρηθούν ως κινούμενα αντικείμενα. Τέτοιες αλλαγές μπορεί να οφείλονται σε θόρυβο λόγω της κάμερας, σε αλλαγές φωτισμού, σε σκιές, σε αέρα αν είμαστε σε εξωτερικό περιβάλλον, κ.ά. Απ' την άλλη, αν η τιμή του κατωφλίου είναι πολύ μεγάλη, τότε κάποιες αλλαγές μπορεί να μην ανιχνευθούν καθόλου, όπως για παράδειγμα ένα αντικείμενο που κινείται με πολύ μικρή ταχύτητα.



Εικόνα 6: Μέθοδος αφαίρεσης φόντου

Στα παρακάτω διαγράμματα βλέπουμε ένα αντικείμενο που κινείται γρήγορα (object 1) και ένα που κινείται αργά (object 2). Στο διάγραμμα που βρίσκεται αριστερά, το κατώφλι τα που επιλέχτηκε, καθορίστηκε με βάση την ταχύτητα των σωμάτων που κινούνται, με αποτέλεσμα το αντικείμενο που κινείται αργά να μη μπορεί ποτέ να ανιχνευθεί ($\tau=\Delta x/\Delta t$). Αντίθετα, στο διάγραμμα που βρίσκεται στα δεξιά, το κατώφλι τ εξαρτάται από ένα προκαθορισμένο καρέ αναφοράς ($\tau=\Delta x$). Αυτό οδηγεί στην ανίχνευση και των δύο αντικειμένων, ανεξάρτητα από την ταχύτητά τους.



Εικόνα 7: Διαγράμματα εν κινήση αντικειμένων

Συνεπώς, φτάνουμε στο συμπέρασμα πως, παρά την απλότητά της, η συγκεκριμένη μέθοδος έχει δύο πολύ μεγάλα μειονεκτήματα. Πρώτον,, ότι για αντικείμενα με ομοιόμορφη κατανομή της τιμής της πυκνότητας τους τα εσωτερικά εικονοστοιχεία τους ερμηνεύονται ως μέρη του φόντου και δεύτερον ότι τα αντικείμενα πρέπει να κινούνται συνεχώς ειδάλλως στην περίπτωση που το αντικείμενο παραμείνει ακίνητο για χρονικό διάστημα ενός καρέ ερμηνεύεται και πάλι σαν μέρος του φόντου.

5.3 Μέθοδος εύρεσης φόντου κατά προσέγγιση μέσης τιμής

Η μέθοδος αυτή αποτελεί μια βελτιωμένη έκδοση της πρώτης μεθόδου που αναφέραμε. Η μεγάλη διαφορά αυτής της μεθόδου από αυτήν που αναλύσαμε παραπάνω είναι ότι για αντικείμενα με ομοιόμορφη κατανομή της τιμής της πυκνότητας τους τα εσωτερικά εικονοστοιχεία τους ερμηνεύονται ως μέρη του φόντου και δεύτερον ότι τα αντικείμενα πρέπει να κινούνται συνεχώς ειδάλλως στην

περίπτωση που το αντικείμενο παραμείνει ακίνητο για χρονικό διάστημα ενός καρέ ερμηνεύεται και πάλι σαν μέρος του φόντου. Φυσικά, η χρήση αυτής της μεθόδου αυξάνει το υπολογιστικό κόστος λόγω του μεγαλύτερου αριθμού αποθηκευμένων καρέ και την επεξεργασία τους.

Ο αλγόριθμος λειτουργεί ως εξής:

Αν ένα εικονοστοιχείο του τωρινού καρέ έχει τιμή μεγαλύτερη από αυτή του αντίστοιχου εικονοστοιχείου του φόντου, τότε το εικονοστοιχείο αυξάνεται κατά ένα. Αντίστοιχα, αν το τωρινό εικονοστοιχείο έχει τιμή μικρότερη από αυτή του αντίστοιχου εικονοστοιχείου του φόντου τότε μειώνεται κατά ένα. Παρατηρούμε λοιπόν ότι η μέθοδος αυτή έχει την ίδια λογική με την προηγούμενη που αναλύσαμε.

Με αυτό τον τρόπο το φόντο συγκλίνει τελικά σε μια τιμή. Όπου τα μισά από τα εικονοστοιχεία της εισόδου είναι μεγαλύτερα από το φόντο και τα μισά είναι μικρότερα, μια κατά προσέγγιση μέση τιμή.

5.4 Μέθοδος ανίχνευσης κίνησης με χρήση στατιστικών μεθόδων

Σχετικά πρόσφατα, κάποιες στατιστικές μέθοδοι για να εξαχθούν οι περιοχές αλλαγών από το φόντο έχουν εμπνευστεί από τις βασικές μεθόδους αφαίρεσης φόντου που ήδη περιγράψαμε. Οι στατιστικές αυτές προσεγγίσεις χρησιμοποιούν χαρακτηριστικά από ατομικά εικονοστοιχεία ή από ομάδες τους για να κατασκευάσουν πιο σύνθετα μοντέλα για το φόντο. Οι στατιστικές αυτών των μοντέλων μάλιστα, αλλάζουν δυναμική κατά τη διάρκεια της επεξεργασίας. Κάθε εικονοστοιχείο στην τρέχουσα εικόνα χαρακτηρίζεται ως εικονοστοιχείο πρώτου πλάνου ή φόντου αφού συγκριθεί με τις στατιστικές του ενίοτε μοντέλου φόντου. Αυτό η προσέγγιση γίνεται όλο και πιο δημοφιλής λόγω της σθεναρότητας που παρουσιάζει στο θόρυβο, στις σκιές, στις αλλαγές των συνθηκών φωτός κλπ.

Το πιο αντιπροσωπευτικό παράδειγμα της κατηγορίας ονομάζεται μίγμα γκαουσσιανών (mixture of Gaussians – MOG). Στη μέθοδο αυτή, το φόντο δεν είναι ένα καρέ με τιμές, όπως είδαμε στη προηγούμενη κατηγορία, αλλά παραμετρικό. Η

κάθε θέση εικονοστοιχείο αντιπροσωπεύεται από έναν αριθμό γκαουσσιανών συναρτήσεων που αθροίζονται όλες μαζί δημιουργώντας μία συσσωρευτική κατανομή πιθανότητας:

$$F(i_t = \mu) = \sum_{i=1}^{k} \omega_{i,t} \cdot \eta(\mu, o)$$

Ο μέσος όρος της κάθε γκαουσσιανής συνάρτησης μπορεί να γίνει αντιληπτός ως μια βάσιμη υπόθεση (educated guess) της τιμής του εικονοστοιχείου στην επόμενη εικόνα, θεωρώντας στο σημείο αυτό ότι τα εικονοστοιχεία ανήκουν γενικά στο φόντο. Το βάρος και η τυπική απόκλιση της κάθε συνάρτησης αποτελούν μέτρα της εμπιστοσύνης μας σε αυτή την υπόθεση όπου μεγαλύτερο βάρος και μικρότερο σ σημαίνουν μεγαλύτερη εμπιστοσύνη. Τυπικά χρησιμοποιούνται από τρείς έως πέντε γκαουσσιανές κατανομές για κάθε εικονοστοιχείο, ανάλογα τους περιορισμούς μας στο μέγεθος της μνήμης του συστήματος.

Για να καθορίσουμε αν ένα εικονοστοιχείο είναι μέρος του σκηνικού το συγκρίνουμε με τις αντίστοιχες γκαουσσιανές συναρτήσεις. Εάν η τιμή του βρίσκεται κοντά στη τυπική απόκλιση ενός στοιχείου του φόντου, σε επίπεδο μιας τάξης μεγέθους, θεωρείται ως μέρος του. Σε αντίθετη περίπτωση, καταχωρείται ως εικονοστοιχείο πρώτου πλάνου.

Η μέθοδος αυτή είναι πολύ ισχυρή και περιέχει πολλές παραμέτρους με την αλλαγή των οποίων μπορούμε να τον προσαρμόσουμε σε οποιοδήποτε περιβάλλον.

Για να δημιουργηθεί η μάσκα με το νέο αντικείμενο που εισήχθη στην εικόνα, γίνεται χρήση μιας εκ των τριών αυτών μεθόδων:

1. Background subtractor MOG.
2. Background subtractor MOG2.
3. Background subtractor GMG.

Η πρώτη μέθοδος βασίζεται στον αλγόριθμο κατάτμησης εικόνας κατά Gauss όπως παρουσιάστηκε στο σύγγραμμα "An improved adaptive background mixture model for real-time tracking with shadow detection" το 2001. Η μέθοδος έχει ως στόχο τη μοντελοποίηση κάθε pixel του φόντου της εικόνας, έχοντας ως γνώμονα μια ανάμιξη Κ Γκαουσιανών (Gaussian K = 3 έως 5) διανομών. Οι τιμές των διανομών αντιπροσωπεύουν τις χρονικές αναλογίες που τα χρώματα παραμένουν στο φόντο. Τα χρώματα που αντιπροσωπεύουν το φόντο είναι εκείνα που παραμένουν αναλλοίωτα ακόμη και με τις μεταβολές του φωτός.

Η μέθοδος Background subtractor MOG2 στηρίζεται στον ίδιο αλγόριθμο κατάτμησης εικόνας με την πρώτη τεχνική, διαφέρει όμως στο κομμάτι των διανομών. Η μέθοδος αυτή επιλέγει τον κατάλληλο αριθμό της Gaussian διανομής για κάθε pixel. Με τον τρόπο αυτό παρέχεται μεγαλύτερη προσαρμοστικότητα του αλγορίθμου στις αλλαγές του φωτός. Ένα ακόμη νέο στοιχείο της μεθόδου αυτής είναι η επιλογή για την ανίχνευση των σκιών. Σε περιπτώσεις όπου η σκιά πρέπει να ανιχνευθεί, τότε το κατώτατο όριο τίθεται χαμηλά εξ’ ορισμού και δεν χρειάζεται περαιτέρω ανάθεση τιμών.

Η τρίτη μέθοδος, Background subtractor GMG, συνδυάζει τη στατιστική εκτίμηση του φόντου μιας εικόνας με την ανά pixel Bayesian κατάτμηση. Εισήχθη το 2012 από τους Andrew B. Godbehere, Akihiro Matsukawa, Ken Goldberg στην εργασία με τίτλο “Visual Tracking of Human Visitors under Variable-Lighting Conditions for a Responsive Audio Art Installation”. Ο αλγόριθμος υλοποίησης της μεθόδου χρησιμοποιεί τις πρώτες εικόνες για τη μοντελοποίηση του φόντου. Για την ανίχνευση πιθανών αντικειμένων στο προσκήνιο βασίζεται σε πιθανολογικό αλγόριθμο σε συνδυασμό με τη μέθοδο Bayesian inference11. Για την ανάπτυξη της μεθόδου αυτής λήφθηκε υπόψη ο θόρυβος που προκαλείται στις εικόνες από τις αλλαγές του φωτισμού και χρησιμοποιήθηκαν αρκετές μορφολογικές λειτουργίες φιλτραρίσματος για την κατάργηση κάθε ανεπιθύμητου θορύβου.

Η μέθοδος αυτή θεωρείται αρκετά καλή, καθώς παραβλέπει το μεγάλο αντίπαλο της στερεοσκοπικής όρασης, τη φωτεινότητα. Χρησιμοποιήθηκε στην παρούσα

εργασία για την εκτέλεση της ανίχνευσης πλήθους στο χώρο, λόγω των υψηλών ποσοστών επιτυχίας της.

Στο επόμενο κεφάλαιο (Κεφάλαιο 6) θα δείξουμε τα αποτελέσματα που πήραμε χρησιμοποιώντας αυτές τις τρεις μεθόδους καθώς και τα συμπεράσματα που βγάλαμε κατά την εκτέλεση.

ΚΕΦΑΛΑΙΟ 6: ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 Εισαγωγή

Για τη σύγκριση των μεθόδων θα χρησιμοποιήσουμε τους αλγόριθμους σε γλώσσα Pyhton και θα τους εφαρμόσουμε στο ίδιο αρχείο βίντεο. Παρακάτω, θα δούμε τα αποτελέσματα των μεθόδων αυτών καθώς και τις διαφορές που παρατηρήσαμε.

6.2 Αποτελέσματα χρήσης της μεθόδου διαφοράς διαδοχικών καρέ

Το αρχείο *motion_detector.py* χρησιμοποιήθηκε για τη δοκιμη της μεθόδου διαφοράς διαδοχικών καρέ.

Κώδικας:

```
# import the necessary packages
import argparse
import datetime
import imutils
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area
size")
args = vars(ap.parse_args())

# video file
camera = cv2.VideoCapture(args["video"])

# initialize the first frame in the video stream
firstFrame = None

# loop over the frames of the video
```

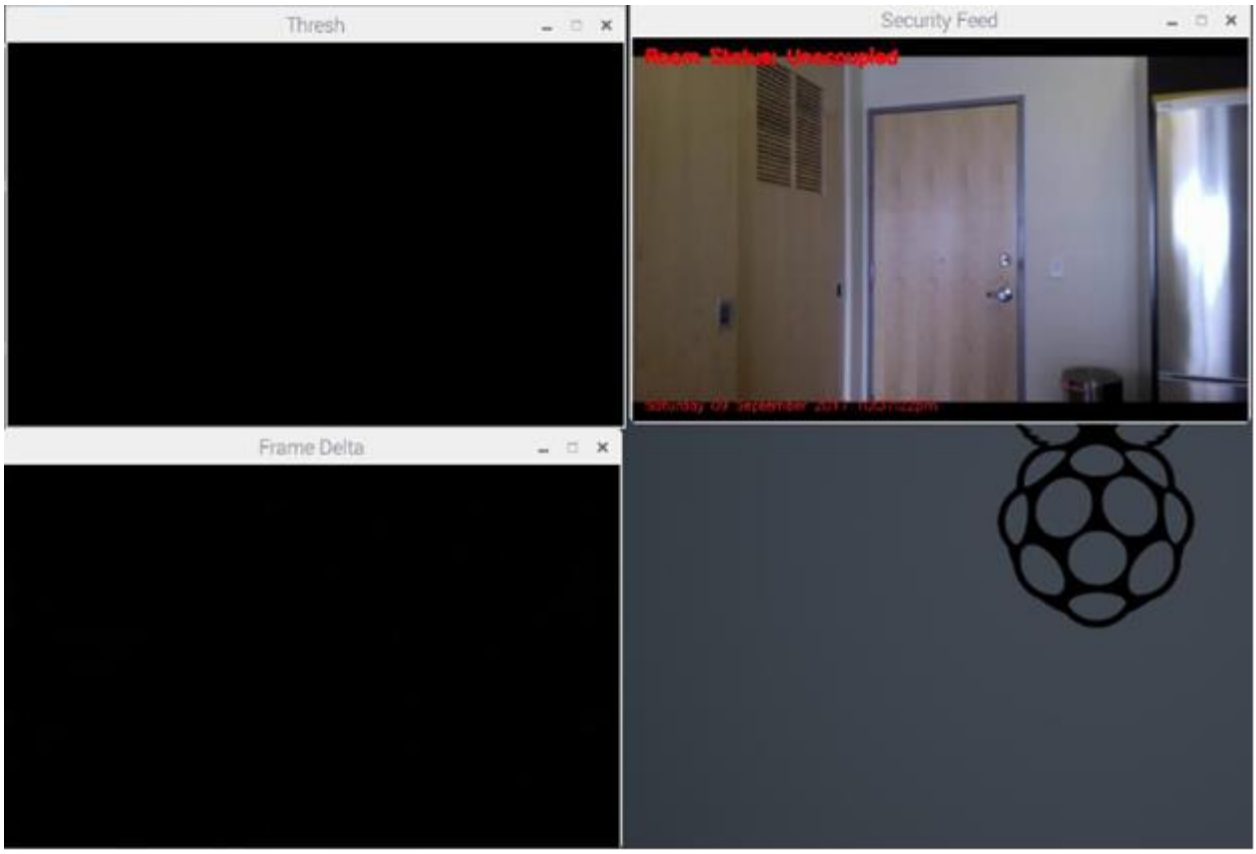
```
while True:
    # grab the current frame and initialize the occupied/unoccupied
    # text
    (grabbed, frame) = camera.read()
    text = "Unoccupied"

    # if the frame could not be grabbed, then we have reached the end
    # of the video
    if not grabbed:
        break

    # resize the frame, convert it to grayscale, and blur it
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    # if the first frame is None, initialize it
    if firstFrame is None:
        firstFrame = gray
        continue

    # compute the absolute difference between the current frame and
    # first frame
    frameDelta = cv2.absdiff(firstFrame, gray)
    thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

    # dilate the thresholded image to fill in holes, then find contours
    # on thresholded image
    thresh = cv2.dilate(thresh, None, iterations=2)
    (_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)

    # loop over the contours
    for c in cnts:
        # if the contour is too small, ignore it
        if cv2.contourArea(c) < args["min_area"]:
            continue

        # compute the bounding box for the contour, draw it on the frame,
        # and update the text
        (x, y, w, h) = cv2.boundingRect(c)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        text = "Occupied"

    # draw the text and timestamp on the frame
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
```

```
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

    # show the frame and record if the user presses a key
    cv2.imshow("Security Feed", frame)
    cv2.imshow("Thresh", thresh)
    cv2.imshow("Frame Delta", frameDelta)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key is pressed, break from the lop
    if key == ord("q"):
        break

# cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
```

Τρέχουμε τον κώδικα με την εντολή : python motion_detector.py –video videos/example_01.mp4

Όπου example_01.mp4 είναι το βίντεο παράδειγμα που θα χρησιμοποιήσουμε.

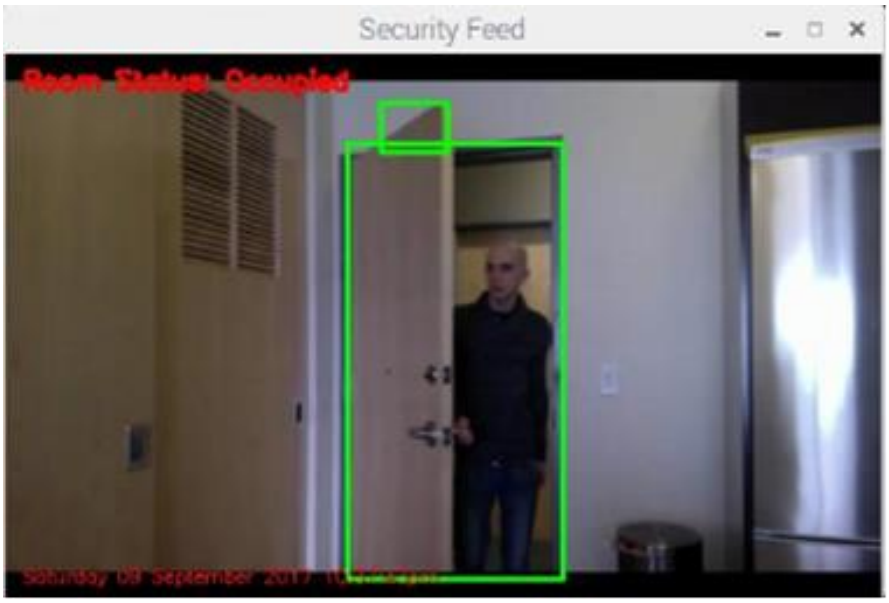
Αρχικά στην οθόνη εμφανίζεται:



Εικόνα 8: Πρώτη φάση εκτέλεσης του motion_detector.py

Παρατηρούμε πως υπάρχουν 3 παράθυρα, στο παράθυρο “Security Feed” βλέπουμε το πρωτότυπο βίντεο και το κείμενο “Room Status:” το οποίο στην τωρινή φάση είναι “Unoccupied” και κάτω αριστερά αναγράφεται η ώρα και η ημερομηνία που επεξεργαζόμαστε το βίντεο. Επίσης υπάρχει το παράθυρο “Thresh” στο οποίο θα δούμε την μάσκα και το παράθυρο “Frame Delta”.

Με την πρώτη εμφάνιση κίνησης παρατηρούμε τις αλλαγές στα παράθυρα:



Εικόνα 9: Εμφάνιση κίνησης στο παράθυρο "Security Feed"

"

Βλέπουμε πως στο πεδίο που εμφανίστηκε κίνηση υπάρχει ένα πράσινο περίγραμμα αυτή τη φορά.

Αλλαγές θα παρατηρήσουμε και στο παράθυρο “Frame Delta”:

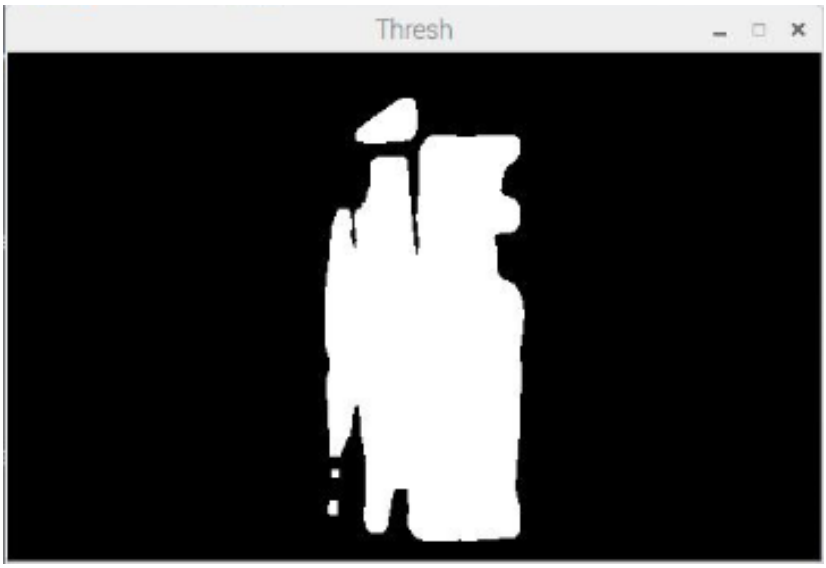
Εικόνα 10: Εμφάνιση κίνησης στο παράθυρο "Frame Delta".

Η αλλαγή που βλέπουμε είναι αποτέλεσμα των συναρτήσεων:

```
cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
cv2.GaussianBlur(gray, (21, 21), 0)
```

Στην ουσία παίρνουμε καρέ (frames) από το πρωτότυπο βίντεο τα μετατρέπουμε σε κλίμακα του γκρι (grayscale) και στη συνέχεια τα θολώνουμε.

Τέλος, δημιουργείται η μάσκα στο παράθυρο “Thresh” όπως βλέπουμε παρακάτω:



Εικόνα 11: Δημιουργία μάσκας.

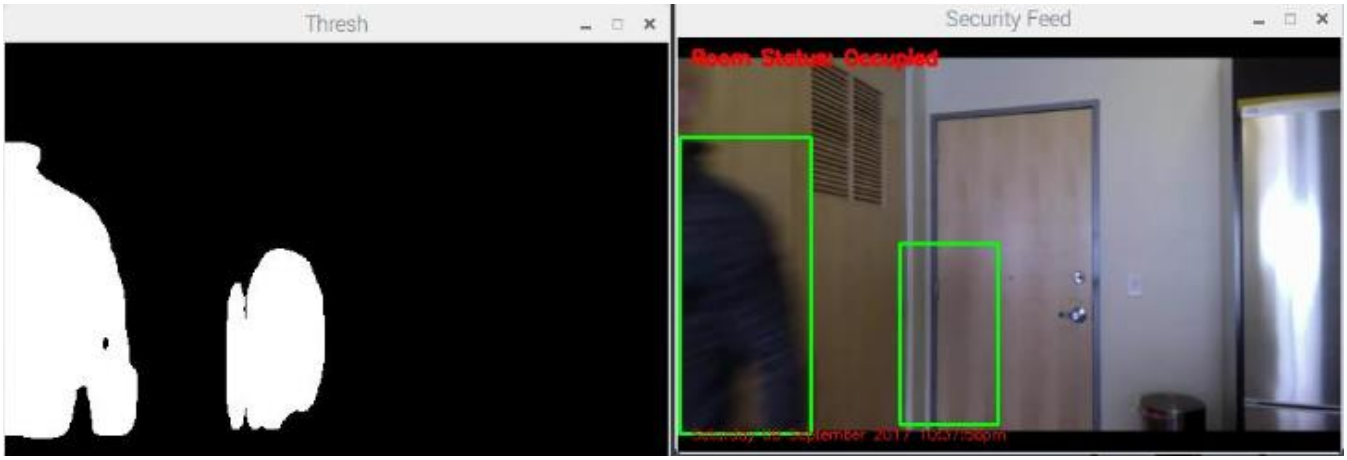
Το αποτέλεσμα αυτό φέρουν οι συναρτήσεις:

```
frameDelta = cv2.absdiff(firstFrame, gray)
thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
```

Υπολογίζεται η απόλυτη διαφορά μεταξύ του πρώτου καρέ και του τρέχον καρέ και ζωγραφίζεται η διαφορά όπως απεικονίζεται στην εικόνα. Η διαφορά υπολογίζεται με τη βοήθεια της συνάρτησης absdiff. Η εικόνα είναι σε άσπρο/μαύρο και αυτό λόγω του τέταρτου ορίσματος της συνάρτησης threshold (cv2.THRESH_BINARY).

Το πρόβλημα με αυτή τη μέθοδο είναι η ταχύτητα παρακολούθησης των διαφορών μεταξύ καρέ και σε συνέχεια η ακρίβεια των αποτελεσμάτων.

Ακολουθεί εικόνα στην οποία φαίνεται το πρόβλημα της ακρίβειας χρησιμοποιώντας τη συγκεκριμένη μέθοδο.



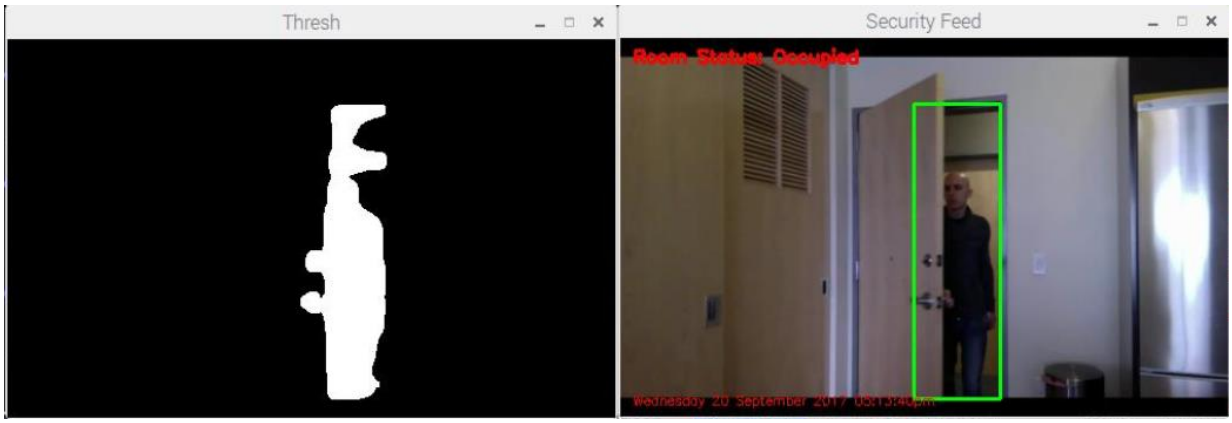
Εικόνα 12: Πρόβλημα ακρίβειας με τη χρήση της πρώτης μεθόδου.

Όπως βλέπουμε λόγω της ταχήτητας αλλαγής των καρέ παρατηρείται κίνηση ενω δεν υπάρχει.

Παρατηρούμε ότι έχουμε δηλώσει ως κατώφλι την τιμή 25. Το κατώφλι δηλώνεται σαν δεύτερο όρισμα στη συνάρτηση threshold της βιβλιοθήκης OpenCV.

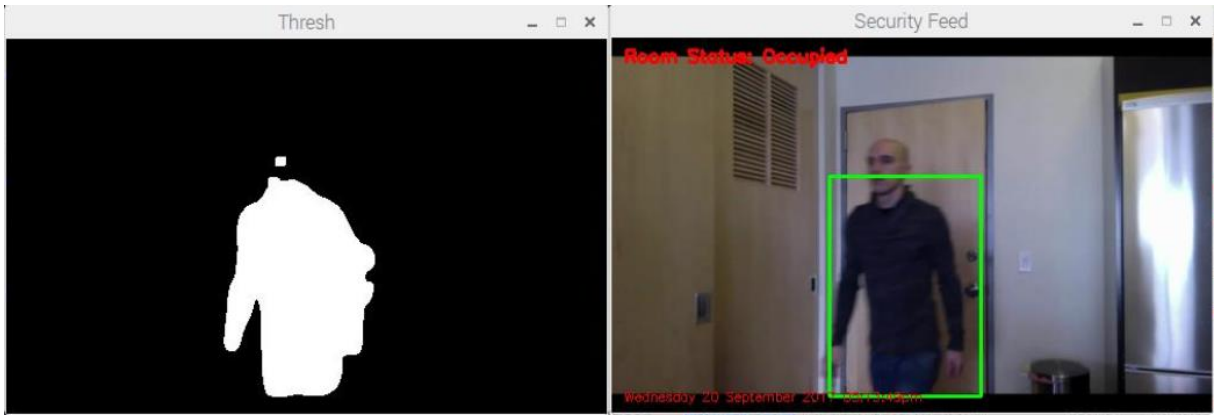
Παρακάτω θα δούμε τα αποτελέσματα της ίδιας μεθόδου αλλάζοντας την τιμή του κατωφλίου.

Για τιμή κατωφλίου 50 πήραμε τα αποτελέσματα:



Εικόνα 13: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 50.

Εικόνα 14: Frame Delta με τιμή κατωφλίου 50.

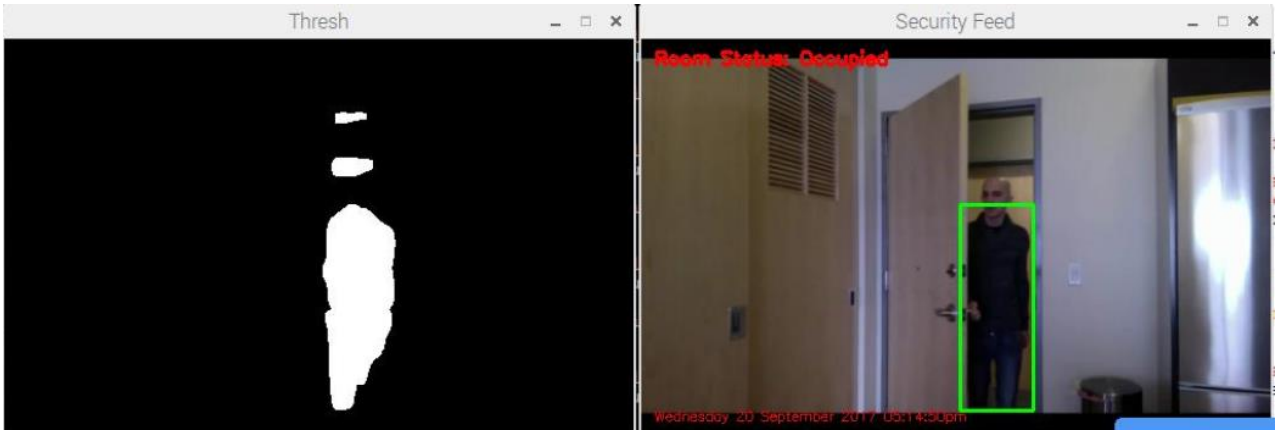


Εικόνα 15: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 50.(2)

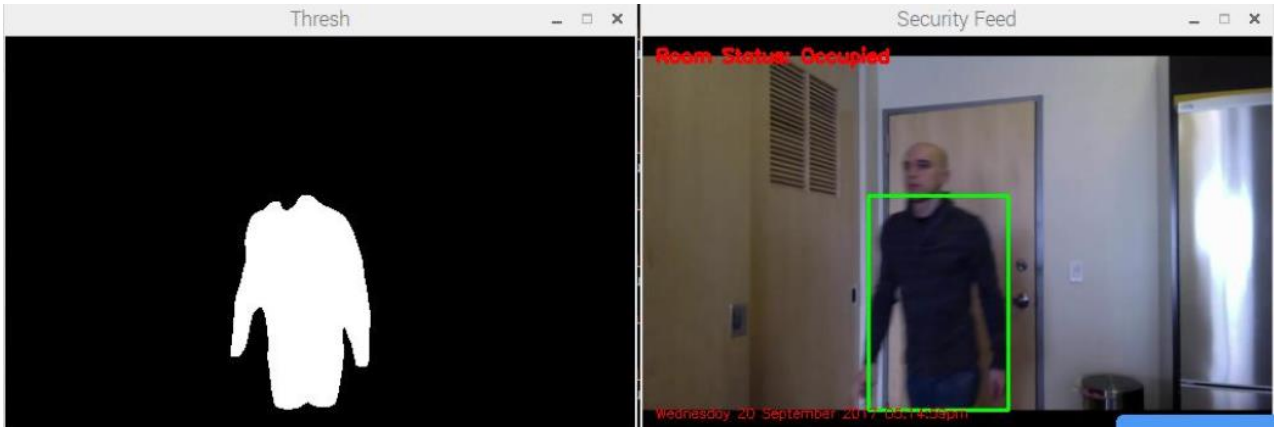
Παρατηρούμε πως με την αλλαγή της τιμής κατωφλίου ο αλγόριθμος τείνει να γίνει πιο «σωστός» από την άποψη ότι δεν είναι τόσο ευαίσθητος στην κίνηση όπως όταν η τιμή κατωφλίου ήταν 25. Βέβαια αυτό έχει σχέση και με την εφαρμογή στην οποία χρησιμοποιείται. Μπορεί σε κάποια εφαρμογή να ήταν χρήσιμη αυτή η ευαισθησία στην κίνηση ενώ σε κάποια άλλη όχι.

Στο παράθυρο “Frame Delta” δεν παρατηρήθηκαν σημαντικές αλλαγές.

Συνεχίζοντας αλλάξαμε την τιμή κατωφλίου σε 75 και πήραμε τα εξής αποτελέσματα:



Εικόνα 16: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 75.

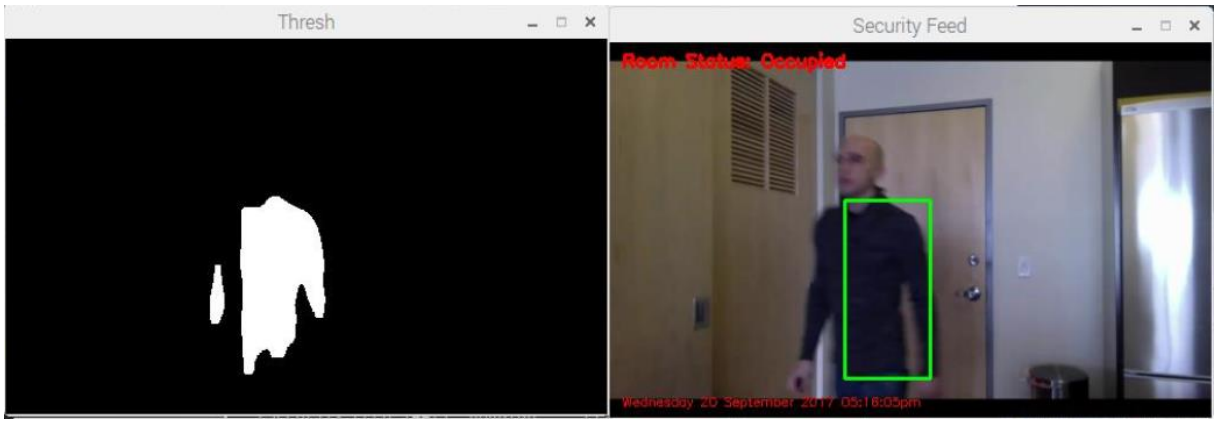


Εικόνα 17: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 75.(2)

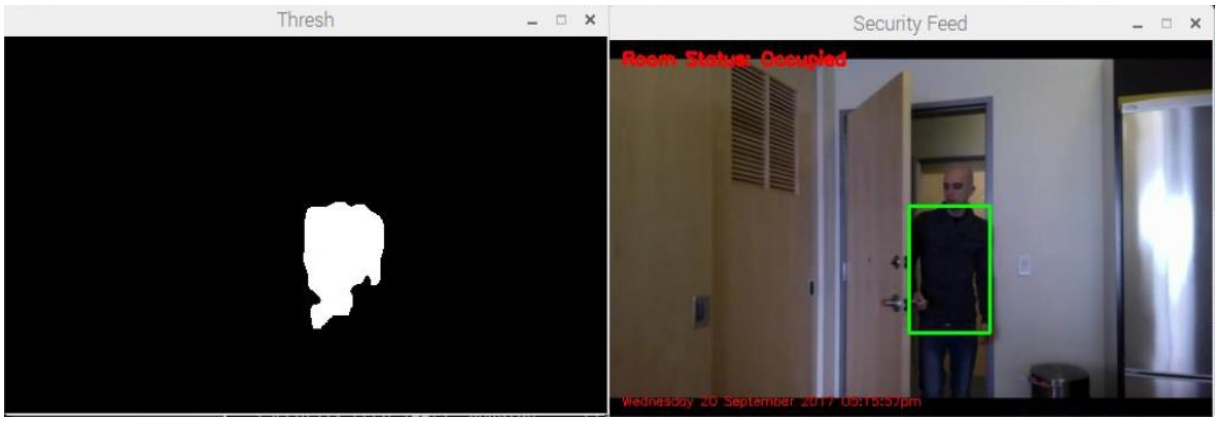
.

Βλέπουμε πως συνχίζει και πέφτει η ευαισθησία στην κίνηση όσο η τιμή κατωφλίου ανεβαίνει.

Τέλος, αλλάξαμε την τιμή κατωφλίου σε 100:



Εικόνα 18: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 100.



Εικόνα 19: Αποτέλεσμα αλλαγής τιμής κατωφλίου σε 100.(2)

Οι αλλαγές στην ευαισθησία του αλγορίθμου στην κίνηση φαίνονται ξεκάθαρα στην μάσκα.

6.3 Αποτελέσματα μεθόδου εύρεσης φόντου κατά προσέγγιση μέσης τιμής.

Όπως αναφέραμε στο Κεφάλαιο 5, η μέθοδος αυτή στηρίζεται στην ίδια λογική με την προηγούμενη με τη διαφορά του φόντου. Βεβαια το υπολογιστικό κόστος αυτής της μεθόδου μπορεί να έχει αρνητικά αποτελέσματα.

Θα δούμε πρώτα τα αποτελέσματα που πήραμε με την τιμή κατωφλίου να είναι 25 και μετά θα αλλάξουμε την τιμή για να δούμε τις αλλαγές.

Το αρχείο *motion_detector_2.py* χρησιμοποιήθηκε για τη δοκιμη της μεθόδου εύρεσης φόντου κατά προσέγγιση μέσης τιμής.

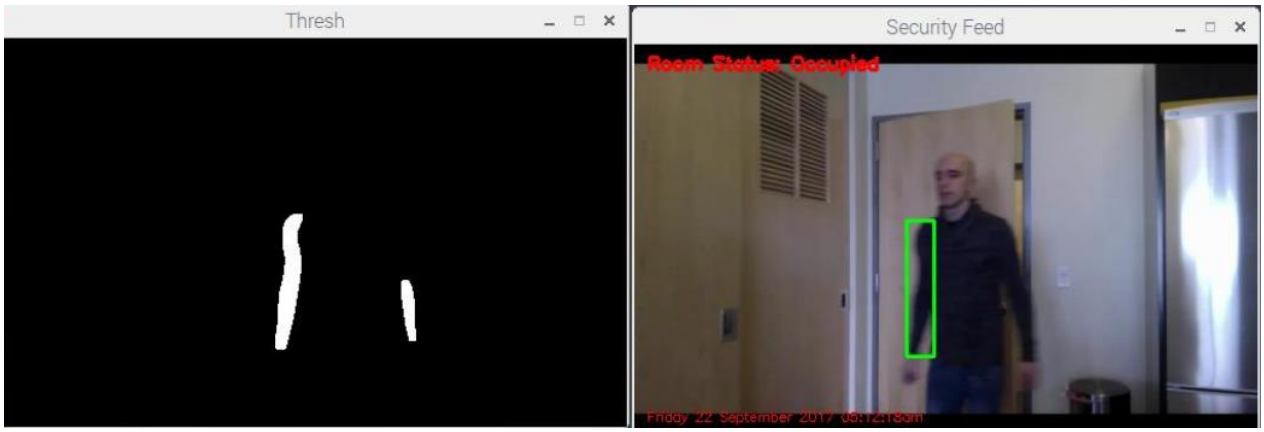
Κώδικας:

```
# import the necessary packages
import argparse
import datetime
import imutils
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area
size")
args = vars(ap.parse_args())

# video file
camera = cv2.VideoCapture(args["video"])

# initialize the average frame
avg = None

# loop over the frames of the video
while True:
    # grab the current frame and initialize the occupied/unoccupied text
    (grabbed, frame) = camera.read()
    text = "Unoccupied"

    # if the frame could not be grabbed, then we have reached the end
    # of the video
    if not grabbed:
        break

    # resize the frame, convert it to grayscale, and blur it
    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)

    # if the average frame is None, initialize it
    if avg is None:
        print ("[INFO] starting background model...")
```

```
        avg = gray.copy().astype("float")
        continue

    # accumulate the weighted average between the current frame and
    # previous frames, then compute the difference between the current
    # frame and running average
    cv2.accumulateWeighted(gray, avg, 0.5)
    frameDelta = cv2.absdiff(gray, cv2.convertScaleAbs(avg))



# threshold the delta image, dilate the thresholded image to fill
    # in holes, then find contours on thresholded image
    thresh = cv2.threshold(frameDelta, 25, 255,
        cv2.THRESH_BINARY)[1]
    thresh = cv2.dilate(thresh, None, iterations=2)
    (_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)

    # loop over the contours
    for c in cnts:
        # if the contour is too small, ignore it
        if cv2.contourArea(c) < args["min_area"]:
            continue

        # compute the bounding box for the contour, draw it on the frame,
        # and update the text
        (x, y, w, h) = cv2.boundingRect(c)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        text = "Occupied"

    # draw the text and timestamp on the frame
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

    # show the frame and record if the user presses a key
    cv2.imshow("Security Feed", frame)
    cv2.imshow("Thresh", thresh)
    cv2.imshow("Frame Delta", frameDelta)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key is pressed, break from the lop
```

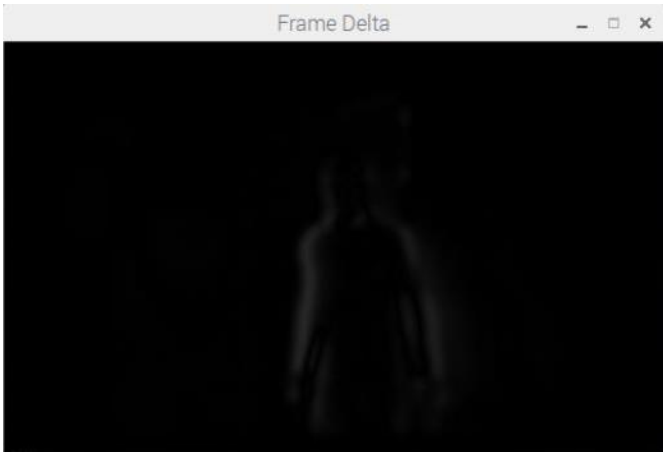
```
if key == ord("q"):
    break
```

Τρέχουμε τον κώδικα με την εντολή : python motion_detector_2.py –video videos/example_01.mp4 και παίρνουμε τα εξής αποτελέσματα:



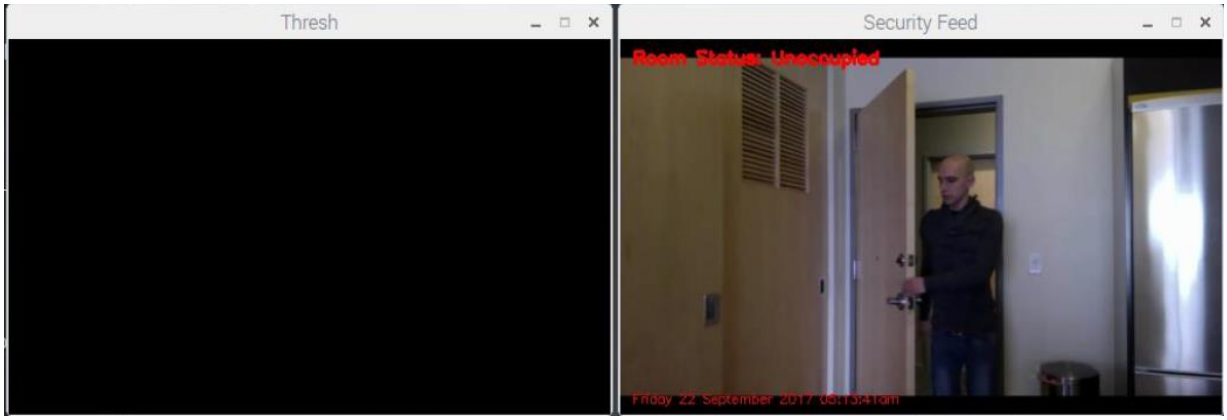
Εικόνα 20: Αποτέλεσμα δεύτερης μεθόδου με τιμη κατωφλίου 25.

Παρατηρούμε μικρή ευαισθησία ως προς την κίνηση όπως φαίνεται και στο παράθυρο “Frame Delta” :

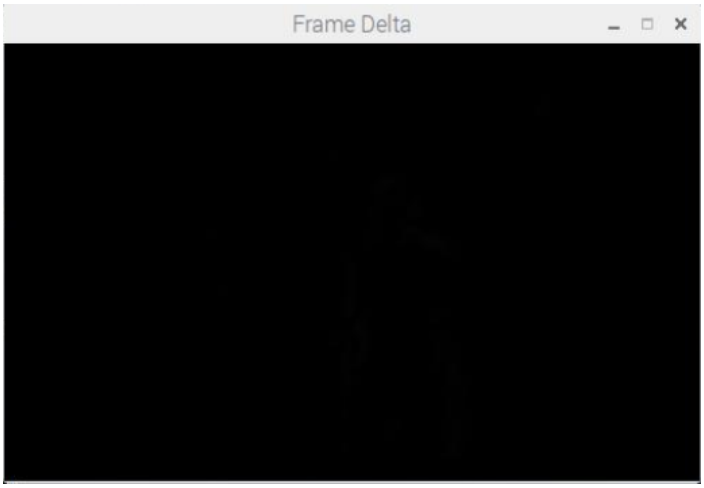


Εικόνα 21: Παράθυρο "Frame Delta" με τη χρήση της δεύτερης μεθόδου με τιμή κατωφλίου 25.

Αλλάζοντας την τιμή κατωφλίου σε 50 πήραμε τα παρακάτω αποτελέσματα:



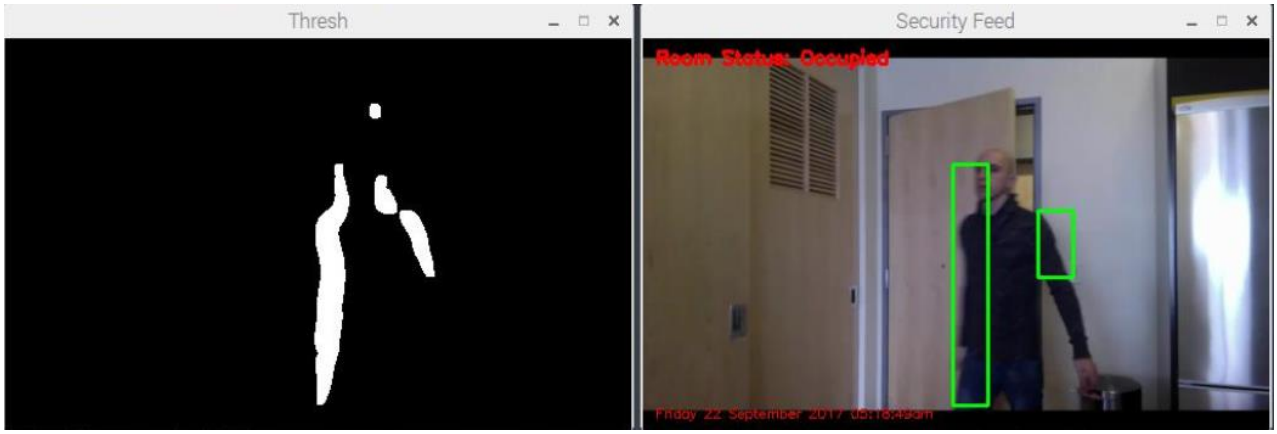
Εικόνα 22: Αποτέλεσμα δεύτερης μεθόδου με τιμη κατωφλίου 50.



Εικόνα 23: Παράθυρο "Frame Delta" με τη χρήση της δεύτερης μεθόδου με τιμή κατωφλίου 50.

Βλέπουμε πως αυξάνοντας την τιμή κατωφλίου ο αλγόριθμος γίνεται λιγότερο ευαίσθητος ως προς την κίνηση.

Μειώνοντας την τιμη κατωφλίου σε 10 πήραμε τα εξής αποτελέσματα:



Εικόνα 24: Παράθυρο "Frame Delta" με τη χρήση της δεύτερης μεθόδου με τιμή κατωφλίου 50.

Είναι εμφανές πως έχει αυξηθεί η ευαισθησία στην κίνηση αλλά η ιδέα της εύρεσης φόντου κατά προσέγγιση μέσης τιμής έχει μειονεκτήματα και φαίνεται πως το υπολογιστικό κόστος είναι μεγάλο, τουλάχιστον για την επεξεργάστικη μονάδα που χρησιμοποιήθηκε στην εργασία, το Raspberry Pi.

6.4 Αποτελέσματα μεθόδου ανίχνευσης κίνησης με χρήση στατιστικών μεθόδων

Για τη μέθοδο αυτή θα χρησιμοποιηθούν 3 τεχνικές για την εύρεση της μάσκας. Θα ξεκινήσουμε με τη συνάρτηση createBackgroundSubtractorMOG2 της βιβλιοθήκης OpenCV και στη συνέχεια θα χρησιμοποιήσουμε τις συναρτήσεις createBackgroundSubtractorMOG και createBackgroundSubtractorGMG.

Το αρχείο *motion_detector_3.py* χρησιμοποιήθηκε για τη δοκιμη της μεθόδου εύρεσης φόντου κατά προσέγγιση μέσης τιμής

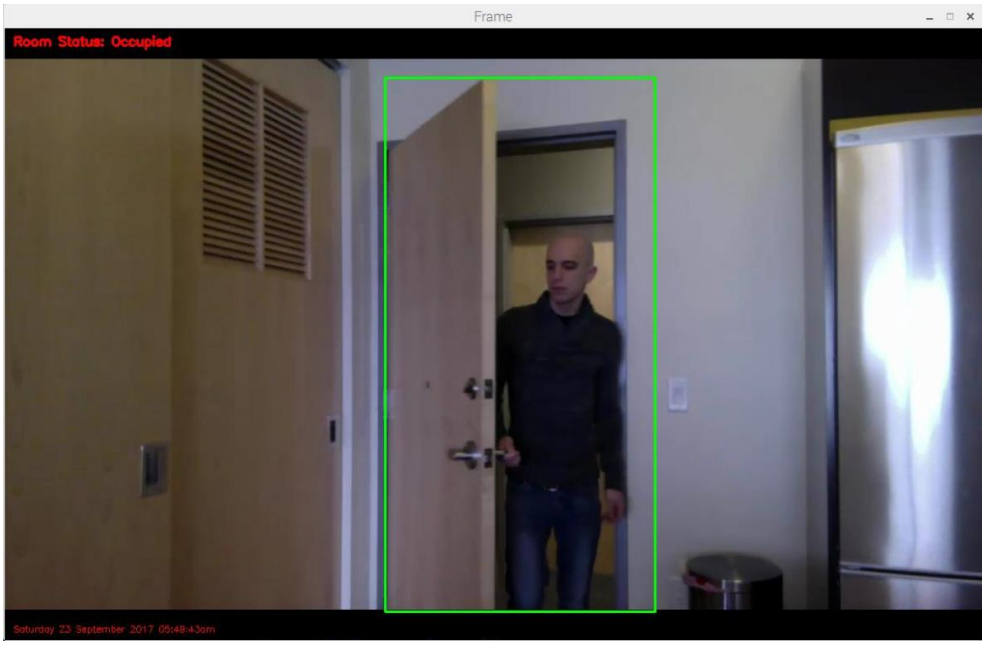
Κώδικας:

```
# import the necessary packages
import numpy as np
import argparse
import datetime
import imutils
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=3000, help="minimum area
size")
args = vars(ap.parse_args())

# video file
camera = cv2.VideoCapture(args["video"])

# GMM background-foreground initialization
fgbg=cv2.createBackgroundSubtractorMOG2()

# loop over the frames of the video
while True:
    # grab the current frame and initialize the occupied/unoccupied text
    (grabbed, frame) = camera.read()
    text = "Unoccupied"

    # if the frame could not be grabbed, then we have reached the end
    # of the video
    if not grabbed:
        break

   # apply GMM
    fgmask=fgbg.apply(frame)

    # remove noise with erosion-dilate, then find contours
    erosion=cv2.erode(fgmask,(21, 21),iterations=2)
    thresh=cv2.dilate(erosion,(21, 21),iterations=2)
    (_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # find the max contour
    if len(cnts) > 0:
            # sort the contours and find the largest one
            cnt = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
```

```
        if cv2.contourArea(cnt) > args["min_area"]:
            # compute the bounding box for the contour, draw it on the frame,
            # and update the text
            (x, y, w, h) = cv2.boundingRect(cnt)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            text = "Occupied"

    # draw the text and timestamp on the frame
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)


    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", thresh)

    k=cv2.waitKey(1) & 0xff
    if k==ord("q"):
            break

# cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
```

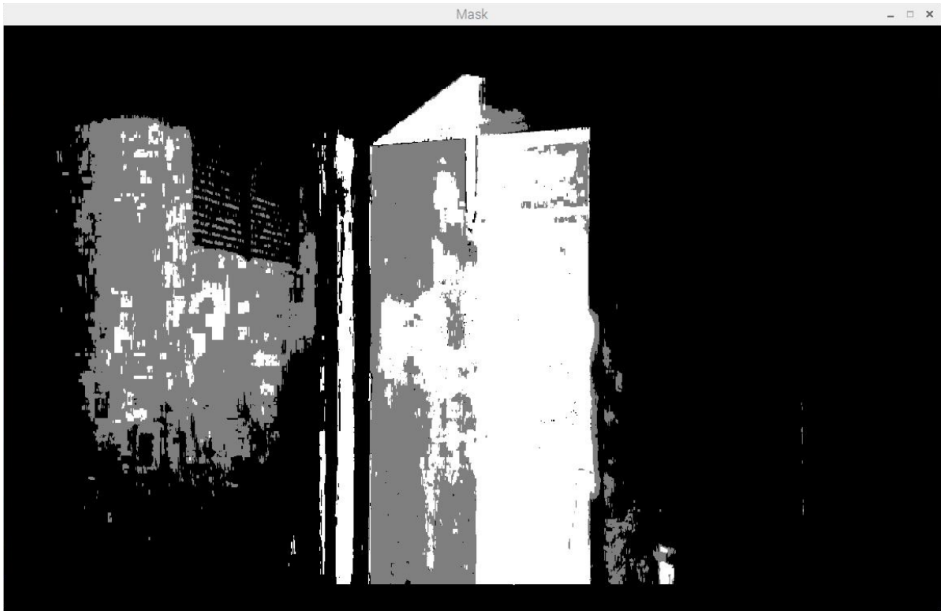
Τρέχουμε τον κώδικα με την εντολή : python motion_detector_3.py –video videos/example_01.mp4

Και έχουμε:



Εικόνα 25: Εμφάνιση κίνησης στο αρχικό παράθυρο.

Θα χρησιμοποιήσουμε τη στιγμή που εμφανίζεται κίνηση και παρακάτω θα δούμε την μάσκα που βγήκε από τη χρήση της συνάρτησης createBackgroundSubtractorMOG2:



Εικόνα 26: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG2.

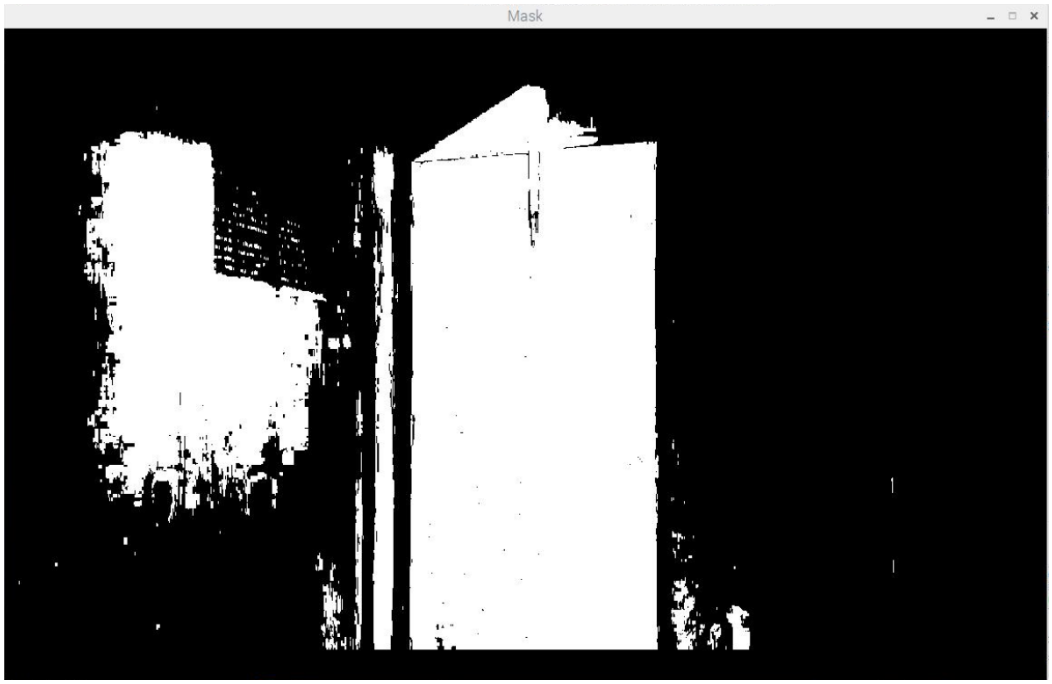
Λίγα δευτερόλεπτα αργότερα:

Εικόνα 27: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG2.(2)

Βλέπουμε ότι η συγκεκριμένη τεχνική είναι αρκετά ευαίσθητη στην κίνηση. Παρατηρούμε, επίσης, κάποια γκρι πεδία στο ασπρόμαυρο της μάσκας. Αυτό οφείλεται στην πάράμετρο “detectShadows” της συνάρτησης createBackgroundSubtractorMOG2. Η συγκεκριμένη λειτουργία μπορεί να φανεί εξαιρετικά χρήσιμη σε ένα μεγάλο αριθμό εφαρμογών αλλά μειώνει και την ταχύτητα επεξεργασίας.

Μπορούμε να «κλείσουμε» αυτή τη λειτουργία προσθέτοντας την παράμετρο “detectShadows=False” (createBackgroundSubtractorMOG2(detectShadows=False).

Αν το κάνουμε αυτό θα πάρουμε τα εξής αποτελέσματα:



Εικόνα 28: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG2 και detectShadows=False. .

Και λίγα δευτερόλεπτα αργότερα:

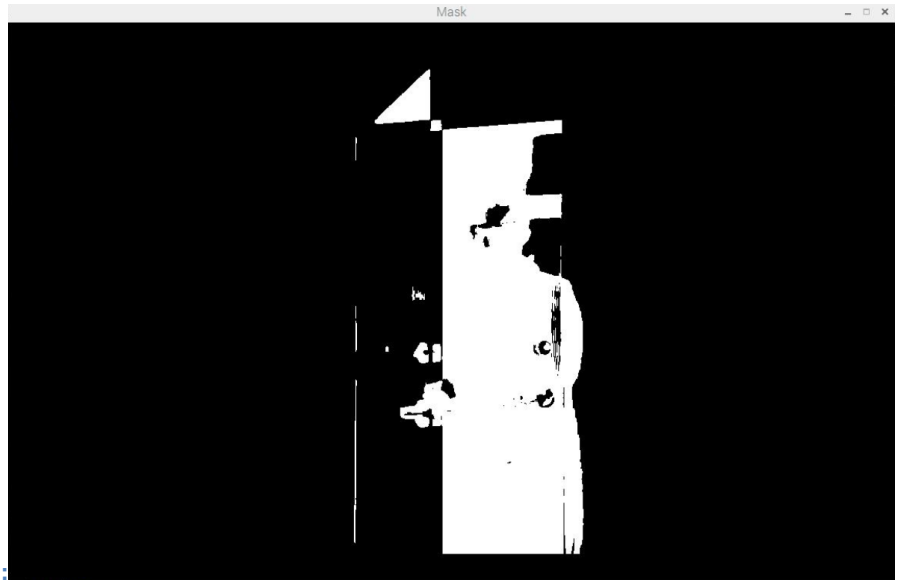
Εικόνα 29: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG2 και detectShadows=False.(2)

Παρατηρούμε πως τα γκρι πεδία δεν υπάρχουν πλέον αυξάνοντας και την ταχύτητα του αλγορίθμου.

Παρακάτω θα δούμε την μάσκα που δημιουργήθηκε με την εφαρμογή της συνάρτησης createBackgroundSubtractorMOG:

Εικόνα 30: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG.

Λίγο αργότερα:



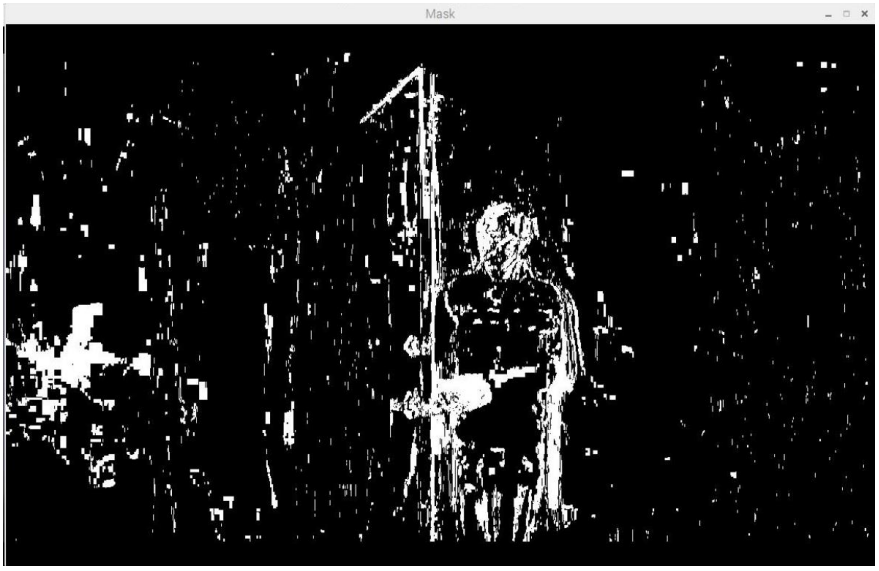
Εικόνα 31: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorMOG.(2)

Παρατηρούμε πως με τη χρήση της συγκεκριμένης συνάρτησης έχουμε σημαντική μείωση στο θόρυβο, σε σύγκριση με την προηγούμενη, αλλά και στην ταχύτητα.

Τέλος, θα δούμε τα αποτελέσματα που πήραμε με τη χρήση της συνάρτησης createBackgroundSubtractorGMG:

Εικόνα 32: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorGMG.

Και αμέσως μετά:



Εικόνα 33: Μάσκα με τη χρήση της συνάρτησης createBackgroundSubtractorGMG.(2)

Βλέπουμε πως και σε αυτήν την τεχνική ο θόρυβος είναι ορατός.

6.5 Αποτελέσματα ανίχνευσης κίνησης με τη χρήση στατιστικών μεθόδων σε ζωντανή ροή βίντεο με τη χρήση της Raspberry Pi Camera.

Για την εισαγωγή της κάμερας ως μέσο για την λήψη των καρέ ως προς επεξεργασία έγιναν κάποιες προσθήκες στον κώδικα. Οι προσθήκες φαίνονται παρακάτω:

Εισαγωγή των απαιτούμενων βιβλιοθηκων και αρχικοποίηση:

```
from picamera.array import PiRGBArray
from picamera import PiCamera
from gpiozero import *
```

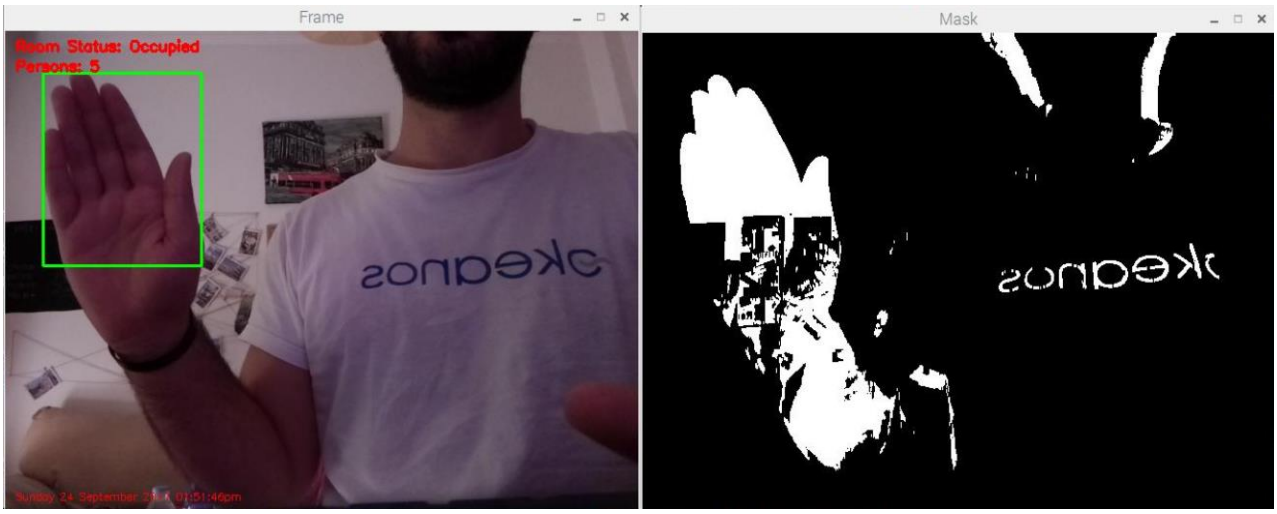
```
cam=PiCamera()
cam.resolution=(640,480)
cam.framerate=32
rawCapture=PiRGBArray(cam,size=(640,480))
cam.vflip = True
```

Όπως βλέπουμε, θέτουμε την ανάλυση που επιθυμούμε, 640x480 στην περίπτωση αυτή, και τον ρυθμό καρέ σε 32. Η σειρά `cam.vflip = True` υπάρχει για την κάθετη αναστροφή της κάμερας.

Στο κύριο σώμα του κώδικα προσθέτουμε:

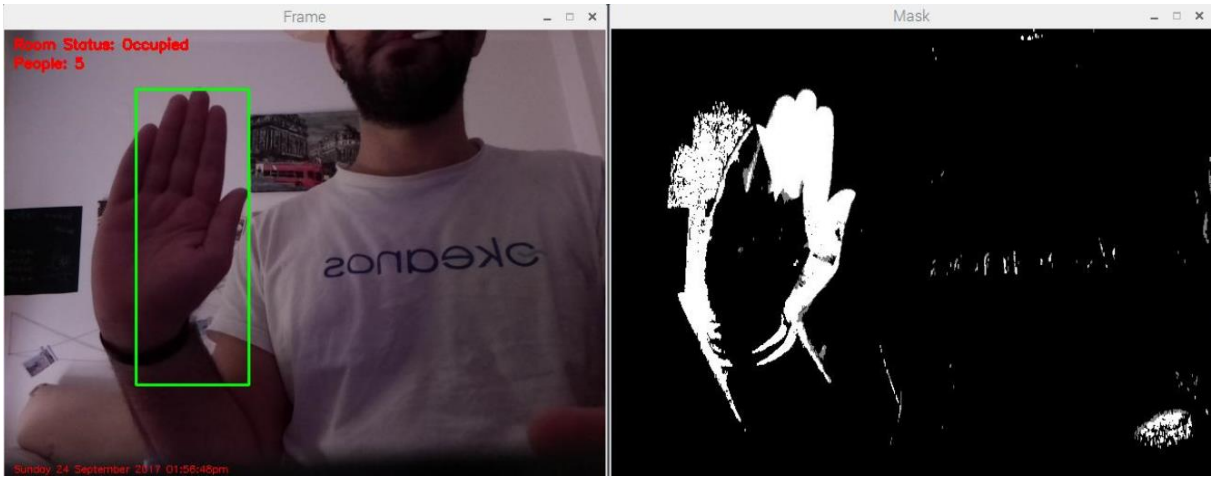
```
cam.capture(rawCapture, format="bgr", use_video_port=True)
    frame=rawCapture.array
    rawCapture.truncate(0)
```

Ο υπόλοιπος κώδικας μένει ως έχει. Για την εκτέλεση του δεν θα προσθέσουμε την παράμετρο –video, εάν επιθυμούμε ζωντανή ροή μέσω της κάμερας. Μετά την εκτέλεση, παίρνουμε τα αποτελέσματα:



Εικόνα 34: Αποτέλεσμα χρήσης της συνάρτησης createBackgroundSubtractorMOG() σε ζωντανή ροή βίντεο.

Εικόνα 35: Αποτέλεσμα χρήσης της συνάρτησης createBackgroundSubtractorGMG() σε ζωντανή ροή βίντεο.

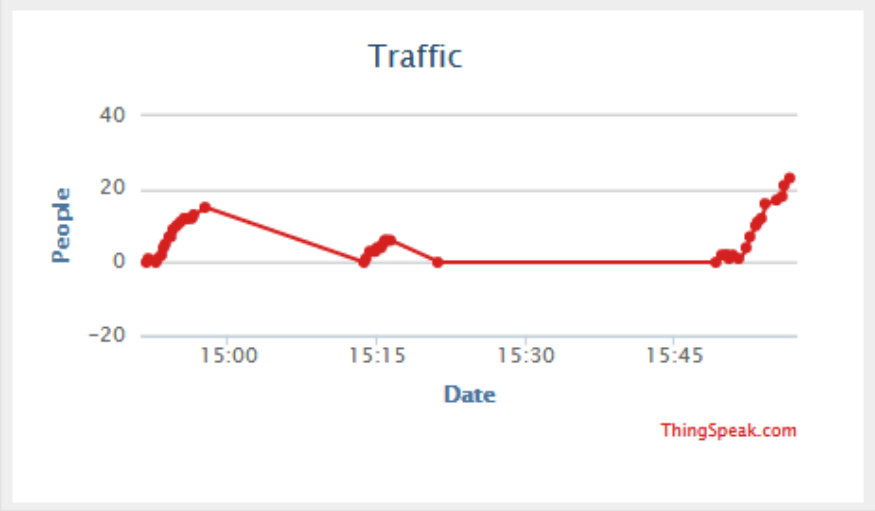


Εικόνα 36: Αποτέλεσμα χρήσης της συνάρτησης createBackgroundSubtractorMOG2() σε ζωντανή ροή βίντεο.

Το κείμενο “People” που βλέπουμε στο παράθυρο “Frame” έχει προστεθεί για να δείχνει τον αριθμό των ατόμων που ανιχνέυει ο αλγόριθμος.

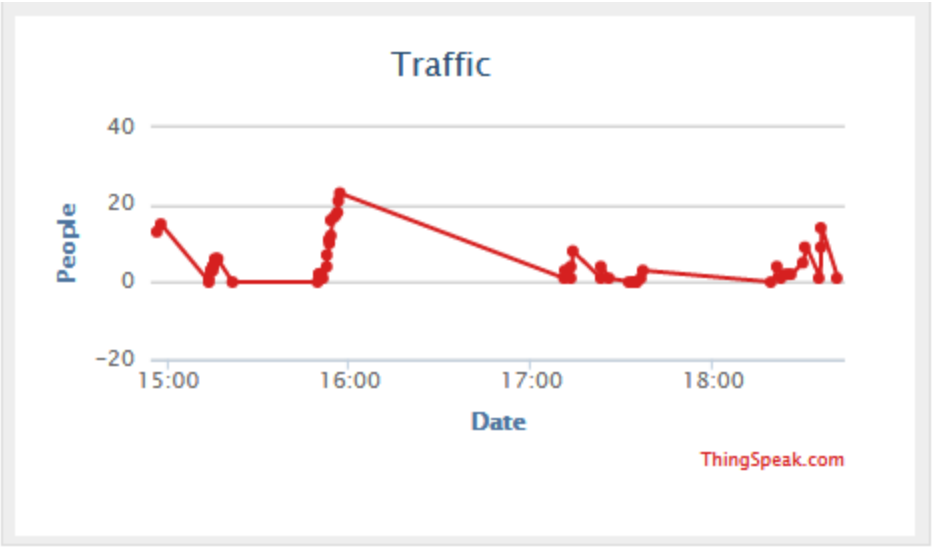
Τέλος, τα δεδομένα μεταφορτώνονται στην πλατφόρμα Thingspeak και δημιουργείται ένα γράφημα το οποίο περιέχει πληροφορίες για τον αριθμό των ατόμων που ανιχνεύθηκαν από την κάμερα καθώς και την ημερομηνία.

Με αυτόν τον τρόπο παίρνουμε πληροφορίες για την κίνηση που υπάρχει σε συγκεκριμένες ώρες. Το γράφημα που προκύπτει είναι ως εξής:



Εικόνα 37: Γράφημα με πληροφορίες για την κίνηση των ατόμων.

Βλέπουμε πως το γράφημα δείχνει τον αριθμό των ατόμων σε συνάρτηση με το χρόνο. Μετά από λίγη ώρα το γράφημα θα μοιάζει όπως θα δούμε παρακάτω:



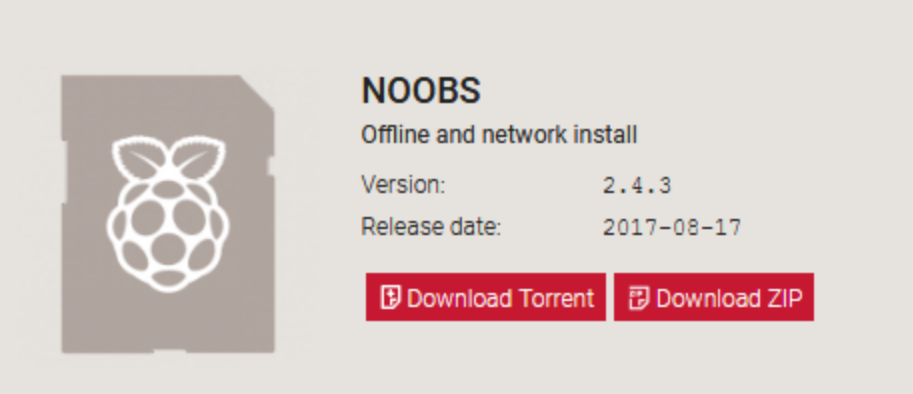
Εικόνα 38: Γράφημα με πληροφορίες για την κίνηση των ατόμων.(2)

ΠΑΡΑΡΤΗΜΑ Α

Πορεία εργασίας

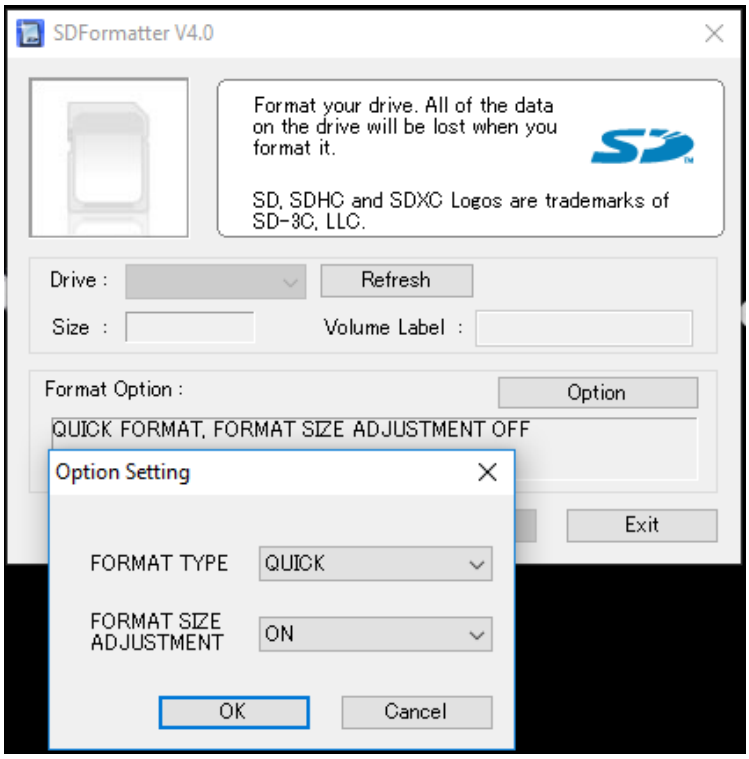
1. Εγκατάσταση συστήματος NOOBS.

Ξεκινώντας την εργασία θα χρειαστούμε το Raspberry Pi, μια SD Card τουλάχιστον 8GB και Class 4 ένα USB Stick καθώς και τροφοδοσία για το Raspberry , ένα ποντίκι, ένα πληκτρολόγιο και ένα καλώδιο HDMI. Θα χρειαστεί επίσης να συνδεθούμε στο διαδίκτυο, ασύρματα μέσω WiFi ή με τη χρήση ενός Ethernet καλωδίου.
Αρχικά πρέπει να εγκαταστήσουμε το σύστημα NOOBS στην κάρτα SD.
Κατεβάζουμε το zip αρχείο NOOBS που βρίσκεται στη σελίδα του Raspberry στον υπολογιστή μας.
https://www.raspberrypi.org/downloads/noobs/



Εικόνα 39: Αρχείο zip NOOBS

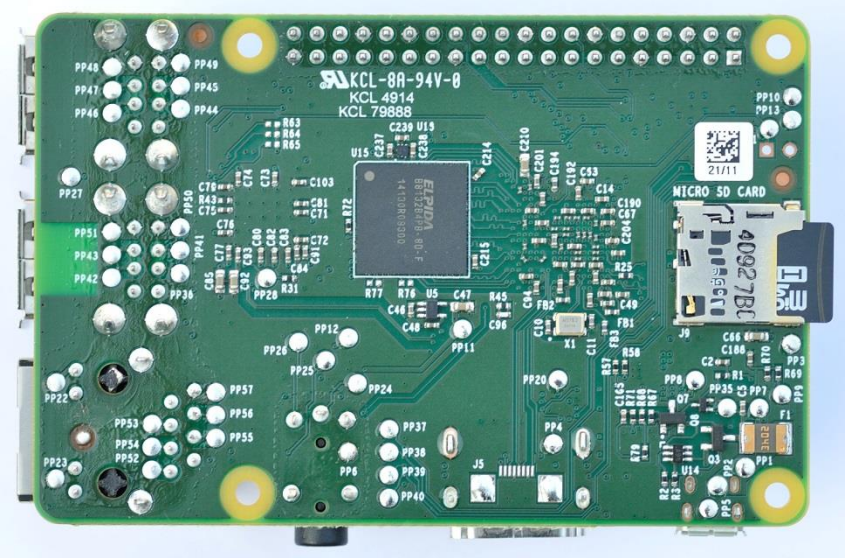
Για να περάσπυμε τα αρχεία στην κάρτα SD πρέπει την διαμορφώσουμε χρησιμοποιώντας το πρόγραμμα SDFormatter που θα κατεβάσουμε από τη σελίδα:
https://www.sdcard.org/downloads/formatter_4/eula_windows/
Αφού κατεβάσουμε και ρυθμίσουμε το πρόγραμμα προχωράμε στη διαμόρφωση της κάρτας:



Εικόνα 40: Ρύθμιση προγράμματος SDFormatter

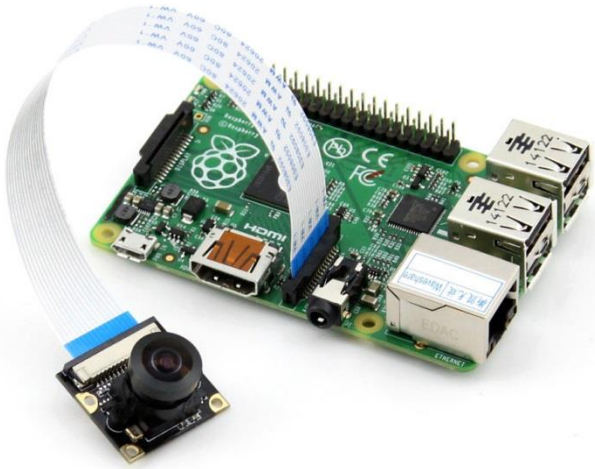
Στη συνέχεια αποσυμπιέζουμε το αρχείο και προσθέτουμε τα περιεχόμενα στην κάρτα SD.

Αφού τελειώσει η αντιγραφή των αρχείων βγάζουμε την κάρτα από τον υπολογιστή και την τοποθετούμε στο Raspberry.



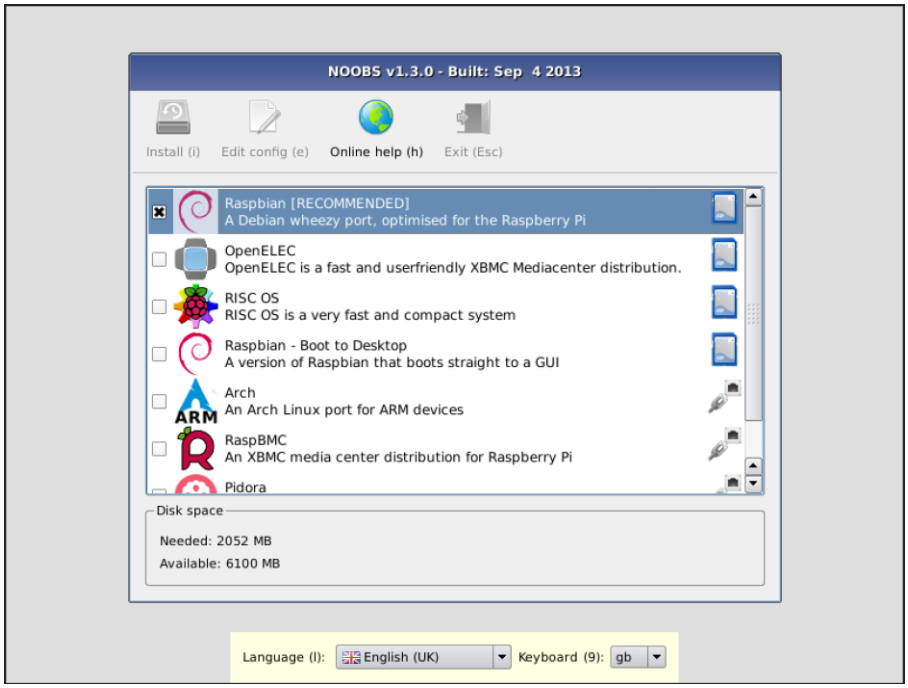
Εικόνα 41: Τοποθέτηση κάρτας SD στο Raspberry

Επίσης θα χρειαστεί να χρησιμοποιήσουμε και την Raspberry Pi Camera την οποία μπορούμε αν τοποθετήσουμε πριν συνδέσουμε το Raspberry με την τροφοδοσία.



Εικόνα 42: Σύνδεση κάμερας με το Raspberry

Στη συνέχεια συνδέουμε το Raspberry με την τροφοδοσία και σύντομα θα φορτώσει το σύστημα NOOBS και θα δούμε τη λίστα με τα λειτουργικά συστήματα που μπορούμε να εγκαταστήσουμε.



Εικόνα 43: Σύστημα NOOBS επιλογή λογισμικού

Στην παρούσα εργασία θα εγκαταστήσουμε το Raspbian (το συγκεκριμένο λόγισμικό δεν χρειάζεται σύνδεση στο διαδίκτυο). Αφού λάβουμε το μήνυμα πως το λειτουργικό εγκαταστήθηκε το Raspberry θα κάνει επανεκίννηση.

2. Ενεργοποίηση Κάμερας

Αρχικά θα σιγουρευτούμε πως η κάμερα δουλέυει κανονικά και δεν υπάρχει λάθος στη σύνδεση. Για να το κάνουμε αυτό θα ανοίξουμε το τερματικό (terminal) και θα πληκτρολογήσουμε τις εντολές:

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

Θα χρειαστεί σύνδεση στο διαδίκτυο για να λειτουργήσουν σωστά οι εντολές.

Μετά πρέπει να ενεργοποιήσουμε την υποστήριξη κάμερας χρησιμοποιώντας την εντολή:

$ sudo raspi-config

Κατευθυνόμαστε στην επιλογή για την κάμερα και την αλλάζουμε από “disabled” σε “enabled” για να την ενεργοποιήσουμε. Θα μας ζητήσει να γίνει επανεκίννηση, μετά μπορούμε να ελέγξουμε αν λειτουργεί η κάμερα πληκτρολογώντας την εντολή:

$ raspistill –v –o test.jpg

Αν η κάμερα λειτουργεί θα μπορέσουμε να δούμε μια προεπισκόπηση από αυτήν και στη συνέχεια θα βγάλει μια φωτογραφία αποθηκευμένη ως test.jpg.

3. Εγκατάσταση βιβλιοθήκης OpenCV

Ξεκινώντας θα εγκαταστήσουμε τα απαραίτητα εργαλεία προγραμματιστή πληκτολογώντας στο τερματικό (terminal):

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

Εγκατάσταση πακέτων εισόδου/εξόδου που επιτρέπουν διάφορες μορφές εικόνας να φορτωθούν από το δίσκο:

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev

Εγκατάσταση πακέτων εισόδου/εξόδου που επιτρέπουν διάφορες μορφές βίντεο να φορτωθούν από το δίσκο αλλά και μας επιτρέπυν να δουλέψουμε με ζωντανές ροές βίντεο:

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

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

Εγκατάσταση βιβλιοθήκης GTK:

$ sudo apt-get install libgtk2.0-dev

Εγκατάσταση πακέτου βελτιστοποίησης της βιβλιοθήκης OpenCV:

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

Εγκατάσταση γλώσσας προγραμματισμού Python, έκδοση 2,7 και 3:

```
$ sudo apt-get install python2.7-dev python3-dev
```

Τα βασικά εξαρτήματα είναι εγκατεστημένα. Τώρα θα κατεβάσουμε τη βιβλιοθηκη OpenCV από το GitHub (ελεύθερη αποθήκη για προγράμματα ανοικτού κώδικα):

```
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
$ unzip opencv.zip
```

Επίσης θα χρειαστούμε και το opencv_contrib:

```
$ wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
$ unzip opencv_contrib.zip
```

Εγκατάσταση pip (διαχειριστής πακέτων της γλώσσας Pyhton)

```
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
```

Εγκατάσταση εικονικού περιβάλλοντος virtualenv και virtualenvwrapper:

```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm –rf ~/.cache.pip
$ source ~/.profile
```

```
$ mkvirtualenv cv
```

Αφού εγκαταστήσουμε το εικονικό περιβάλλον μπαίνουμε σε αυτό:

```
$ workon cv
```

Και στη συνέχεια εγκαθιστούμε τη βιβλιοθήκη Numpy

```
$ pip install numpy
```

Όταν τελειώσει και η εγκατάσταση της βιβλιοθήκης Numpy προχωράμε στη μεταγλώττιση και την εγκατάσταση της βιβλιοθήκης OpenCV αφού βεβαιωθούμε πως βρισκόμαστε στο εικονικό περιβάλλον cv:

```
$ cd ~/opencv-3.1.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

```
$ make –j4
```

(η εντολή –j4 ελέγχει τον αριθμό των πυρήνων που θα τρέξουν καθώς μεταγλωττίζεται η βιβλιοθήκη)

```
$ sudo make install
$ sudo ldconfig
$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
```

```
$ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

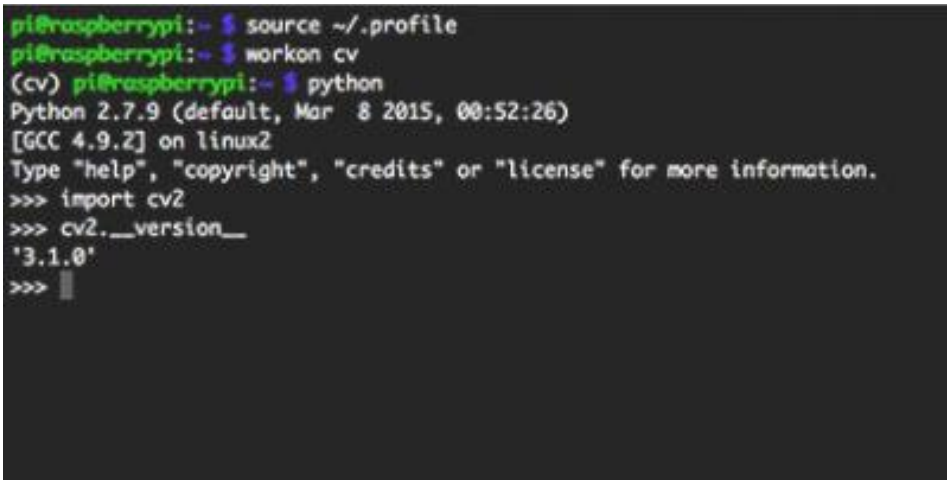
Για να σιγουρευτούμε πως όλα έχουν πάει καλά τρέχουμε τις εντολές:

```
$ source ~/.profile
$ workon cv
$ python
```

Και στο περιβάλλον της python γράφουμε;

```
>> import cv2
>> cv2.__version__
```

Και θα μπορέσουμε να δούμε τον αριθμό της έκδοσης αν η εγκατάσταση ολοκληρώθηκε επιτυχώς.



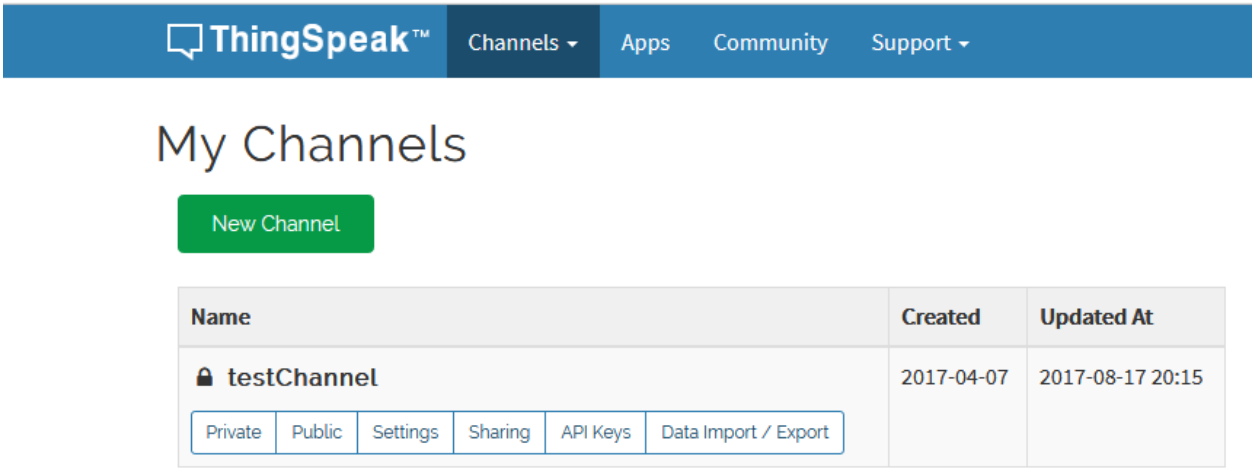
Εικόνα 44: Έλεγχος έκδοσης OpenCV

3. Εγγραφή στην εφαρμογή ThingSpeak

Θα χρησιμοποιήσουμε την εφαρμογή ThingSpeak για την επεγεργασία των δεδομένων και την ανάλυσή τους.

Αρχικά πρέπει να δημιουργήσουμε ενα λογαριασμό στην εταιρία MathWorks ακολουθώντας τον παρακάτω σύνδεσμο:
https://thingspeak.com/users/sign_up

Αφού δημιουργήσουμε τον λογαριασμό μπορούμε να ξεκινήσουμε τη δημιουργία του δικού μας καναλιού το οποίο θα χρειαστούμε για να ανεβάσουμε τα δεδομένα. Για τη δημιουργία του καναλιού μπαίνουμε στον λογαριασμό μας στην εφαρμογή ThingSpeak και επιλέγουμε “New Channel”



Εικόνα 45: Δημιουργία καναλιού στην εφαρμογή ThingSpeak.

ΠΑΡΑΡΤΗΜΑ Β

Τελικός κώδικας εργασίας

Έχουμε δημιουργήσει τρείς εκδοχές του κώδικα δύο εκ των οποίων είναι αλλάζουν κατά πολύ λίγο. Η πρώτη εκδοχή είναι η εξής:

```
# import the necessary packages
import numpy as np
import argparse
import datetime
import imutils
import urllib
import httplib
import time
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
from gpiozero import *

cam=PiCamera()
cam.resolution=(640,480)
cam.framerate=32
rawCapture=PiRGBArray(cam,size=(640,480))
cam.vflip = True

time.sleep(1)

# initialize the average frame
avg = None
min_area = 2000

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=3000, help="minimum area
size")
args = vars(ap.parse_args())

# video file
camera = cv2.VideoCapture(args["video"])

# GMM background-foreground initialization
fgbg=cv2.bgsegm.createBackgroundSubtractorMOG()

# initialize variables
flag = 0
movement = 0
ppl = 0
```

```
# loop over the frames of the video or the live stream
while True:

    # grab the current frame and initialize the occupied/unoccupied text
    (grabbed, frame) = camera.read()
    text = "Unoccupied"

    # if the frame could not be grabbed, then read from camera
    if not grabbed:
        cam.capture(rawCapture, format="bgr", use_video_port=True)
            frame=rawCapture.array
            rawCapture.truncate(0)

    # apply GMM (Gaussian Mixture Model)
    fgmask=fgbg.apply(frame)

    # remove noise with erosion-dilate, then find contours
    erosion=cv2.erode(fgmask,(21, 21),iterations=2)
    thresh=cv2.dilate(erosion,(21, 21),iterations=2)
    (_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    # find the max contour
    if len(cnts) > 0:

            # sort the contours and find the largest one
            cnt =    sorted(cnts, key = cv2.contourArea, reverse = True)[0]

        if cv2.contourArea(cnt) > args["min_area"]:

                    # compute the bounding box for the contour, draw it on the frame,
                    # and update the text
                    (x, y, w, h) = cv2.boundingRect(cnt)
                    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                    text = "Occupied"
                    flag = 1
                    movement = 1

                else:
                    flag = 0



        if movement == 1:
            if flag == 0:
```

```
                movement = 0
                ppl +=1

                #Upload data to ThingSpeak
                f = urllib.urlencode({'field1' : ppl,'key' : 'B4LDXX5PEE7HHBKI' })
                headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
                conn=httplib.HTTPConnection("api.thingspeak.com:80")

                try:
                        conn.request("POST", "/update", f, headers)
                        response=conn.getresponse()
                        print response.status, response.reason
                        data = response.read()
                        conn.close()
                except:
                        print "Connection Failed"


    # draw the text and timestamp on the frame
    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
    cv2.putText(frame, "People: {}" .format(ppl), (10,40),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", thresh)

    k=cv2.waitKey(1) & 0xff
    if k==ord("q"):
            break

# cleanup the camera and close any open windows
cv2.destroyAllWindows()
```

Στη συγκεκριμένη εκδοχή για την καταμέτρηση των ατόμων χρησιμοποιούμε την μεταβλητή ppl. Επίσης, έχουμε τις μεταβλητές movement και flag. Και οι δύο μεταβλητές περιέχουν την πληροφορία του αν υπάρχει κίνηση ή όχι στο καρέ του βίντεο. Η movement χρησιμοποιείται για ξέρουμε την κίνηση και του προηγούμενου καρέ. Για να αυξηθέι ο μετρητής πρέπει η μεταβλητή movement να είναι 1 και η

μεταβλητή flag να είναι 0, δηλαδή να μην υπάρχει κίνηση στο τρέχον καρέ αλλά να υπήρχε στο προηγούμενο. Άρα, με το που χαθεί η κίνηση που υπήρχε ο μετρητής αυξάνεται κατά ένα.

Στη δεύτερη εκδοχή η ιδέα είναι η ίδια αλλά ο μετρητής αυξάνεται όταν υπάρχει κίνηση στο τρέχον καρέ αλλά δεν υπήρχε και στο προηγούμενο.

Σε αυτές τις δύο εκδοχές παρατηρήθηκε μικρή καθυστέρηση κατα την καταμέτρηση και τη μεταφόρτωση των δεδομένων.

Στην Τρίτη εκδοχή παραλλάξαμε λίγο τον κώδικα έτσι ώστε να μετράει κανονικά, όπως η δεύτερη εκδοχή, αλλά να μεταφορτώνει τα δεδομένα μετά από συγκεκριμένο χρονικό διάστημα. Το σημείο του κώδικα που εκτελεί αυτή τη λειτουργία είναι το παρακάτω:

```
    #Upload data after an amount of time

    if time_delay < datetime.datetime.now():

        now = datetime.datetime.now()
        time_delay = now + datetime.timedelta(minutes = 15)


        #Upload data to ThingSpeak
        f = urllib.urlencode({'field1' : ppl,'key' : 'B4LDXX5PEE7HHBKI' })
        headers = {"Content-typZZe": "application/x-www-form-urlencoded","Accept":
"text/plain"}
        conn=httplib.HTTPConnection("api.thingspeak.com:80")

        try:
            conn.request("POST", "/update", f, headers)
            response=conn.getresponse()
            print response.status, response.reason
            data = response.read()
            conn.close()
            ppl = 0
        except:
            print "Connection Failed"
```

Στην αρχή του κώδικα έχουμε αρχικοποιήσει τις μεταβλητές:

```
now = datetime.datetime.now()
time_delay = now + datetime.timedelta(minutes = 15)
```

Έχουμε το επιθυμητό χρονικό όριο στα δεκαπέντε λεπτά, συνεπώς μετά από δεκαπέντε λεπτά τα δεδομένα θα σταλθούν στην πλατφόρμα ThingSpeak και ο μετρητής θα μηδενιστεί ξανά. Έτσι, αναλύουμε την κίνηση ανα δεκαπέντε λεπτά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Αναφορά σε πτυχιακές εργασίες:

Πρωτονοτάριος Ιωάννης (2011). Ανάπτυξη ενσωματωμένων αισθητήρων για ασύρματη μετάδοση εικόνας και δεδομένων. Τμήμα ηλεκτρολόγων μηχανικών, Πανεπιστήμιο Πατρών.

Γιαννόπουλος Ευθύμιος (2014). Σχεδιασμός – υλοποίηση ενσωματωμένου συστήματος για κίνηση και έλεγχος οχήματος με τη βοήθεια ασύρματης επικοινωνίας. Τμήμα ηλεκτρολόγων μηχνανικών και τεχνολογίας υπολογιστών,Πανεπιστήμιο Πατρών.

Παύλος Ε. Αλεξανδράκης (2017). Κίνηση οχήματος με Raspberry Pi και Line Camera. Τμήμα μηχανικών ηλεκτρονικών υπολογιστών, ΑΕΙ Πειραιά Τ.Τ.

Αναστασία Αγγελίδου, Θωμαή Τσούκα (2013). Ανάπτυξη λογισμικού για την αναγνώριση οπτικών σημάτων μέσω βλεφαρίσματος. Τμήμα μηχανικών πληροφορικής. ΤΕΙ Κεντρικής Μακεδονίας.

Τσούρμα Μαρία (2016). Σχεδιασμός και κατασκευή ενός υβριδικού αμυντικού αυτοματοποιημένου ρομποτικού συστήματος. Τμήμα μηχανικών πληροφορικής και τηλεπικοινωνιών, Πανεπιστήμιο δυτικής Μακεδονίας.

Αρβανίτη Γερασίμου (2012). Ανίχνευση και παρακολούθηση κίνησης (motion detection and tracking). Τμήμα ηλεκτρονικής και επεξεργασίας της πληροφορίας, Πανεπιστήμιο Πατρών.

Σταμούλη Μαρία (2008). Εξαγωγή κινούμενου αντικειμένου σε βίντεο. Τμήμα μηχανικών υπολογιστών και πληροφορικής. Πανεπιστήμιο Πατρών.

David Andrew Ryan (2013). Crow monitoring using computer vision (PhD Thesis). Faculty of built environment and engineering, Queensland University of Technology.

Αναφορά σε ιστοσελίδες:

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3
http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/

Python
https://el.wikipedia.org/wiki/Python#.CE.99.CF.83.CF.84.CE.BF.CF.81.CE.B9.CE.BA.CF.8C

Μηχανική όραση
https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CF%8C%CF%81%CE%B1%CF%83%CE%B7

OpenCV: Background Subtraction
http://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html#gsc.tab=0

Raspberry Pi
https://en.wikipedia.org/wiki/Raspberry_Pi

OpenCV
https://en.wikipedia.org/wiki/OpenCV

Μηχανική Όραση

https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CF%8C%CF%81%CE%B1%CF%83%CE%B7