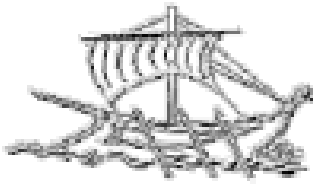


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝΤ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Καταμέτρηση και ανάλυση κυκλοφορίας με Raspberry PI και κάμερα στο Internet of Things

Ιωάννης Γαλανάκης 42926

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΜΑΡΤΙΟΣ 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Καταμέτρηση αυτοκινήτων και ανάλυση κυκλοφορίας με Raspberry PI,κάμερας και Cloud

Ιωάννης Γαλανάκης
Α.Μ. 42926

Εισηγητής:

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η ……Γαλανάκης Ιωάννης……, του ……Κωνσταντίνου……, με αριθμό μητρώου ……42926…….. φοιτητής 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

Η παρούσα πτυχιακή εργασία αποτελεί προϊόν πολλών χρόνων μελέτης και διαβάσματος καθώς και μελλοντικών ονείρων και συντελεί στην αυτοπραγμάτωση των στόχων μου που είχα θέσει πολλά χρόνια πριν στο αντικείμενο της μηχανικής και της τεχνητής νοημοσύνης καθώς η παρούσα εργασία έχει ως θεμέλια την εκμάθηση μηχανών (machine learning) και την ανίχνευση πολλαπλών αντικειμένων κίνησης μιας τεχνολογίας που έχει πολλές εφαρμογές σήμερα όπως π.χ. στα πολεμικά συστήματα (multi object detection lock).

Στην προσπάθειά μου αυτή θέλω να ευχαριστήσω θερμά τον καθηγητή μου κ. Έλληνα που με υποστήριξε σε αυτή την προσπάθεια να ολοκληρώσω το δύσκολο αλλά και πρωτότυπο αυτό έργο καθώς και την οικογένεια μου που μετά από πολλά χρόνια κόπου μου έδωσε τα απαραίτητα πνευματικά και υλικά εφόδια να φτάσω μέχρι εδώ καθώς και όλους τους καθηγητές μου που μου προσέφεραν την γνώση τους και την καθοδήγησή τους σε αυτό το μεγάλο ταξίδι.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη αλγορίθμων - μοντέλων σε Matlab και Simulink αλλά και σε κώδικα Python και την εφαρμογή αυτών στην συσκευή Raspberry Pi 3 για την πολλαπλή ανάλυση και ανίχνευση αντικειμένων συγκεκριμένου ενδιαφέροντος και την ανάλυση των στατιστικών από τα αποτελέσματα στο Internet of Things και στο ThingSpeak.

Οι τεχνικές που εφαρμόζονται για την περάτωση αυτού του έργου είναι ένα μέρος τεχνητής νοημοσύνης (Machine Learning) καθώς και εφαρμογή πολλαπλών μαθηματικών αλγορίθμων και τεχνικών (Gaussian Mixture Model, Morphological Opening, Foreground Detection, Bayesian Inference, Kalman Filter) και πολλά άλλα. Σε αρχικό στάδιο θα εφαρμόσουμε τους αλγορίθμους σε κώδικα Objective C και Matlab και σε μεταγενέστερο στάδιο σε Python για την εφαρμογή του ίδιου σκοπού και την βέλτιστη απόδοση του αλγορίθμου με μηδαμινή προσέγγιση σφάλματος

ABSTRACT

The present thesis concerns the development of algorithms-models on Matlab and Simulink and also on Python Code and the application on a Raspberry Pi 3 for multi-object detection and statistic analysis of the results on Internet of Things and ThingSpeak.

The following methods that took place for developing this project are based on Artificial Inteligence (Machine Learning) and also the application of multiple mathematical algorithms and techniques such as (Gaussian Mixture Model, Morphological Opening, Foreground Detection, Bayesian Inference, Kalman Filter) etc. On alpha stage we will apply the algorithms on Objection C and Matlab and then on Python for the application of the same purpose and the improvement of the algorithm’s accuracy with zero potential error.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Εκμάθηση Έξυπνων Συστημάτων (A.I)
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Τεχνητή Νοημοσύνη, Raspberry P, IoT Analytics, Matlab, Simulink, Cloud, MultiObject Detection.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

Raspi: Raspberry Pi

GMM: Gaussian Mixture Model

IoT: Internet of Things

ΕΙΣΑΓΩΓΗ

Rasapberry Pi

Το raspberry Pi αποτελεί μια καινοτομία που δημιουργήθηκε στο Ηνωμένο Βασίλειο από την εταιρία Raspberry Pi Foundation για εκπαιδευτικούς λόγους και κυρίως για να εισάγει την υπολογιστική μηχανική στα συστήματα εκπαίδευσης.

Η κατασκευή του, παρόλο που ήταν μια απλή πλακέτα υπολογισμών είχε τεράστια επιτυχία σε πωλήσεις εκτός του τομέα για τον οποίο προοριζόταν, όπως στη ρομποτική και στη μηχατρονική. Αυτός ήταν και ο λόγος που έγινε πολύ πιο διάσημη από ό,τι ήταν αναμενόμενο.

Στην αγορά έχουν βγει αρκετές γενιές τους RasPi με την πρώτη από αυτές να είναι το Raspberry Pi 1 Model B με επίσημη ημερομηνία κυκλοφορίας τον Φεβρουάριο του 2012. Στην συνέχεια βγήκαν διάφορες φθηνές εκδόσεις Model A, και Model B+ με λίγο πιο βελτιωμένο σχεδιασμό πλακέτας καθώς και τα μοντέλα Pi Zero (με μικρότερο μέγεθος από μια πιστωτική κάρτα) αλλά και το Pi 2 που πρόσθεσε μεγαλύτερη χωρητικότητα Ram.

Πλέον το τελευταίο μοντέλο Raspberry Pi 3 Model B (αυτό στο οποίο θα διεξαχθεί η έρευνα της παρούσας μελέτης) που κυκλοφόρησε τον Φεβρουάριο του 2016.Το Raspberry Pi 3 Model B αποτελείτε από:

-SoC: *Broadcom BCM2837*
-Επεξεργαστή CPU: *4 πυρήνων ARM Cortex-A53, 1.2GHz*
-Κάρτα γραφικών GPU: *Broadcom VideoCore IV*
-Μνήμη RAM: *1GB LPDDR2 (900 MHz)*
-Δίκτυο: *10/100 Ethernet, 2.4GHz 802.11n wireless*
-Bluetooth: *Bluetooth 4.1 Classic, Bluetooth Low Energy*
-Αποθηκευτικός χώρος: *microSD*
-GPIO: *40-πινς header, populated*
-Περιφεριακές θύρες: *HDMI, 3.5mm αναλογικό audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), οθόνη Serial Interface (DSI)*

Όλα τα μοντέλα αποτελούνται από ένα ολοκληρωμένο Broadcom system που περιλαμβάνει μια CPU αρχιτεκτονικής της εταιρίας ARM (Advanced RISC Machine) γνωστή για τους σχεδιασμούς και την αρχιτεκτονική της σε πολλές εταιρίες ολοκληρωμένων. Επίσης έχει ενσωματωμένη on-chip μονάδα επεξεργασίας

γραφικών GPU VideoCore χαμηλής κατανάλωσης από την Alpha mosaic Ltd θυγατρική της Broadcom.

Οι ταχύτητες της CPU κυμαίνονται από 700MHz μέχρι 1.2 GHz για το Pi 3 και μνήμη RAM από 256 MB μέχρι 1024 MB. Για αποθηκευτικό χώρο χρησιμοποιούν μια Secure Digital κάρτα μνήμης που χρησιμοποιείτε σαν σκληρός δίσκος για την αποθήκευση του λειτουργικού συστήματος. Οι περισσότερες πλακέτες έχουν από 1 έως 4 σειριακές θήρες USB,μία θήρα High Definition Media Interface,μίας θήρα τοπικής σύνδεσης Ethernet και μία θήρα audio jack 3.5 mm. Για περαιτέρω χρήση χαμηλής κατανάλωσης υπάρχουν τα 40 general purpose Input Output pins. Τέλος έχει ενσωματωμένο Wi-Fi 802 wireless adapter και Bluetooth.

Όπως προαναφέραμε ήδη το Raspberry Pi είχε τεράστια επιτυχία και χρησιμοποιήθηκε σε πολλές εφαρμογές ρομποτικής και γενικού ενδιαφέροντος καθώς και σε πολλά παγκόσμια projects.

Μερικά από τα πιο δημοφιλή projects που έχουν γίνει με raspberry Pi είναι:

1.Tinkernut Raspberry Pi SUPERCOMPUTER:

https://www.youtube.com/watch?v=1R0UgIgcb5g

2.Virtual Desktop with graphical desktop interface:

https://www.realvnc.com/docs/raspberry-pi.html#raspberry-pi-viewer

3.Aircraft overall control and system information:

http://stratux.me/

4.Xbox Zero Retro Gaming Console with RetroPie Emulation Software:

https://shkspr.mobi/blog/2015/11/raspberry-pi-zero-hidden-in-an-xbox-controller/

5.Raspberry Pi Cluster Server:

https://www.youtube.com/watch?v=i_r3z1jYHAc

Internet of Things (IoT)

Το Internet of Things ή αλλιώς IoT αποτελεί ένα Internetwork φυσικών συσκευών, οχημάτων, κτηρίων, που είναι ενσωματωμένα με ηλεκτρονικές συσκευές, hardware, λογισμικά και αισθητήρια τα οποία μέσω σύνδεσης internet ενεργοποιούν τη δυνατότητα αυτών των αντικειμένων να συλλέγουν και να ανταλλάσουν δεδομένα.

Internetworking είναι η τεχνική της σύνδεσης ενός δικτύου υπολογιστή με άλλα δίκτυα μέσω gateways που δίνουν μια κοινή μέθοδο δρομολόγησης των πληροφοριών των πακέτων μεταξύ τους. Το πιο γνωστό Internetworking είναι το TCP/IP Protocol.

Το IoT επιτρέπει σε αντικείμενα να σταλθούν η να ελεηθούν ασύρματα μέσω μέσο της υποδομής του υπάρχοντος δικτύου. Αποτέλεσμα αυτού είναι η καλύτερη επικοινωνία μεταξύ του ανθρώπινου παράγοντα και της μηχανής δίνοντας την δυνατότητα για άμεση επικοινωνία και καλύτερα αποτελέσματα σε θέματα, προχωρημένης απόδοσης, ακρίβειας και εργονομικά και οικονομικά προνόμια όσων αφορά την ανθρώπινη επέμβαση αφού την κάνει λιγότερη αναγκαία. Ένα πολύ καλό παράδειγμα εφαρμογής του IoT είναι ότι όταν του προσθέτουμε αισθητήρες και ενεργοποιητές, μετατρέπεται σε ένα cyber-physical σύστημα που περιλαμβάνει τεχνολογίες όπως τα ευφυή δίκτυα, τα έξυπνα σπίτια, smart phones και πολλά άλλα. Γενικά το IoT προσφέρει προχωρημένη συνδεσιμότητα μεταξύ συσκευών και συστημάτων καλύπτοντας μια μεγάλη γκάμα από πρωτόκολλα, domains και εφαρμογών.

Μερικές από τις πολυπληθής εφαρμογές του Internet of Things μπορούν να αναφέρονται σε μια ευρεία ποικιλία συσκευών όπως η παρακολούθηση ενός μοσχεύματος καρδιάς, bio-chips αναμεταδότες σε μια φάρμα με ζώα που κάθε αναμεταδότης καταγράφει την ζωτικότητα των ζώων, την ποσότητα της διαθέσιμης τροφής τους ή του νερού, τον ρυθμό μετάδοσης του νερού ενός οδοφράγματος, αυτοκίνητα με ενσωματωμένους αισθητήρες, συσκευές ανάλυσης DNA για παρακολούθηση οικολογικών και τροφικών καταστάσεων ή ακόμα και συσκευές επιχειρήσεων πυροσβεστών που τους βοηθάνε να πραγματοποιήσουν διασώσεις.

Με μια γενική εικόνα θα μπορούσαμε να χαρακτηρίσουμε το internet of things σαν ένα αξεδιάλυτο μείγμα hardware, software, δεδομένων και υπηρεσιών από συσκευές που συλλέγουν δεδομένα για να βοηθήσουν ήδη υπάρχουσες τεχνολογίες και να διανέμουν τα δεδομένα ανάμεσα σε πλήθος συσκευών. Ήδη στην αγορά υπάρχουν πολλά παραδείγματα εφαρμογής συμπεριλαμβανομένων του

αυτοματισμού σπιτιού (smart home devices), ο έλεγχος της θερμότητας, του φωτισμού και της κατανάλωσης ενέργειας του σπιτιού, εξαερισμός,air conditioning καθώς και ψύξη τροφίμων που χρησιμοποιούν ασύρματη Wi-Fi σύνδεση για απομακρυσμένη παρακολούθηση.

Σαν ιδέα το internet of things ανακαλύφθηκε από τον Peter T. Lewis τον Σεπτέμβριο του 1985 σε μια ομιλία του στην Ομοσπονδιακή Επιτροπή Επικοινωνιών της Αμερικής.

Mathworks

Η Mathworks είναι μια αμερικάνικη εταιρία που ειδικεύεται σε λογισμικά μαθηματικών υπολογισμών με τα 2 πιο μεγάλα της και γνωστά προϊόντα το Matlab και το Simulink. Το 2014 έδωσε πάνω από 3000 θέσεις εργασίας με 70% αυτού στα κεντρικά γραφεία της εταιρίας στην Μασαχουσέτη.

Ιδρύθηκε στην Portola Valley της Καλιφόρνιας από την Jack Little, Cleve Moler και Steve Bangert στης 5 Δεκεμβρίου το 1984. Το Matlab έκανε την πρώτη του εμφάνιση στο διάσκεψη της IEEE στην Νεβάδα, Las Vegas την ίδια χρονιά. Η εταιρία πούλησε την πρώτη της παραγγελία με 10 αντίγραφα του Matlab στο MIT (Massachusetts Institute of Technology) τον φεβρουάριο του 1985.

Το 1986 η Mathworks μεταφέρθηκε στα μέχρι και τώρα κεντρικά γραφεία της στο Natick τον Ιούλιο του 1999. Το 2007 απέκτησε την Polyspace Technologies και ξεκίνησε να διαθέτει προϊόντας της στις εκδόσεις Matlab το 2008. Έπειτα συνέχισε με την SciFace Sofware και το 2013 την Steepest Ascent.

ThingSpeak

Το ThingSPeak είναι μια Internet of Things πλατφόρμα της Mathworks που επιτρέπει την live συγκέντρωση, απεικόνιση και ανάλυση δεδομένων που υπάρχουν στο cloud. Το thing speak παρέχει άμεση απεικόνιση των δεδομένων που έχουν αναρτηθεί από τις συσκευές του χρήστη στο ThingSpeak. Με την δυνατότητα να εκτελεί κώδικα σε Matlab κατευθείαν στο ThingSpeak ώστε να γίνει online ανάλυση και επεξεργασία των δεδομένων που εισέρχονται από την συσκευή, καθιστά το ThingSpeak μια πολύ δυνατή και χρήσιμη IoT πλατφόρμα υπηρεσιών.

Περισσότερες δυνατότητες του ThingSpeak:

-***Εύκολη ρύθμιση των συσκευών*** *στην αποστολή δεδομένων χρησιμοποιώντας δημοφιλή IoT Protocols.*

-***Real Time απεικόνιση των δεδομένων*** *που εισάχθηκαν από τους αισθητήρες της συσκευής*

-***Συγκεντρωτικά δεδομένα*** *κατόπιν εντολής από 3rd party πηγές*

-***Χρήση του Matlab*** *για ανάλυση και κατανόηση των δεδομένων*

-***Αυτόματη εκτέλεση των αναλύσεων του IoT*** *βασισμένα σε συγκεκριμένα χρονικά διαγράμματα.*

-***Πρωτότυπα IoT συστήματα*** *χωρίς την ανάγκη servers για την ανάπτυξη κώδικα software*

-***Αυτόματη κίνηση των δεδομένων*** *και επικοινωνία χρησιμοποιώντας 3rd party υπηρεσίες όπως το Twitter*

Σύμφωνα με τους προγραμματιστές το ThingSpeak είναι μια open source IoT πλατφόρμα API για την αποθήκευση και ανάκτηση δεδομένων χρησιμοποιώντας το HTTP πρωτόκολλο μέσω ιντερνέτ ή τοπικής σύνδεσης. Το Thing Speak κυκλοφόρησε για πρώτη φορά το 2010 από την ioBridge σε μια προσπάθεια υποστήριξης εφαρμογών του Internet of Things.

API είναι ένα σετ από υπορουτίνες και συναρτήσεις, πρωτόκολλα και εργαλεία για την ανάπτυξη software εφαρμογών

Κεφάλαια

Η Κατασκευή

Η κατασκευή για την παρούσα μελέτη είναι πολύ απλή. Ένα Raspberry pi 3 model B μέσα σε μία θήκη από Plexiglas για να το προστατέψουμε από εξωτερικούς κραδασμούς, σκόνη και άλλα. Τοποθετούμε την κάμερα του Raspberry Pi σε ένα τρίποδο για φωτογραφική μηχανή και δένουμε στα δύο από τα τρία πόδια της βάσης το Raspberry με tie wrap. Τέλος συνδέουμε την καλώδιοταινία με το Raspi και την κάμερα του. Το αποτέλεσμα είναι ακόλουθο:

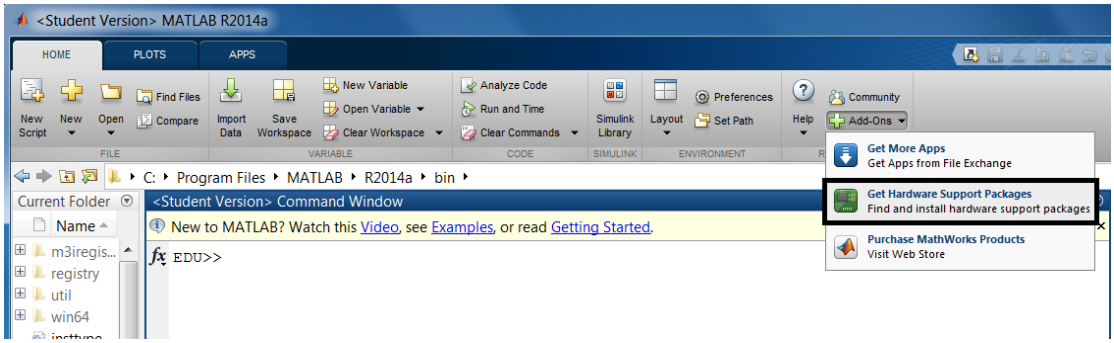
1.1 Κατασκευή raspberry pi και κάμερας σε τρίποδο

1.2 Κατασκευή raspberry pi και κάμερας σε τρίποδο

Raspberry Pi Hardware Support Package Installation

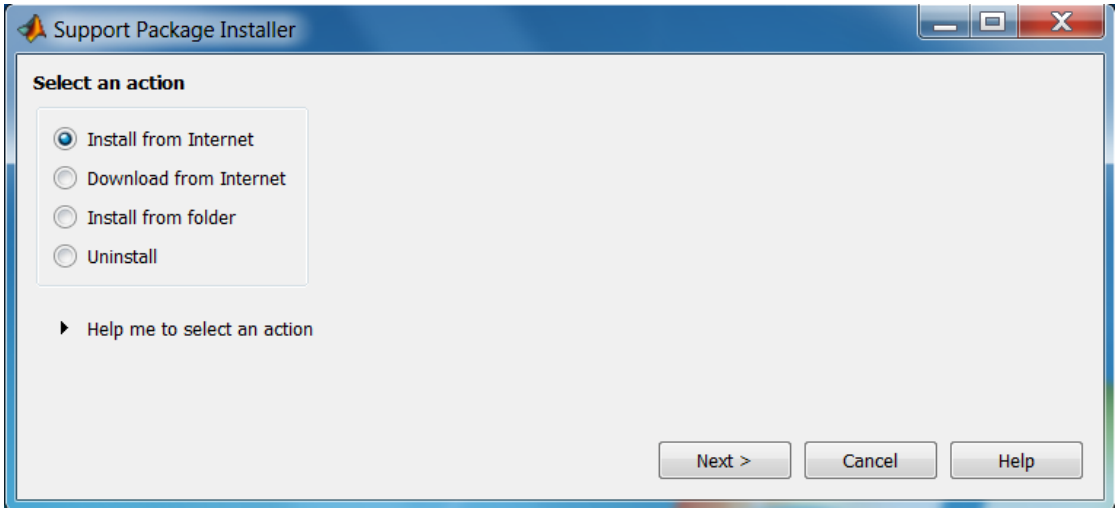
Για να μπορέσουμε να ρυθμίσουμε το Raspberry Pi ώστε να δέχεται μοντέλα Simulink και κώδικα Matlab και να τα αναγνωρίζει ώστε να εκτελεί τις λειτουργίες που εμείς θέλουμε πρέπει πρώτα να εγκαταστήσουμε τα κατάλληλα πακέτα Raspi 3 Hardware Support Packages.

-Σε έναν υπολογιστή με εγκατεστημένο το Matlab και το Simulink με έκδοση 2016^{α} και ανοίγουμε στα Add-Ons την επιλογή Get Hardware Support Packages.



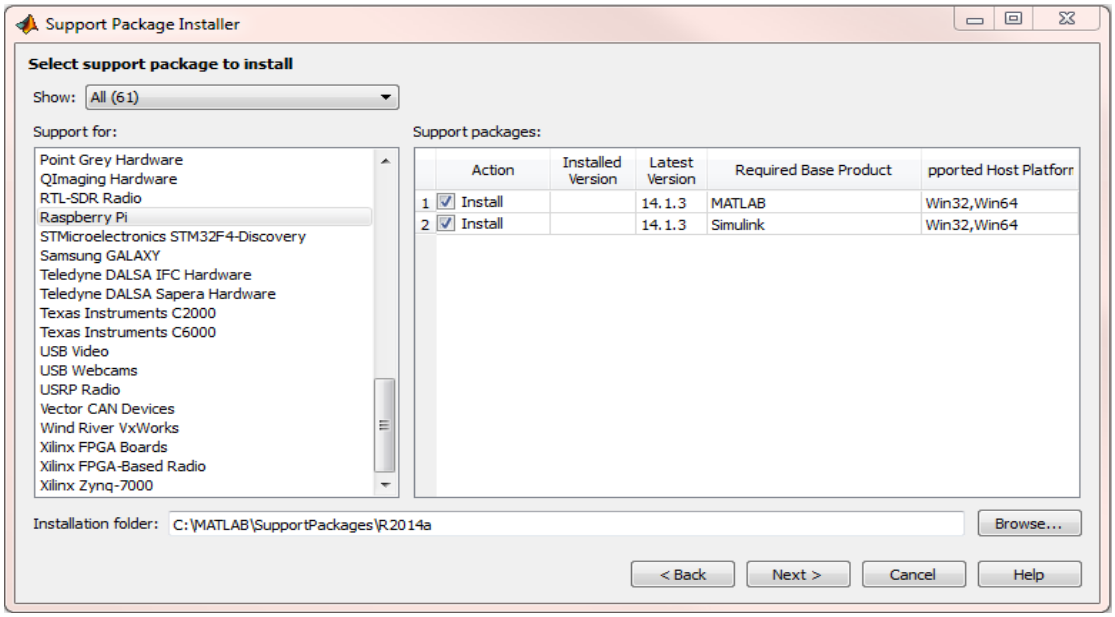
2.1 Installing Support Packages

Στη συνέχεια θα μας εμφανιστεί ένα παράθυρο που μας δίνει την δυνατότητα να επιλέξουμε αν θέλουμε να εγκαταστήσουμε τα πακέτα υποστήριξης μέσω internet ή τοπικά από τον υπολογιστή μας αφού τα κατεβάσουμε. Όλοι οι τρόποι είναι αποδεκτοί. Για την παρούσα εργασία πατάμε Install From Internet.



2.2 Installing Support Packages

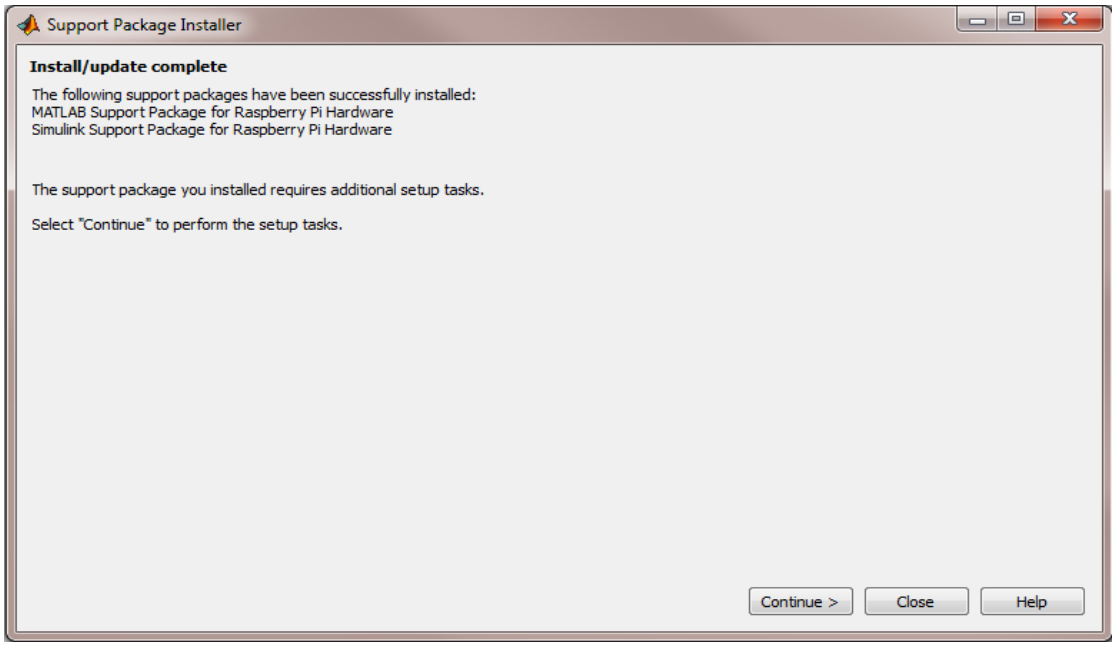
Στο επόμενο παράθυρο μας δίνει μια πλήρη λίστα το Simulink για τα πακέτα υποστήριξης που διαθέτει το κατάστημα της Mathworks. Βρίσκουμε από την λίστα στο παράθυρο αριστερά την επιλογή Raspberry Pi και το επιλέγουμε. Στο δεξί παράθυρο (Support Packages) μας δίνεται η δυνατότητα να επιλέξουμε ποια πακέτα υποστήριξης θέλουμε για το Raspberry Pi. Η πρώτη επιλογή είναι πακέτα αναγνώρισης και εκτέλεσης κώδικα Matlab στο Raspi ενώ η δεύτερη είναι για την αναγνώριση και εκτέλεση σχηματικών μοντέλων του Simulink. Επιλέγουμε και τα δύο και προχωράμε στο επόμενο βήμα (Next).



2.3 Installing Support Packages

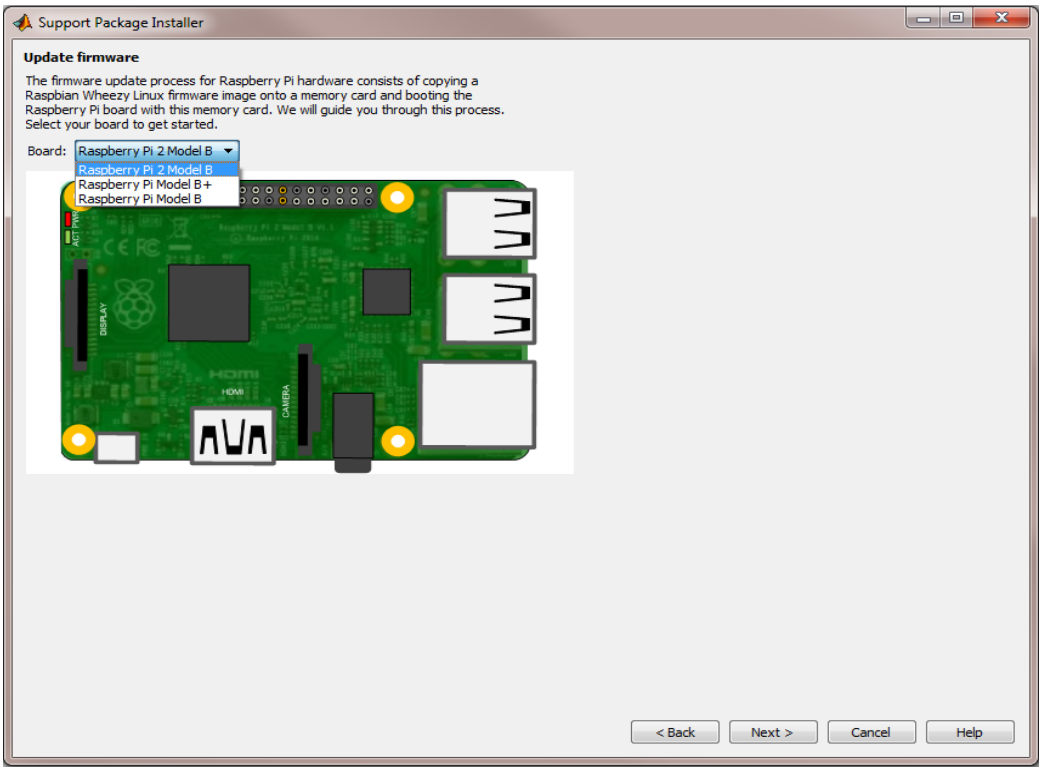
Συνεχίζοντας ένα παράθυρο Pop-Up θα εμφανιστεί που θα μας ζητάει να κάνουμε Log In στον λογαριασμό μας της Mathworks. Κάνουμε είσοδο και συνεχίζουμε (εάν δεν έχουμε λογαριασμό φτιάχνουμε έναν καθώς είναι απαραίτητο για την συνέχεια)

Περιμένουμε μέχρι να εγκατασταθούν τα πακέτα υποστήριξης και να βρεθούμε στην παρακάτω οθόνη.



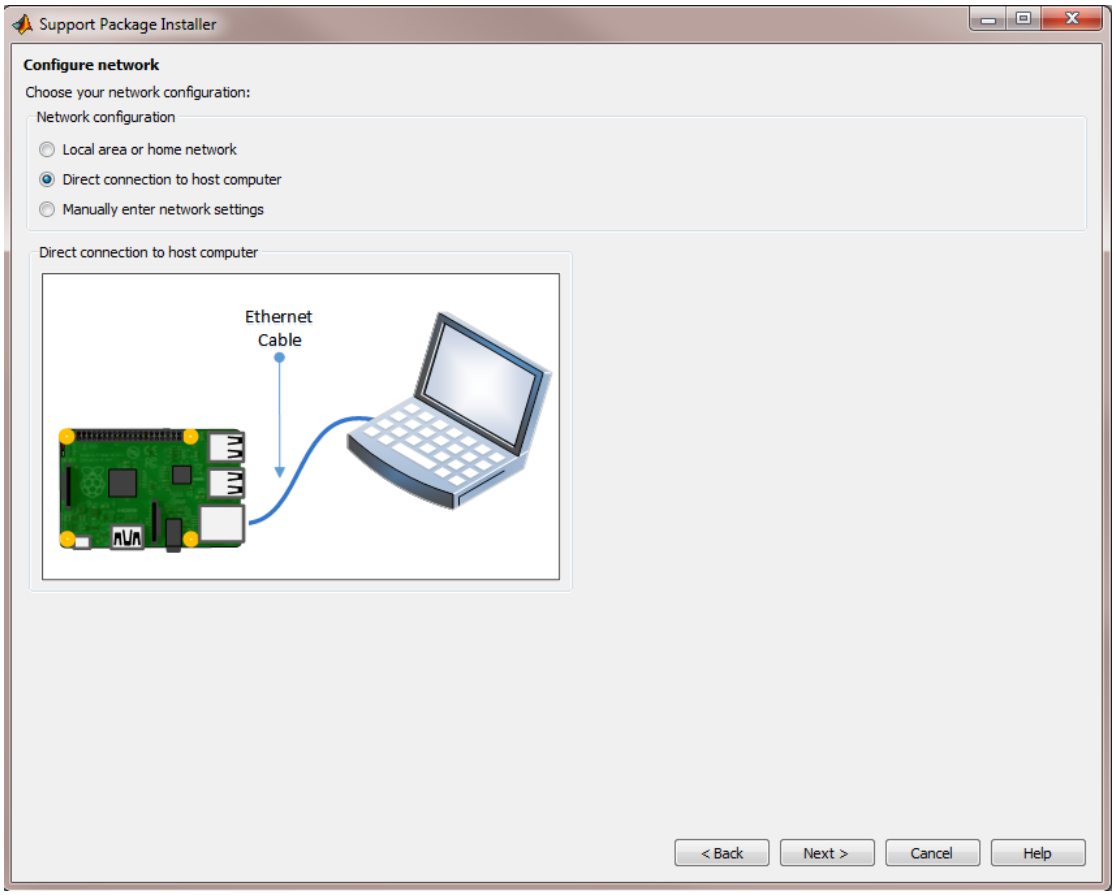
2.4 Installing Support Packages

Στο επόμενο βήμα θα εμφανιστεί μια οθόνη για να κάνουμε αναβάθμιση του Firmware της συσκευής RasPi. Διαλέγουμε το μοντέλο της συσκευής που έχουμε (Raspberry Pi 3 Model B για το παρών project) και συνεχίζουμε.



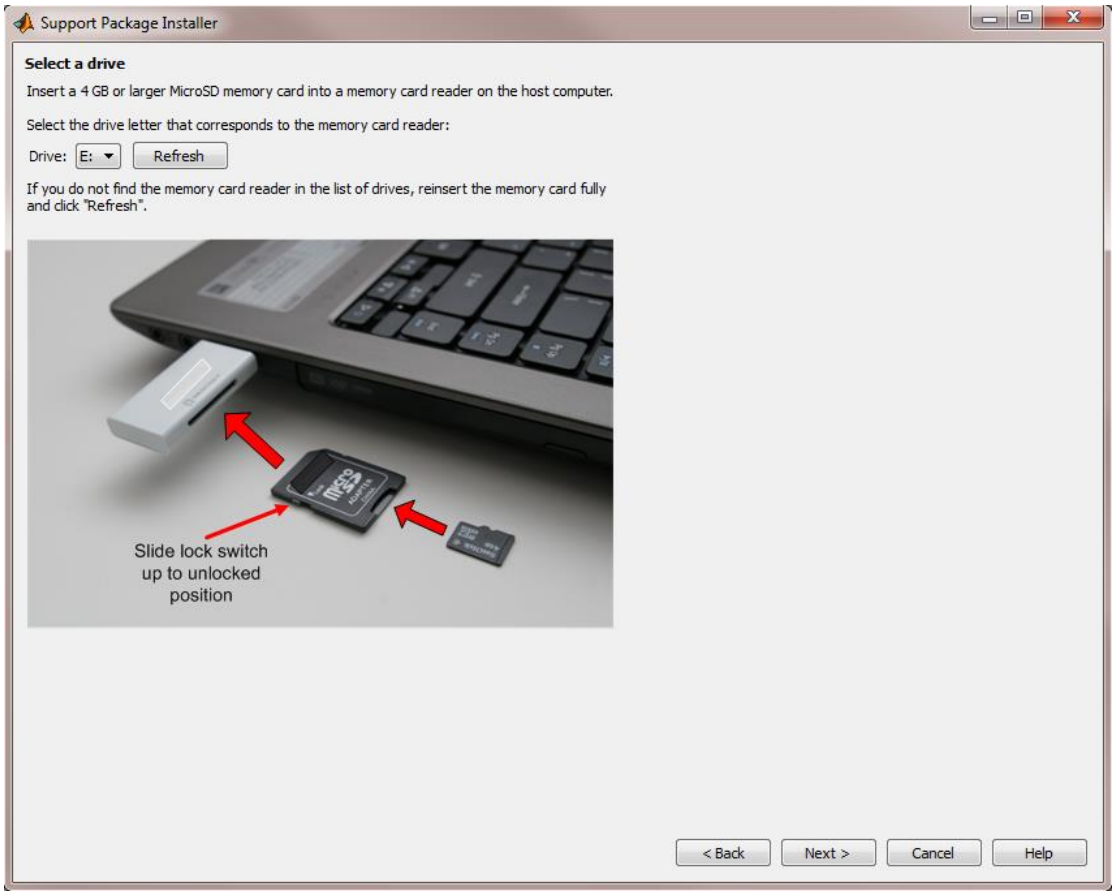
2.5 Installing Support Packages

Στην επόμενη οθόνη θα μας ζητηθεί να συνδέσουμε τοπικά με καλώδιο Ethernet τον υπολογιστή μας με την συσκευή. Το κάνουμε έχοντας επιλεγμένο την επιλογή: Direct connection to host computer.



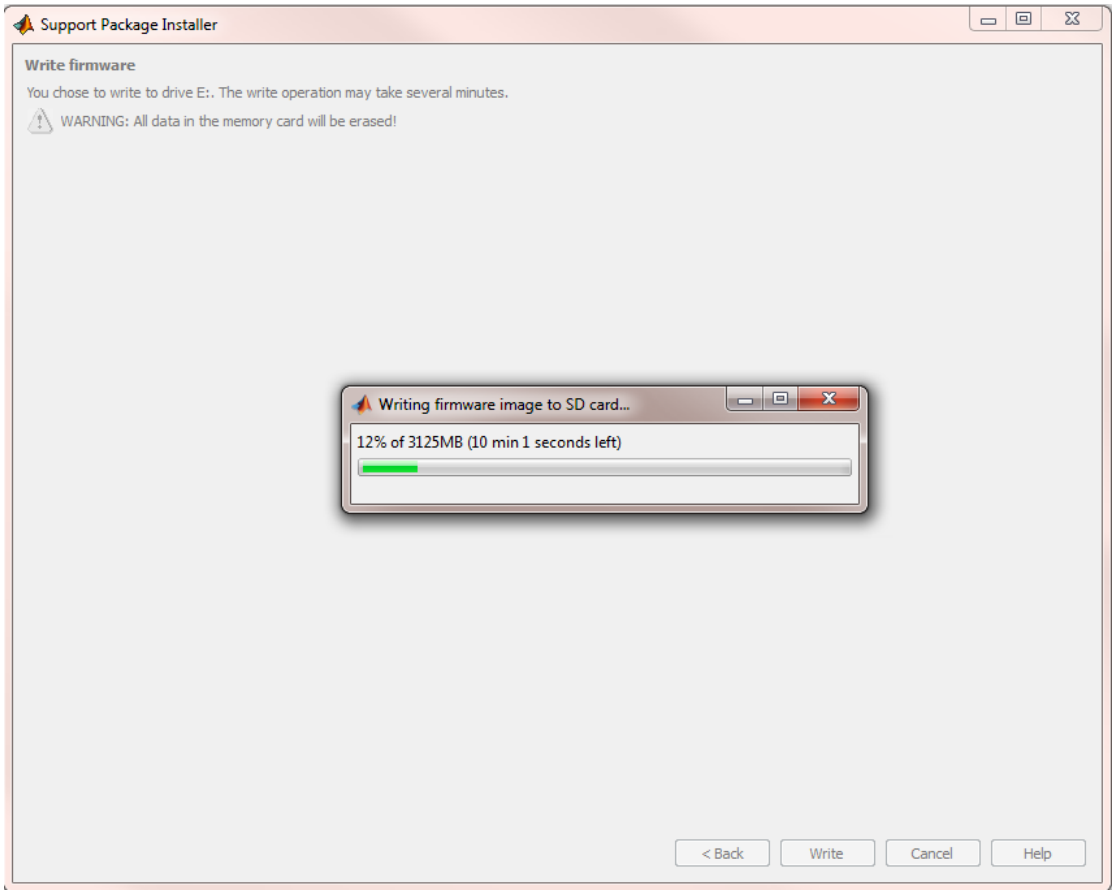
2.6 Installing Raspi Support Packages

Στο επόμενο βήμα θα ζητηθεί από το πρόγραμμα εγκατάστασης να εισάγουμε στον υπολογιστή μας μια SD card για να εγκαταστήσει σε αυτήν τα απαραίτητα προγράμματα οδήγησης για την σωστή λειτουργία του Raspberry Pi. Συνδέουμε μέσω ενός USB adapter την micro SD card στον υπολογιστή μας και πατάμε επόμενο. Εάν δεν αναγνωρίζεται η SD card από το Matlab αλλά αναγνωρίζεται από το λειτουργικό κλείνουμε το Matlab και το ανοίγουμε σαν διαχειριστές και γράφοντας στο Matlab την εντολή `targetupdater`



2.7 Installing Support Packages

Φτάνοντας στο τελικό στάδιο οι ρυθμίσεις μας έχουν ετοιμαστεί και γράφονται στην SD Card διαγράφοντας οτιδήποτε άλλο έχει μέσα. Αν η Sd card είναι καινούρια δεν θα παρουσιαστεί κάποιο πρόβλημα, εάν όμως όχι καλό θα ήταν να γίνουν backup τα αρχεία που περιέχει σε μια άλλη συσκευή καθώς θα χαθούνε κατά τη διάρκεια της εγγραφής.



2.9 Installing Support Packages

Με την ολοκλήρωση της διαδικασίας η συσκευή μας είναι αναβαθμισμένη και έτοιμη για λειτουργία.

Έλεγχος Σύνδεσης

Για να βεβαιωθούμε ότι το Raspberry Pi έχει αναγνωριστεί και έχει συνδεθεί τοπικά με τον υπολογιστή μας ανοίγουμε το Matlab και πληκτρολογούμε την παρακάτω εντολή έχοντας ακόμα συνδεδεμένο το Raspi με το PC μας τοπικά (Ethernet):

```
mypi = raspi
```

το αποτέλεσμα από την εκτέλεση του κώδικα είναι:

```
>> mypi = raspi

mypi =

  raspi with properties:

         DeviceAddress: 169.254.0.2
                  Port: 18730
             BoardName: Raspberry Pi 3 Model B
         AvailableLEDs: {'led0'}
  AvailableDigitalPins:
[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24
,25,26,27]
  AvailableSPIChannels: {}
     AvailableI2CBuses: {'i2c-1'}
      AvailableWebcams: {'mmal service 16.1
(platform:bcm2835-v4l2):'}
           I2CBusSpeed: 0

  Supported peripherals
```

Από το αποτέλεσμα της εκτέλεσης του κώδικα βλέπουμε ότι η συσκευή μας έχει συνδεθεί κανονικά με τον υπολογιστή μας. Στο DeviceAddress φαίνεται η IP που έχει δοθεί στην συσκευή. Την αποθηκεύουμε κάπου διότι θα μας χρειαστεί.

Στην συνέχεια θα αποκτήσουμε πρόσβαση στο SSH shell server της συσκευής.

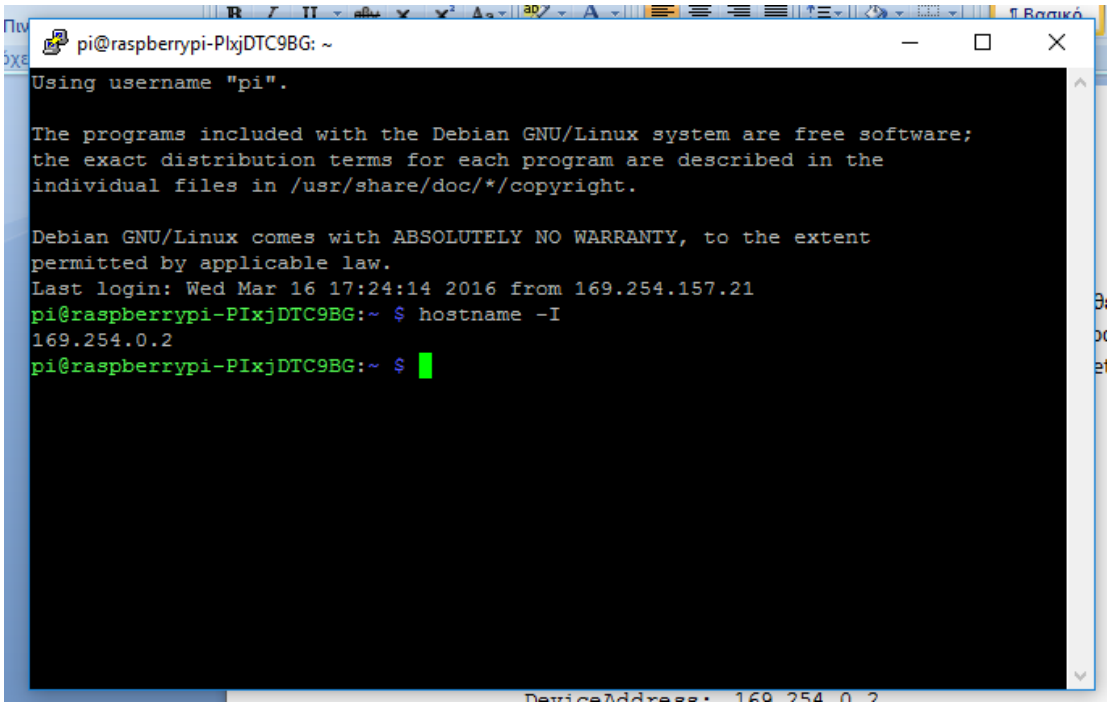
1. ***Σημειώνουμε την Ip Address της συσκευής. Έπειτα ανοίγουμε το Matlab και πατάμε την εντολή:***

h = raspberrypi (‘169.254.0.2’)

h.openShell()

με αυτό τον τρόπο δίνουμε στην συσκευή την IP διεύθυνση σαν παράμετρο και της την περνάμε αναγκάζοντας έτσι με την εντολή h.openShell() να επιβάλουμε σύνδεση στον Ssh server της συσκευής.

2. ***Μετά από αυτό το βήμα μας ανοίγει το το τερματικό του:***

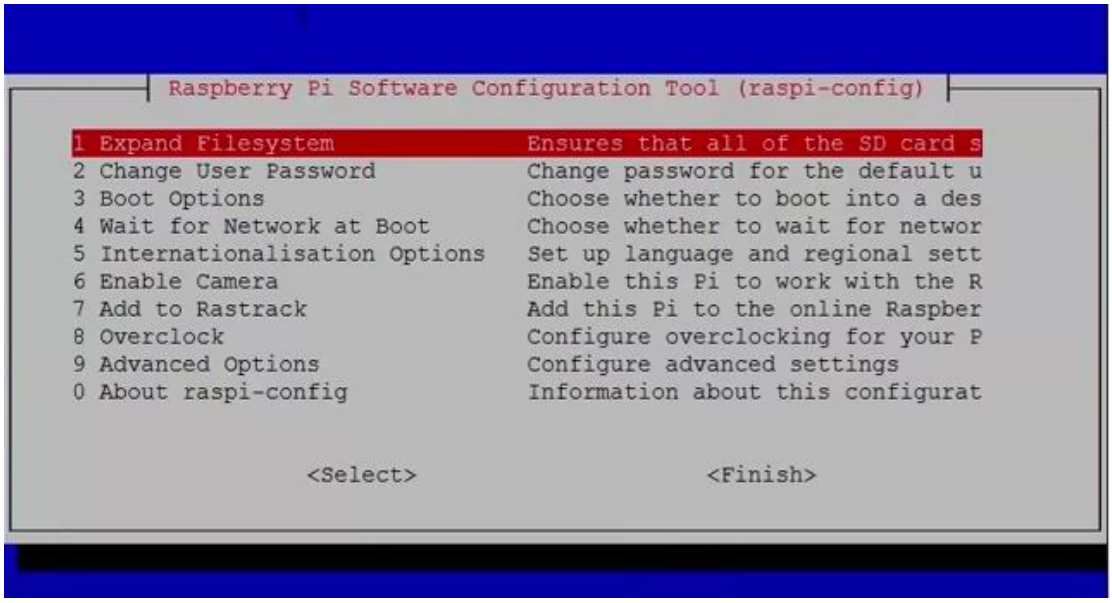


3.1 Opening raspi ssh

Από Deafult το username για να συνδεθούμε στον τερματικό του Raspi είναι Pi και ο κωδικός Raspberry.

3. ***Πλέον έχουμε αποκτήσει πλήρη πρόσβαση στην συσκευή και μπορούμε να την διαμορφώσουμε ανάλογα.***

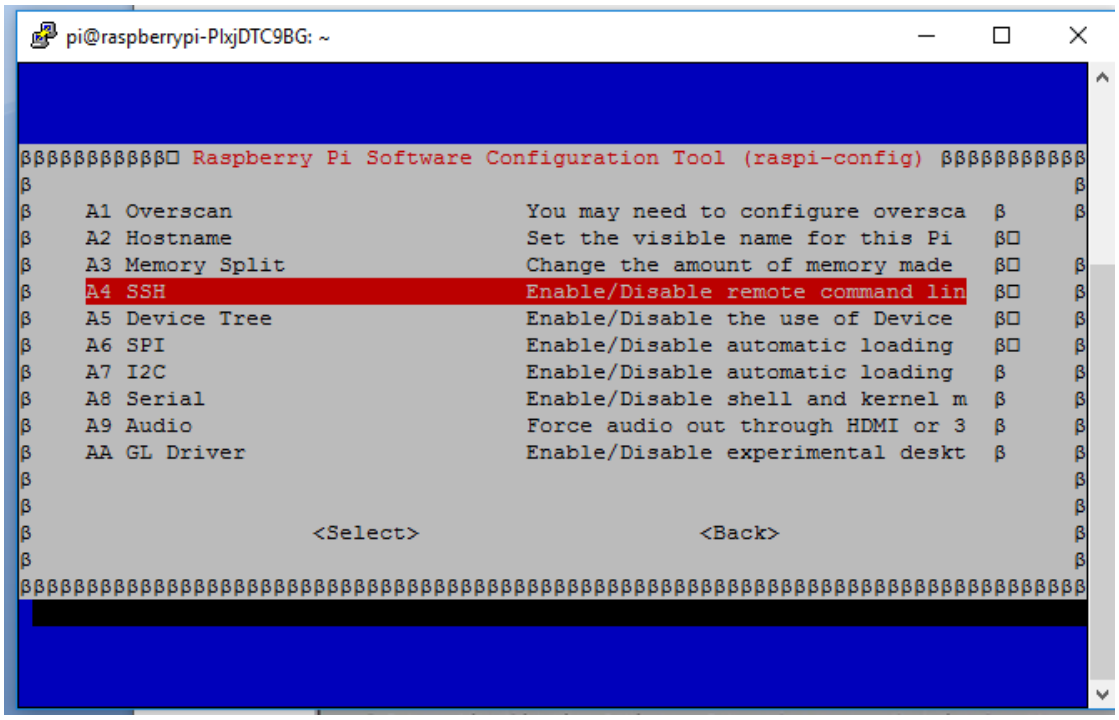
Για να μπούμε στο BIOS και να διαμορφώσουμε περαιτέρω την συσκευή πατάμε στον τερματικό την εντολή: sudo raspi-config. Το Sudo είναι απαραίτητο γιατί μας δίνει τα δικαιώματα Power User στην συσκευή.



3.2 ssh config

Πηγαίνουμε στην επιλογή 9.Advanced Options και επιλέγουμε το SSH SERVER.

Σιγουρεύουμε ότι είναι στο Enable:



3.3 ssh config

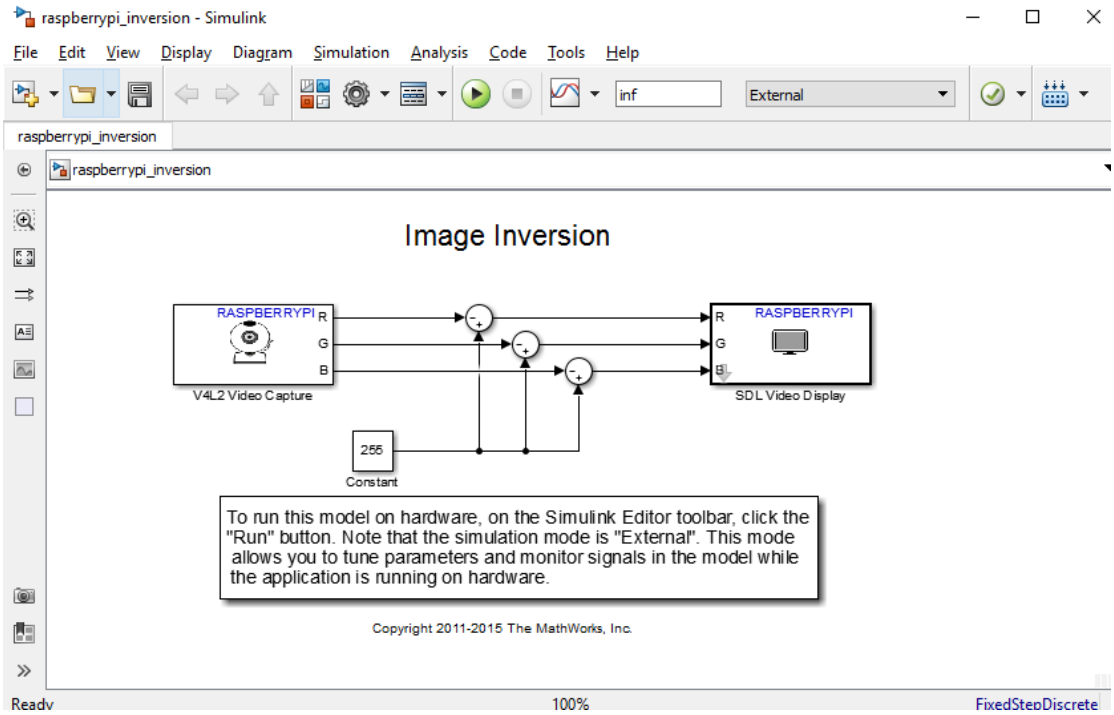
ΜΕΡΟΣ ΠΡΩΤΟ:Εκτελώντας ένα απλό Image Inversion στο Simulink

Εφόσον καταφέραμε να συνδεθούμε επιτυχώς με την συσκευή μας από τον τοπικό μας υπολογιστή και αποκτήσαμε πρόσβαση είναι καιρός να δοκιμάσουμε ένα απλό Image Inversion για να ελέγξουμε ότι η κάμερα του Raspberry Pi είναι αναγνωρίσιμη και λειτουργική καθώς και ότι το Raspberry μας δέχεται και μπορεί να εκτελέσει μοντέλα που θα του εισαχθούν από το Simulink.

Δημιουργούμε στην επιφάνεια εργασίας μας έναν Working Folder για το Matlab, ανοίγουμε το Matlab και επιλέγουμε τον φάκελο αυτόν σαν προεπιλεγμένο Working Folder με την εντολή: cd και το path του φακέλου

cd C:\Users\big_r\Desktop\Work

Στη συνέχεια εκτελούμε την εντολή raspberrypi_inversion και ανοίγει το ακόλουθο παράθυρο:

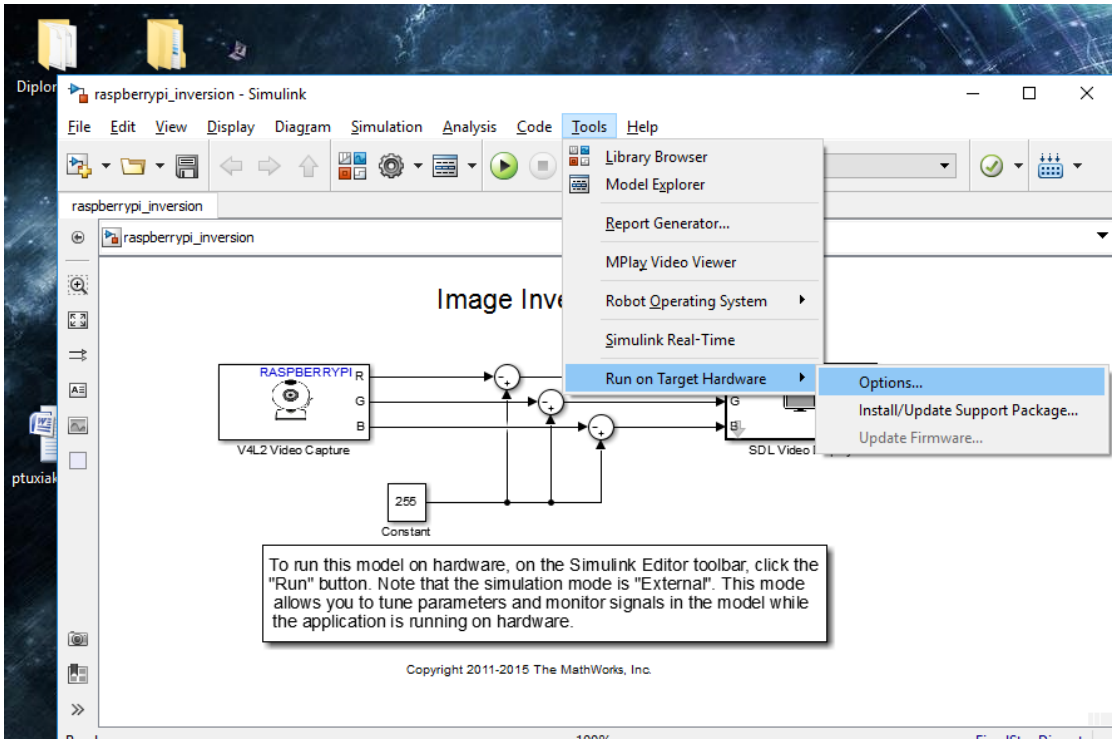


4.1 Image Inversion

Όπως βλέπουμε πρόκειται για ένα πολύ απλό σχηματικό του Simulink που παίρνει σαν παραμέτρους 3 εισόδους Red, Green and Blue και τις αναστρέφει και τις βγάζει σαν έξοδο στην κάμερα.

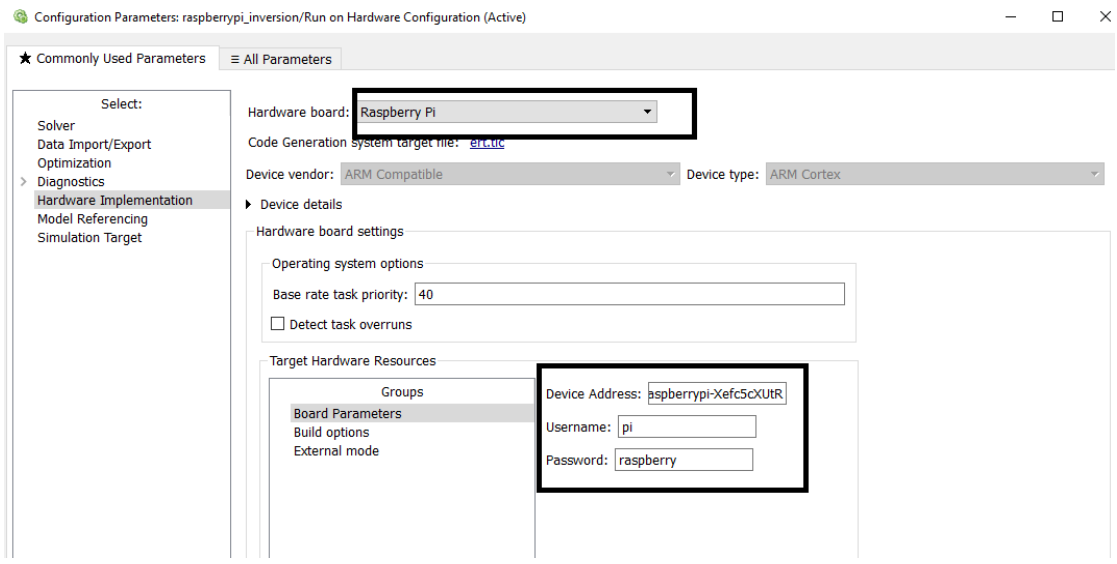
Πρόκειται για μια τεχνική επεξεργασίας εικόνας, όπου οι φωτεινές περιοχές μετατρέπονται σε σκοτεινές, και οι σκοτεινές περιοχές μετατρέπονται σε φωτεινές. Με άλλα λόγια, μετά την αναστροφή της εικόνας το μαύρο γίνεται άσπρο και το άσπρο γίνεται μαύρο. Μια ανεστραμμένη ασπρόμαυρη εικόνα (Inverted Image) μπορεί να θεωρηθεί ως ένα ψηφιακό αρνητικό της αρχικής εικόνας.

Πριν τρέξουμε το σχηματικό μας πηγαίνουμε στην καρτέλα Tools>Run on Target Hardware>Options

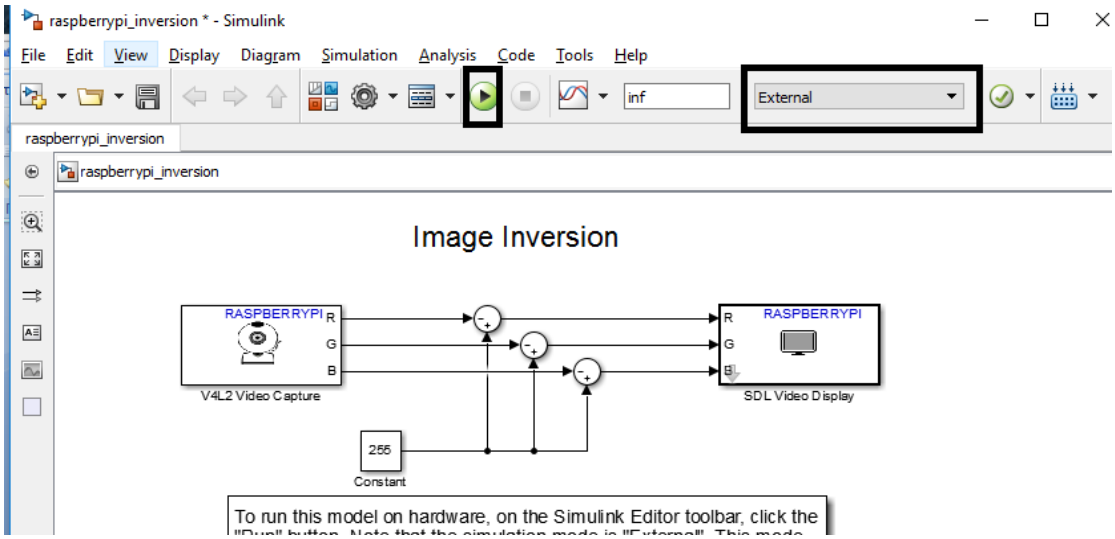


4.2 Image Inversion

Και επιβεβαιώνουμε ότι όλα τα παιδία είναι σωστά:

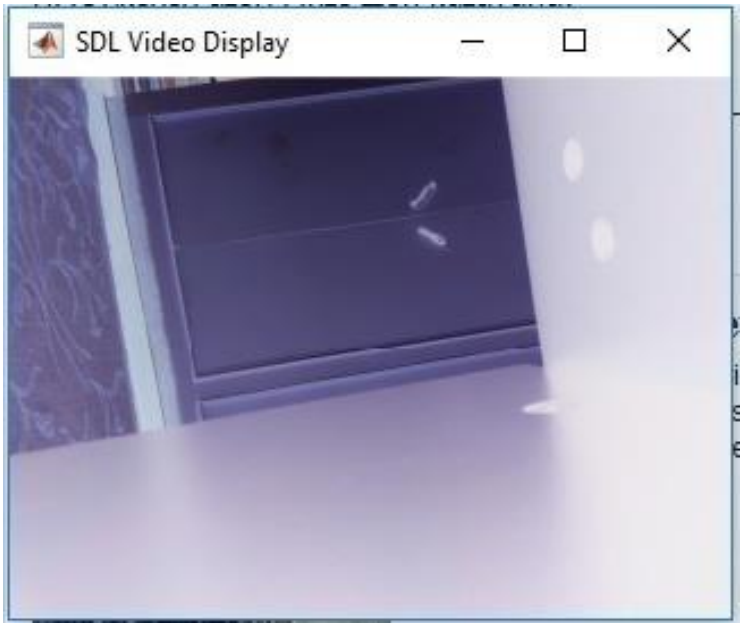


Επίσης φροντίζουμε να έχουμε επιλεγμένη την επιλογή External και τρέχουμε την προσομοίωση:



4.3 Image Inversion

Το τελικό αποτέλεσμα που παίρνουμε είναι η ανεστραμμένη εικόνα χρωμάτων της κάμερας:



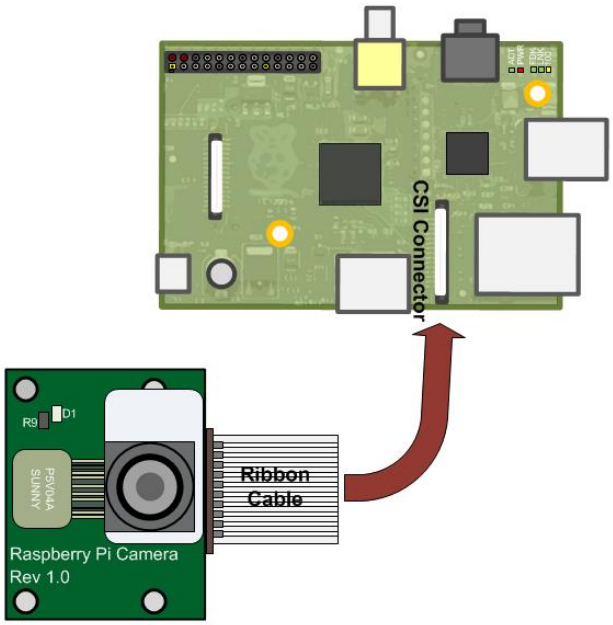
4.4 Results of Image Inversion

Το RasPi μας είναι πλέον λειτουργικό και μπορεί να τρέξει οποιοδήποτε μοντέλο του Simulink του δώσουμε.

Δουλεύοντας με την κάμερα του Raspberry Pi και την συσκευή

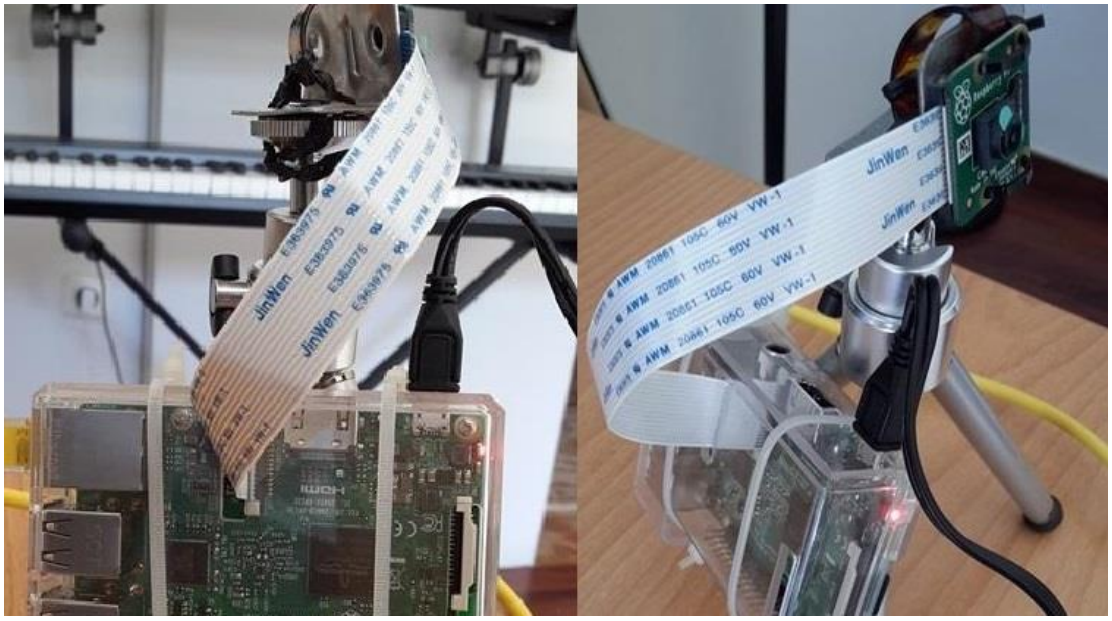
Συνδέοντας την κάμερα

Αρχικά συνδέουμε την Rasapberry Camera στο board του Raspi με την καλώδιο-ταινία που παρέχεται με την αγορά της κάμερας όπως δείχνετε στο σχηματικό παρακάτω:



5.1 Raspi Cam

Το τελικό αποτέλεσμα πρέπει της συνδεσμολογίας μας μεταξύ κάμερας και το raspberry πρέπει να είναι αυτό:



5.2 Raspi Cam

Εφόσον εκτελέσαμε σωστά την συνδεσμολογία πρέπει να ελέγξουμε αν η κάμερα είναι ορατή από την συσκευή και λειτουργεί κανονικά. Για να το κάνουμε αυτό πηγαίνουμε στο Matlab και πληκτρολογούμε την εντολή

>>mypi=raspi

Με αυτή την εντολή δημιουργούμε εκ νέου σύνδεση της συσκευής με τον υπολογιστή μας σε περίπτωση που δεν το έχουμε κάνει ήδη. Εάν έχουμε ήδη μια υπάρχουσα σύνδεση μέσω αυτής της εντολής από πριν μπορούμε να το παραλείψουμε.

Στη συνέχεια για να δημιουργήσουμε μια σύνδεση με την κάμερα περνώντας κάποιες προαιρετικές παραμέτρους (σε αυτή την περίπτωση την ανάλυση) πληκτρολογούμε την εντολή:

>> cam = cameraboard (mypi, 'Resolution', '1280x720')

Το αποτέλεσμα που βγάζει ο κώδικας είναι το εξής:

```
>> cam=cameraboard(mypi,'Resolution', '1280x720')

cam =

  cameraboard with properties:

                    Name: Camera Board
              Resolution: '1280x720'      (View available resolutions)
                 Quality: 10              (1 to 100)
                Rotation: 0               (0, 90, 180 or 270)
          HorizontalFlip: 0
            VerticalFlip: 0
               FrameRate: 30              (2 to 90)
               Recording: 0

   Picture settings
              Brightness: 50              (0 to 100)
                Contrast: 0               (-100 to 100)
              Saturation: 0               (-100 to 100)
               Sharpness: 0               (-100 to 100)

   Exposure and AWB
            ExposureMode: 'auto'          (View available exposure modes)
     ExposureCompensation: 0              (-10 to 10)
                AWBMode: 'auto'           (View available AWB modes)
            MeteringMode: 'average'       (View available metering modes)

   Effects
             ImageEffect: 'none'          (View available image effects)
       VideoStabilization: 'off'
                     ROI: [0.00 0.00 1.00 1.00] (0.0 to 1.0 [top, left, width, height])
```

5.3 Raspi Cam Config

Βλέπουμε λοιπόν ότι Matlab δημιούργησε το αντικείμενο cameraboard.object και μας δείχνει στην οθόνη όλες τις παραμέτρους διαθέσιμες που μπορούμε να αλλάξουμε από το Matlab. Αν για παράδειγμα θέλουμε να αλλάξουμε την φωτεινότητα της κάμερας πληκτρολογούμε την εντολή cam.Brightness=70

```
>> cam.Brightness=70

cam =

  cameraboard with properties:

                    Name: Camera Board
              Resolution: '1280x720'      (View available resolutions)
                 Quality: 10              (1 to 100)
                Rotation: 0               (0, 90, 180 or 270)
          HorizontalFlip: 0
            VerticalFlip: 0
               FrameRate: 30              (2 to 90)
               Recording: 0

   Picture settings
              Brightness: 70              (0 to 100)
                Contrast: 0               (-100 to 100)
              Saturation: 0               (-100 to 100)
               Sharpness: 0               (-100 to 100)

   Exposure and AWB
            ExposureMode: 'auto'          (View available exposure modes)
     ExposureCompensation: 0              (-10 to 10)
                AWBMode: 'auto'           (View available AWB modes)
            MeteringMode: 'average'       (View available metering modes)

   Effects
             ImageEffect: 'none'          (View available image effects)
       VideoStabilization: 'off'
                     ROI: [0.00 0.00 1.00 1.00] (0.0 to 1.0 [top, left, width, height])
```

5.4 Raspi Cam Config

Βλέπουμε λοιπόν ότι όπως και στην ανάλυση έτσι και στην φωτεινότητα το Matlab δέχτηκε την παράμετρο που του δώσαμε και την πέρασε στις ιδιότητες της κάμερας σαν την νέα παράμετρο.

Με την ίδια λογική εργαζόμαστε για να αλλάξουμε την έκθεση φωτός σε νυχτερινή λήψη με την εντολή:

>>cam.ExposureMode = ’night’ κ.ο.κ.

Τραβώντας στιγμιότυπα με την κάμερα μέσω Matlab

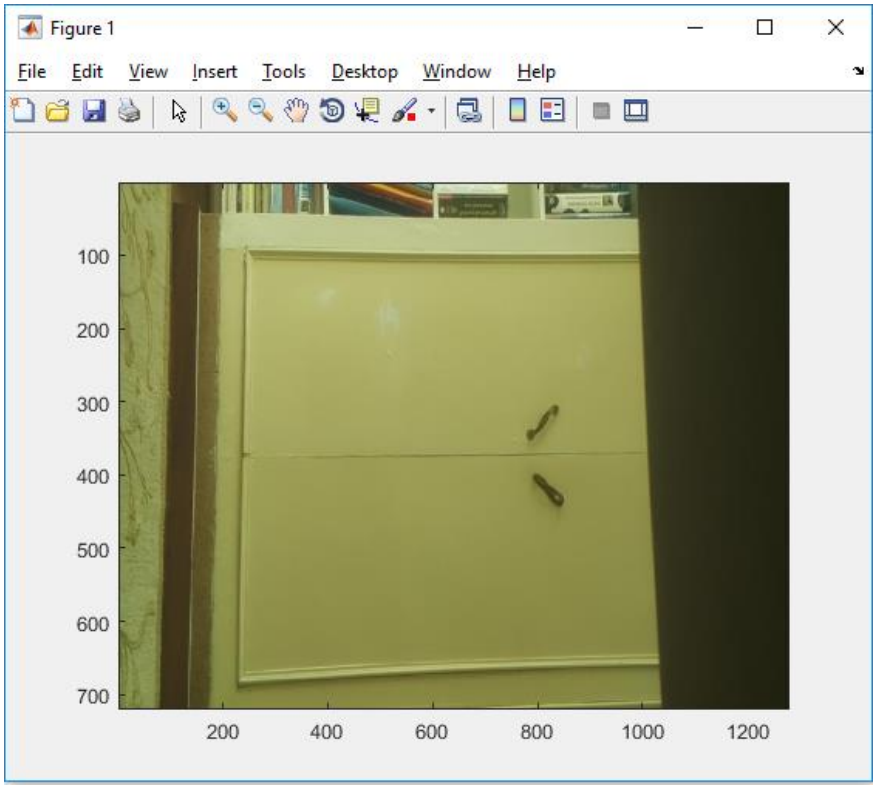
Για να αποτυπώσουμε ένα στιγμιότυπο στην κάμερα πληκτρολογούμε την εντολή

>>img=snapshot(cam);

Το στιγμιότυπο έχει πλέον αποθηκευτεί. Για να δούμε την φωτογραφία που τραβήξαμε πατάμε την εντολή

>>imagesc(img)

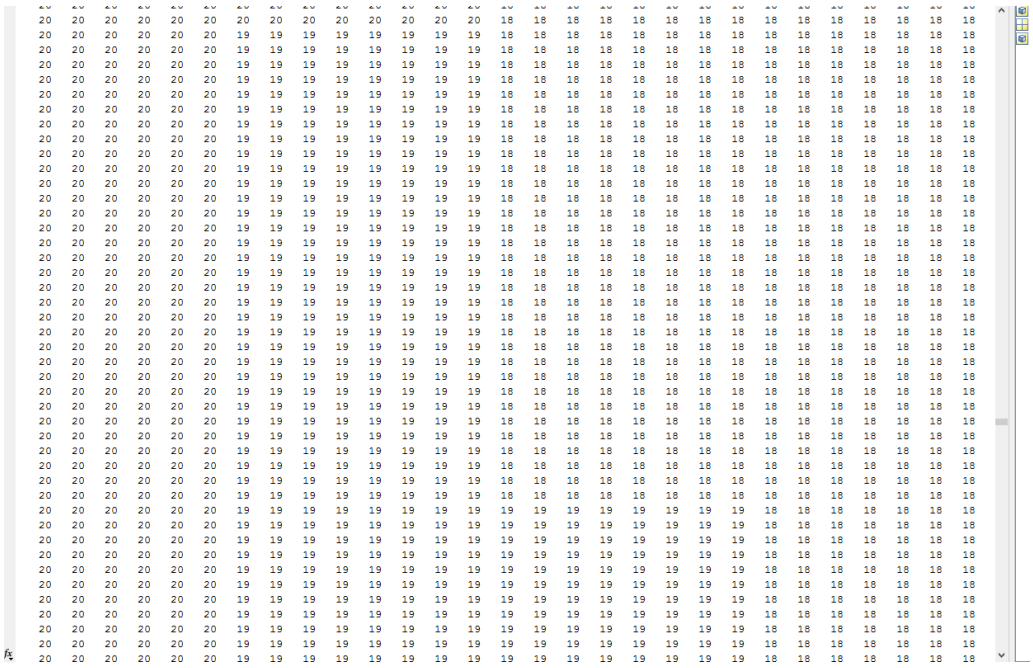
Ένα νέο figure θα ανοίξει και θα μας εμφανίσει την εικόνα που τραβήξαμε



5.4 Raspi Cam Snapshot

Αξιοσημείωτο είναι να αναφέρουμε ότι αν στην εντολή img=snapshot(cam);

παραλείψω το ερωτηματικό στο τέλος το Matlab θα μου εμφανίσει τον πίνακα με τις τιμές των pixels της εικόνας:



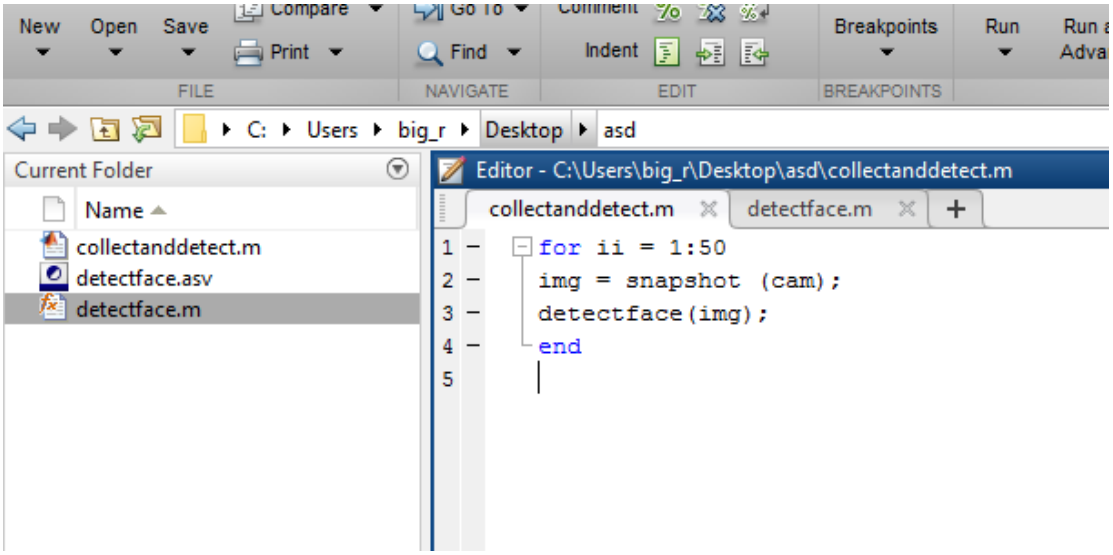
5.4 Raspi Cam Image Pixel Table

Συλλέγοντας πολλαπλά στιγμιότυπα από την κάμερα

Για το επόμενο παράδειγμα δημιουργούμε ένα νέο script file στο Matlab και το ονομάζουμε collectanddetect.m

Αυτό που επιδιώκουμε είναι ένας βρόγχος ο οποίος θα συγκεντρώνει εικόνες από την κάμερα 50 φορές. Με προγραμματιστική λογική σε C ο κώδικας θα είναι ο εξής:

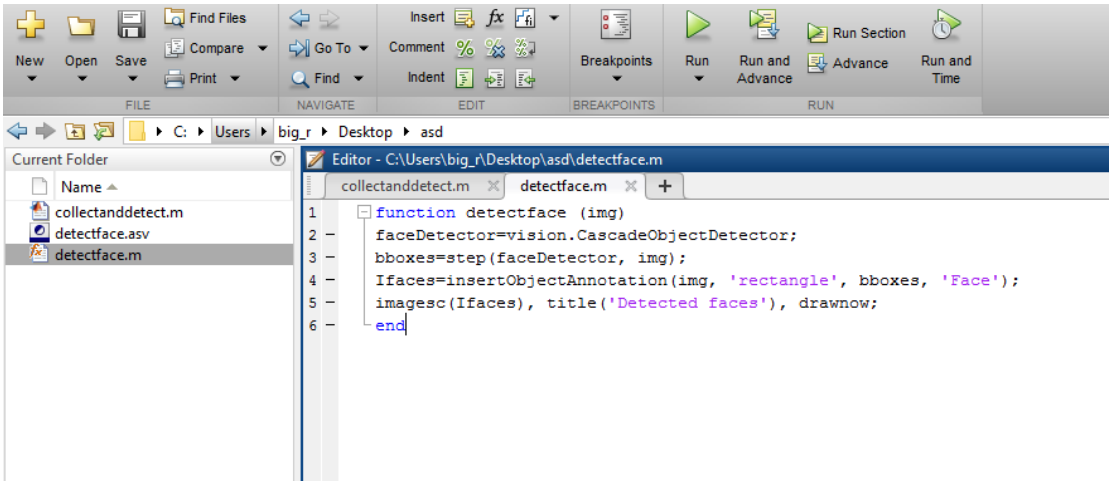
```
for ii = 1:50 % Απο 1 έως 50 φορές επανέλαβε.Το ii είναι ο μετρητής μας

    img = snapshot (cam); % η συνάρτηση που χρησιμοποιήσαμε προηγουμένως για την λήψη φωτογραφιών

    detectface(img); % κλήση συνάρτηση που δημιουργεί ένα κουτί γύρο από το πρόσωπο αν υπάρχει

end
```



6.1 Raspi Multi frame capture

Ανάλυση της detectface(img);

Δημιουργώντας ένας ακόμα Script file ονομάζοντας το detectface.m ανοίγουμε τον editor. Φτιάχνουμε την συνάρτηση detectface όπως φαίνεται παρακάτω:



6.2 Raspi Multi frame capture

```
function detectface (img)%δημιουργία της συνάρτησης detectface

      faceDetector=vision.CascadeObjectDetector; %κάλεσμα της
      συνάρτησης vision.CascadeObjectDetector από το System
      Vision ToolBox του Matlab.

      bboxes=step(faceDetector, img); %δημιουργία του κουτιού
      γύρω από το πρόσωπο της εικόνας

      Ifaces=insertObjectAnnotation(img, 'rectangle', bboxes,
      'Face'); %εισαγωγή του παραλληλόγραμμου με ταμπέλα στην
      τοποθεσία που του έχει υποδειχτεί πριν από την bboxes

      imagesc(Ifaces), title('Detected faces'), drawnow;
      %συνεχής αναβάθμιση του figure για τυχών αλλαγές στην
      θέση του αντικειμένου
end
```

Vision.CascadeObjectDetector: Ανιχνεύει τα αντικείμενα στις εικόνες σέρνοντας ένας παράθυρο πάνω από την εικόνα. Ο ανιχνευτής χρησιμοποιεί έπειτα ένα ταξινομητή αλληλουχίας για να αποφασίσει αν το παράθυρο περιέχει το αντικείμενο ενδιαφέροντος ή όχι. Το μέγεθος του παραθύρου τροποποιείτε σε διαφορετικά μεγέθη και αναλογίες αλλά η αναλογία απεικόνισης παραμένει σταθερή. Ο ανιχνευτής είναι ευαίσθητος όσον αφορά την διάσταση στον χώρο για αυτό το λόγο η αναλογία απεικόνισης αλλάζει στα περισσότερα 3D αντικείμενα.

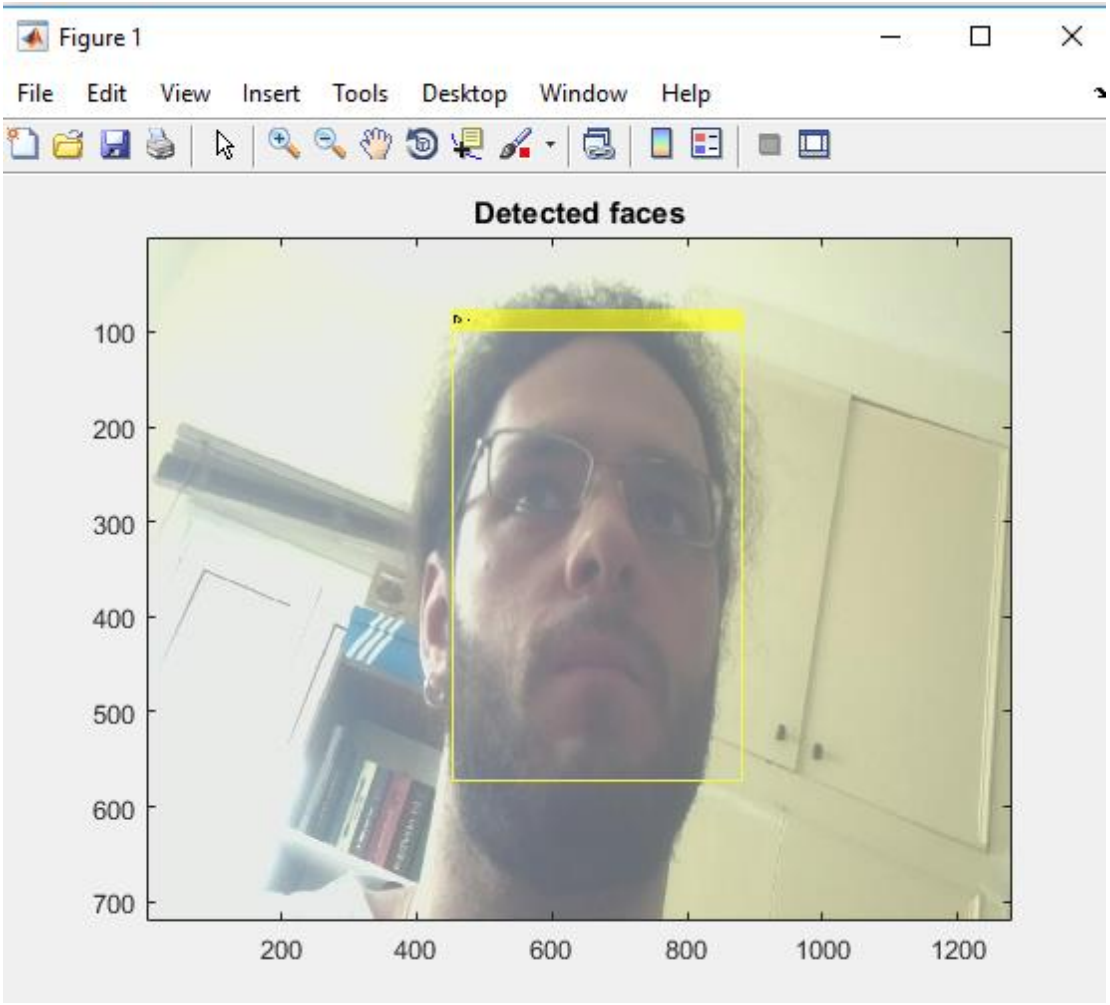
Drawnow: Αναβαθμίζει τα figures και επεξεργάζεται οποιοδήποτε κάλεσμα μετατροπής. Εν τέλει ανανεώνει συνεχώς το αποτέλεσμα που βλέπουμε εμείς στην οθόνη όταν υπάρξει μεταβολή στο figure.

Step: Υπολογίζει την βηματική απόκριση ενός δυναμικού συστήματος. Στο συγκεκριμένο παράδειγμα δημιουργεί το κουτί γύρω από το πρόσωπο και δείχνει τα αποτελέσματα

insertObjectAnnotation: Εισάγει παραλληλόγραμμα και ταμπέλες στην τοποθεσία που υποδεικνύει ο πίνακας θέσεως

Π.χ `insertObjectAnnotation(I,'rectangle',position,label)`

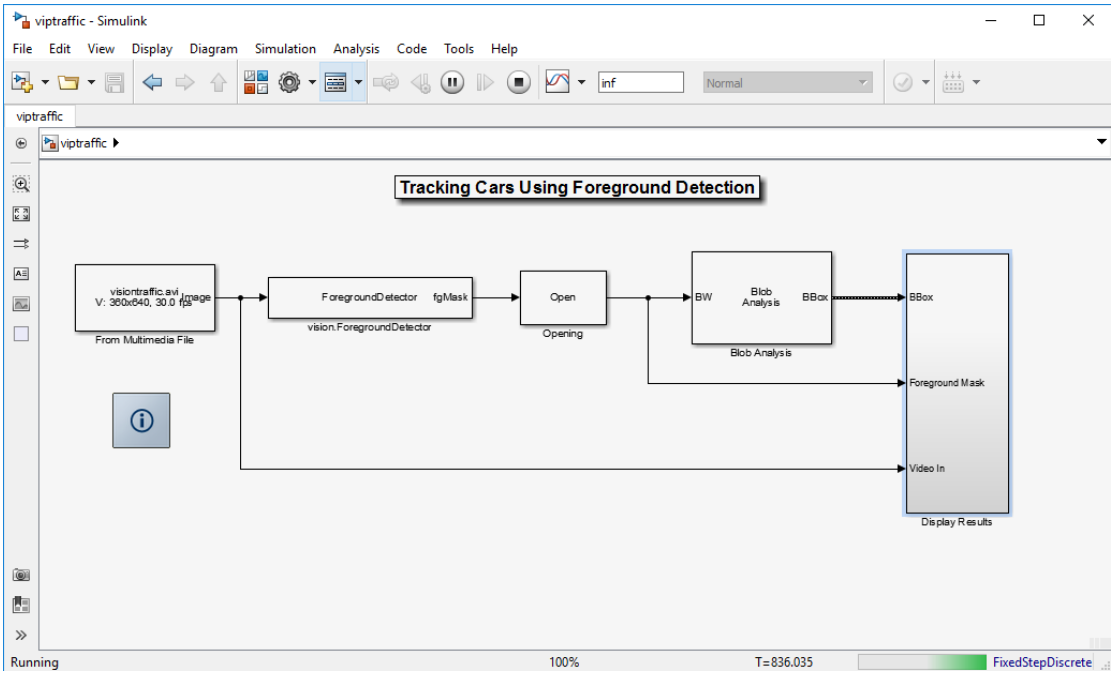
Το τελικό αποτέλεσμα :



6.2 Raspi Multi frame capture with frame around the point of interest

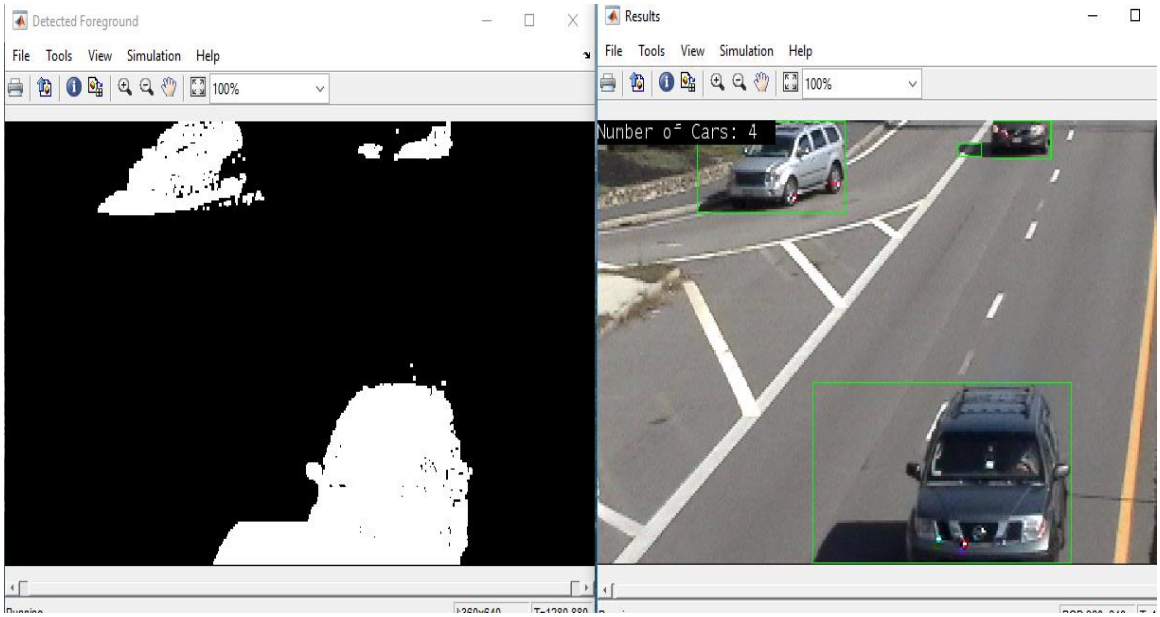
Καταμέτρηση αυτοκινήτων με ανίχνευση Foreground Detection

Ένα απλό παράδειγμα το οποίο αντιστοιχεί στο αντικείμενο της μελέτης μας είναι ένα έτοιμο μοντέλο που μας παρέχει το simulink το οποίο κατά την εκτέλεση χρησιμοποιεί foreground detection για την ανίχνευση οχημάτων σε έναν δρόμο και δείχνει στην οθόνη τον αριθμό τον αμαξιών που βρίσκονται στο οπτικό πεδίο της κάμερας κάθε φορά. Για να ανοίξουμε το μοντέλο αυτό πατάμε στο Matlab >>viptraffic. Αυτό μας ανοίγει το παρακάτω σχηματικό στο Simulink:



7.1 Simulink Foreground Model

Βλέπουμε ότι το σχηματικός το παράδειγμά μας παίρνει είσοδο από ένα multimedia αρχείο βίντεο το οποίοι ήδη έχουν καταγράψει μερικά αμάξια σε έναν δρόμο κυκλοφορίας για χάρη του παραδείγματος, από ένα μοντέλο Foreground Detection, μία ανάλυση Blob και το τελικό Output που θα εμφανιστεί στην οθόνη. Εκτελώντας το μοντέλο σε debugging mode παίρνουμε το εξής αποτέλεσμα:



7.1 Simulink Foreground Model Results

Επεξήγηση και ανάλυση της μεθόδου καταμέτρησης με Foreground Detection

Η ανίχνευση παρασκηνίου για να πραγματοποιηθεί πρέπει πρώτα να υπολογιστούν τα pixels του παρασκηνίου στο video που καταγράφει μία στατική κάμερα σε έναν δρόμο κυκλοφορίας. Το Vision ForegroundDetector εκτελεί ακριβώς αυτή την διαδικασία δηλαδή υπολογίζει το παρασκήνιο χρησιμοποιώντας (Γκαουσιανό Μοντέλο Μίξης) Gaussian Mixture Models και παράγει μία μάσκα παρασκηνίου όπου έχει κάνει highlight τα τα αντικείμενα ενδιαφέροντος στο παρασκήνιο δηλαδή στην δική μας περίπτωση τα κινούμενα αμάξια.

Στη συνέχεια η μάσκα αυτή εισάγεται και υπόκειται σε επεξεργασία από την Blob Analysis block που παράγει τα πράσινα κουτάκια που βλέπουμε στην εικόνα γύρω από τα αμάξια τα οποία παραμένουν στο αντικείμενο-αμάξι μέχρι αυτό να εξαφανιστεί από το οπτικό πεδίο της κάμερας. Τέλος ανάλογα με το πόσα bounding boxes έχουμε στο οπτικό μας πεδίο, ένας καταμετρητής αυξάνεται ή μειώνεται ανάλογα.

Gaussian Mixture Model (GMM) - Επεξήγηση και ανάλυση

Όπως προαναφέραμε η μέθοδος Foreground Detection χρησιμοποιεί Gaussian Mixture Models για να τονίσει με highlight τα αμάξια που περνάνε.

Το Gaussian mixture model λοιπόν είναι ένα πιθανοθεωρητικό μοντέλο που αντιπροσωπεύει την ύπαρξη υποπλυθησμών σε ένα μεγαλύτερο και γενικότερο σύνολο. Σημαντικό είναι ότι δεν απαιτείτε το παρατηρούμενο σύνολο δεδομένων να προσδιορίζει τον υποπλυθησμο στον οποίο ανήκει μια μεμονωμένη παρατήρηση. Αυτή η μέθοδος είναι ιδιαίτερα χρήσιμη σε εφαρμογές για machine-learning.

Ανάλυση αλγορίθμου του Foreground Detection με GMM Machine Learning

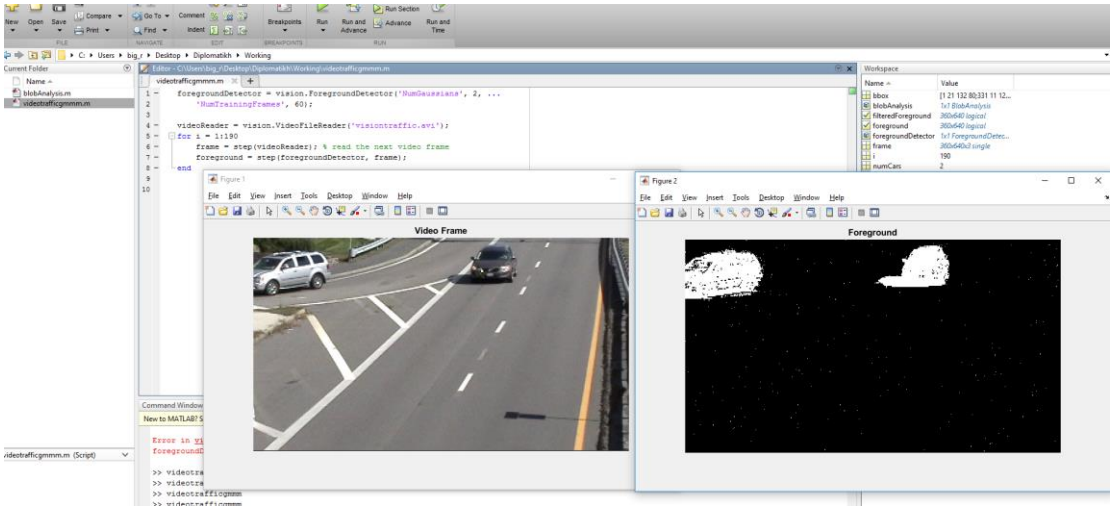
Όπως είδαμε στο παράδειγμα πίσω το μοντέλο μας στο Simulink χρησιμοποιεί Foreground Detection για να ανιχνεύσει τα αυτοκίνητα στο παρασκήνιο. Το τελικό αποτέλεσμα του κώδικα που βρίσκεται πίσω από αυτό είναι πολύπλοκο για αυτό θα το χωρίσουμε σε μικρά μέρη.

Αντί να επεξεργαστεί ο αλγόριθμος ολόκληρο το βίντεο θα αρχίσουμε με το να παίρνουμε ένα αρχικό βίντεο frame που τα κινούμενα αμάξια είναι διαχωρισμένα στο παρασκήνιο. Για να εφαρμόσει Gaussian mixture model ο Foreground Detector και να ανιχνεύσει τα αμάξια απαιτεί έναν συγκεκριμένο αριθμό frames του βίντεο. Για το συγκεκριμένο παράδειγμα θα εφαρμόσουμε 60 frames για 2 Gaussian μοντέλα.

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 2,
'NumTrainingFrames', 60);  %εισαγωγή τιμών για gmm και video frame
videoReader = vision.VideoFileReader('visiontraffic.avi'); %ανάγνωση από αρχείο
for i = 1:190 % στιγμή λήψη στιγμιότυπου
   frame = step(videoReader); % ανάγνωση του επόμενου video frame
   foreground = step(foregroundDetector, frame); %μετάβαση στο επόμενο αντικείμενο για Fgd
end
```

Μετά από αυτό ο ανιχνευτής αρχίζει να δίνει πιο αξιόπιστα αποτελέσματα εξόδου. Αν εκτυπώσουμε τα αποτελέσματα από το βίντεο με τον κώδικα που εκτελέσαμε πατώντας

```
>>figure; imshow(frame); title('Video Frame');
>>figure; imshow(foreground); title('Foreground');
```

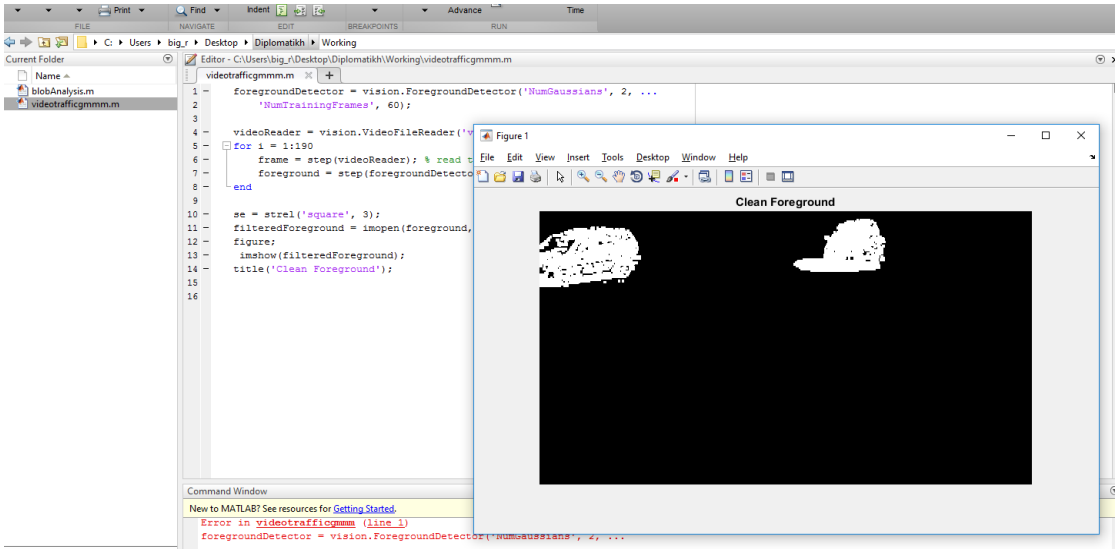


7.2 Simulink Foreground Model with GMM results

Παρόλα αυτά η διαδικασία διαχωρισμού του παρασκηνίου δεν είναι ικανοποιητική και συχνά υπάρχει ανεπιθύμητος θόρυβος (noise) που μας αλλοιώνει τα αποτελέσματα. Για την εύρεση και τον διαχωρισμό του θορύβου χρησιμοποιούμε Morphological Opening. Morphological Opening είναι η διαστολή της διάβρωσης ενός συνόλου A από ένα δομικό στοιχείο B δηλαδή διαχωρίζει μικρά αντικείμενα από το παρασκήνιο μιας εικόνας (συνήθως είναι Pixels μεγάλης φωτεινότητας) και τα τοποθετεί στο παρασκήνιο κλείνοντας όλες τις υπάρχουσες ‘τρύπες’ και μετατρέποντας μικρές περιοχές του παρασκηνίου σε πρώτο πλάνο (foreground). Επίσης αυτή η τεχνική είναι πολύ χρήσιμη για την εύρεση συγκεκριμένων σχημάτων σε μια εικόνα.

```
se = strel('square', 3);  %η συνάρτηση strel εκτελεί το Morphological Opening
filteredForeground = imopen(foreground, se); %άνοιγμα του πίνακα που περιέχει τα pixels με όλες τις πληροφορίες για τον background
figure;
 imshow(filteredForeground);
title('Clean Foreground');
```

Το αποτέλεσμα είναι μια κάπως πιο βελτιωμένη έκδοση της προηγούμενης προσπάθειας μας:



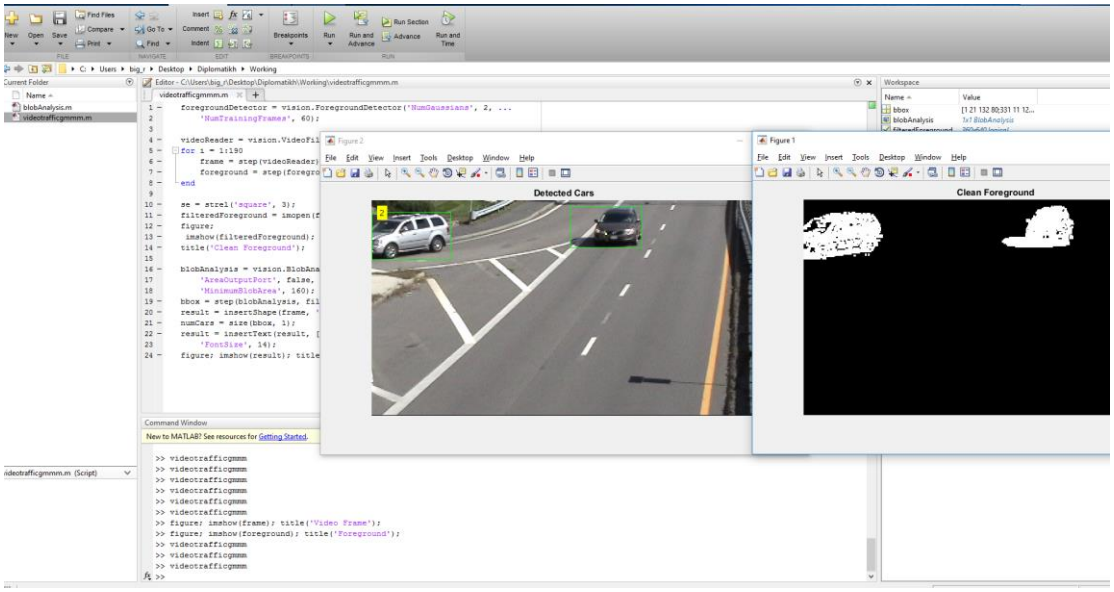
7.3 Simulink Foreground Model with GMM results

Στη συνέχεια χρησιμοποιούμε το μοντέλο του Simulink vision.BlobAnalysis object για να φτιάξουμε τα πλαίσια οριοθέτησης που αντιστοιχούν σε ένα κινούμενο αμάξι. Το αντικείμενο φιλτράρει περαιτέρω την foreground detection απορρίπτοντας άμορφες μάζες που έχουν μέγεθος μικρότερο από 160 pixels.

```
blobAnalysis = vision.BlobAnalysis('BoundingBoxOutputPort', true, 'AreaOutputPort',
false, 'CentroidOutputPort', false,  'MinimumBlobArea', 160);
bbox = step(blobAnalysis, filteredForeground);
result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green'); %δημιουργία
πράσινων κουτιών γύρω από τα αμάξια για να τα μαρκάρουμε
```

Για να κάνουμε καταμέτρηση πόσα αμάξια περνάνε από το οπτικό μας πεδίο μετράμε τον αριθμό των πλαισίων οριοθέτησης καθώς κάθε πλαίσιο αντιστοιχεί σε ένα αμάξι.

```
numCars = size(bbox, 1);
result = insertText(result, [10 10], numCars, 'BoxOpacity', 1, ...
   'FontSize', 14);
figure; imshow(result); title('Detected Cars');
```

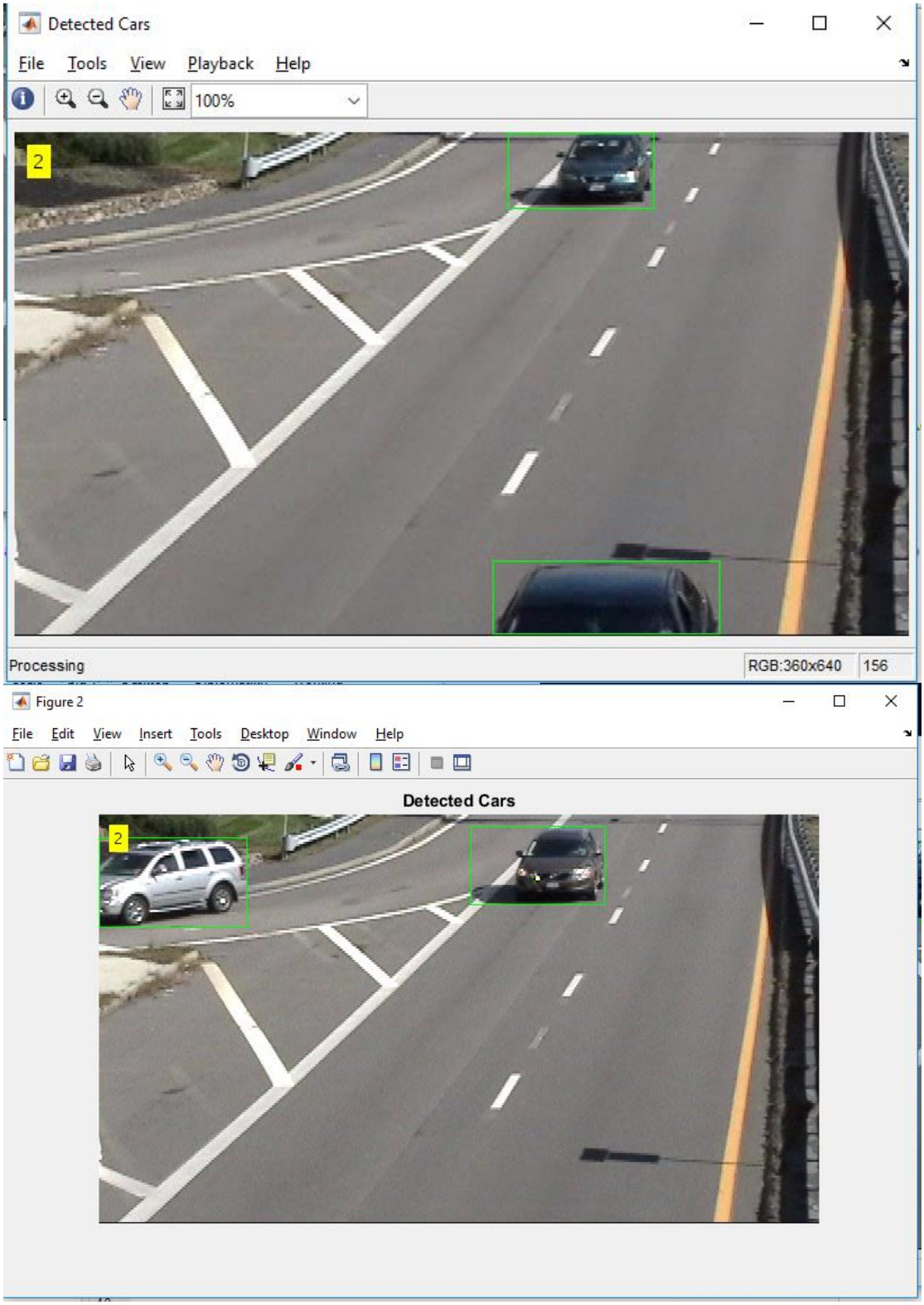


7.4 Simulink Foreground Results with GMM and green highlight

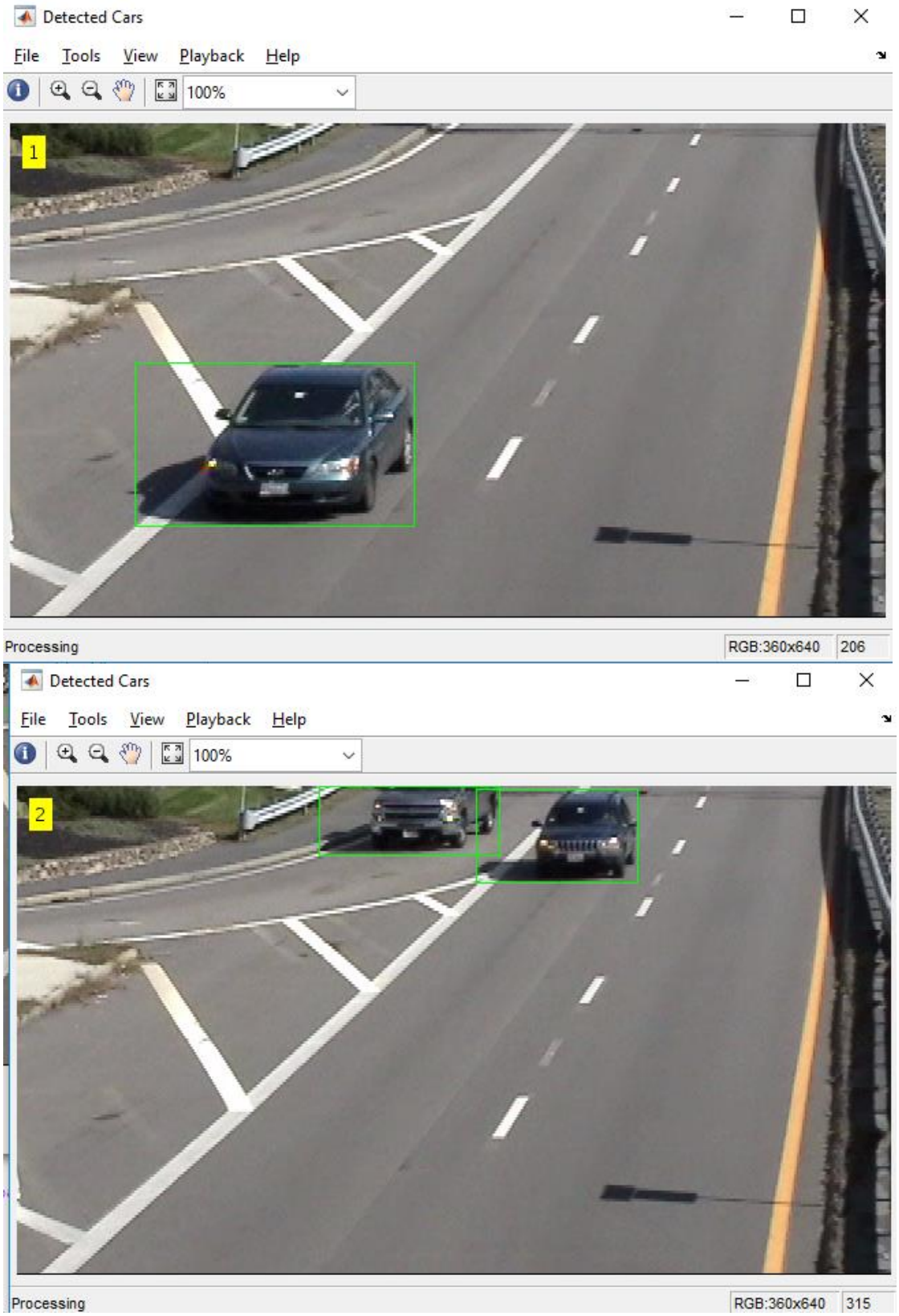
Τέλος για να ολοκληρώσουμε την διαδικασία πρέπει να επεξεργαστούμε και τα υπόλοιπα Video Frames:

```
videoPlayer = vision.VideoPlayer('Name', 'Detected Cars');
videoPlayer.Position(3:4) = [650,400];  % μέγεθος παραθύρου [πλάτος,ύψος]
se = strel('square', 3); % διαδικασία Morphological filter για την αφαίρεση θορύβου

while ~isDone(videoReader) %επανάληψη μέχρι την λήξη του video reader

   frame = step(videoReader); % ανάγνωση του επόμενου video frame

   foreground = step(foregroundDetector, frame); % ανίχνευση του foreground στο τρέχων video frame

   filteredForeground = imopen(foreground, se); %αφαίρεση θορύβου του foreground με morphological filter

   bbox = step(blobAnalysis, filteredForeground); %ανίχνευση συνδεδεμένων στοιχείων με την ελάχιστη περιοχή που δώθηκε και υπολογισμός πλασίων οριοθέτησης

   result = insertShape(frame, 'Rectangle', bbox, 'Color', 'green');  % δημιουργία πλαισίων οριοθέτησης γύρω από το κινούμενο αμάξι

   numCars = size(bbox, 1);
   result = insertText(result, [10 10], numCars, 'BoxOpacity', 1,'FontSize', 14); % ένδειξη αριθμού αμαξιών που βρέθηκαν στο συγκεκριμένο video frame

   step(videoPlayer, result);  % ένδειξη αποτελεσμάτων
end

release(videoReader); % κλείσιμο αρχείου βίντεο και λήξη του βρόγχου
```

Με την εκτέλεση του κώδικα θα εμφανιστεί ένα βίντεο το οποίο θα δείχνει αμάξια να κινούνται και τα ανάλογα πλαίσια οριοθέτησης γύρω τους με τον αριθμό αναγνώρισης αμαξιών πάνω αριστερά. Μερικά στιγμιότυπα από το video:



7.5-7.6 Simulink GMM results with green highlight and counter



7.7-7.8 Simulink GMM results with green highlight and counter

Ανίχνευση πολλαπλών αντικειμένων βασισμένα στην κίνηση (Moving Average Background)

Σε αυτό το σημείο θα παρουσιάσουμε ένα παράδειγμα αυτόματης κίνησης πολλαπλών αντικειμένων σε κίνηση από μία στατική κάμερα. Αυτή η εφαρμογή είναι πολύ σημαντική σε εφαρμογές computer-vision, συμπεριλαμβανομένων των εφαρμογών ανίχνευσης κίνησης, traffic monitoring και αυτόματης ασφάλειας. Για να απλουστεύσουμε την πολυπλοκότητα του αλγορίθμου αυτού θα τα αναλύσουμε σε δύο επιμέρους θέματα.

1. Ανίχνευση αντικειμένων σε κάθε frame
2. Συσχέτιση της ανίχνευσης που αντιστοιχεί στο ίδιο αντικείμενο ανά χρόνο.

Όπως και πριν η ανίχνευση των αντικειμένων εφαρμόζει έναν αλγόριθμο αφαίρεσης παρασκηνίου βασισμένο στο Gaussian Mixture Model. Στην συνέχεια εφαρμόζεται morphological opening και εφαρμόζονται στην μάσκα foreground για να ελαχιστοποιήσουν τον θόρυβο. Τέλος η blob analysis ανιχνεύει τις ομάδες των συνδεδεμένων pixel τα οποία υποδηλώνουν την ύπαρξη του κινούμενου αντικειμένου ενδιαφέροντος.

Η Συσχέτιση των ανιχνεύσεων στο ίδιο αντικείμενο βασίζεται αποκλειστικά στην κίνηση. Η κίνηση του κάθε αντικειμένου βασίζεται σε ένα φίλτρο Kalman. Το φίλτρο χρησιμοποιείτε για να προβλέψει την τοποθεσία κάθε frame και να καθορίσει την πιθανότητα να εφαρμοστεί εκ νέου ανίχνευση σε κάθε βήμα του αντικειμένου.

Σε ένα δοσμένο frame κάποιες ανιχνεύσεις μπορεί να εκχωρηθούν στις διαδρομές ενώ άλλες ανιχνεύσεις να μείνουν ανεκχώρητες. Οι καταχωρημένες διαδρομές γίνονται συνεχώς updated χρησιμοποιώντας αντίστοιχες ανιχνεύσεις. Οι ανεκχώρητες διαδρομές δηλώνονται σαν αόρατες από τον αλγόριθμο και ξεκινάει μια εκ νέου αναγνώριση για αυτή τη διαδρομή. Κάθε ανίχνευση κρατάει τον αριθμό των συνεχόμενων frames που παρέμειναν ανεκχώρητα. Αν ο αριθμός αυτός ξεπεράσει ένα συγκεκριμένο όριο ο αλγόριθμος θεωρεί ότι το αντικείμενο έφυγε από το πεδίο όρασης και διαγράφει το ίχνος.

Kalman Filter

Γνωστό και ως τετραγωνική εκτίμηση το Kalman Filter είναι ένας αλγόριθμος ο οποίος μέσα σε παρατηρήσεις συγκεκριμένου χρονικού διαστήματος χρησιμοποιεί μια σειρά μετρήσεων που περιέχουν στατιστικό θόρυβο και άλλες ανακρίβειες-σφάλματα. Στο τέλος παράγει υπολογισμούς των άγνωστων μεταβλητών που τείνουν να είναι πιο ακριβείς από αυτούς που βασίζονται σε μια απλή μέτρηση μόνο. Αυτό το επιτυγχάνει χρησιμοποιώντας Bayesian Inference και υπολογίζοντας τις από κοινού κατανομές πιθανοτήτων από τις μεταβλητές σε κάθε δείγμα.

Bayesian Inference

Είναι μία μέθοδος στατιστικής συμπερασματολογίας όπου χρησιμοποιείτε το Bayes Theorem για να ανανεώσει την πιθανότητα μιας υπόθεσης καθώς περισσότερες πληροφορίες γίνονται διαθέσιμες.

Joint Probability Distribution

Είναι μια διανομή πιθανοτήτων που δίνει το ενδεχόμενο ότι κάθε μεταβλητή X,Y που είναι καθορισμένη στον χώρο πιθανότητας μπορεί να παρουσιάσει σφάλμα σε οποιαδήποτε έκταση ή διακριτή ομάδα από καθορισμένες μεταβλητές.

Blob Analysis

Η ανάλυση αυτή ωφείλεται σε 3 βήματα:

1) Extraction – στα αρχικα βηματα εφαρμοζεται τεχνικη image thresholding για να παρουμε την περιοχη ενδιαφεροντος για τα αντικειμενα μας
2) Refinement– η περιοχη που εχουμ εξεαγει εχει συχνα θορυβο η αλλα προ βληματα όπως κακος φοτισμος. Σε αυτό το βημα η περιοχη ενδιαφεροντος ενισχυεται μεσα από τεχνικες μεταμορφωσης ευκρινειας
3) Analysis- στο τελικο βημα η αποσαφηνιμενη περιοχη υποοκειτε μετρησεις κ υπολογιζονται τα τελικα αποτελεσματα. Αν μια περιοχη εχει πολλαπλα αντικειμενα τοτε χωριζονται σε 2 blobs που ελεγχονται το κάθε ένα ξεχωριστα

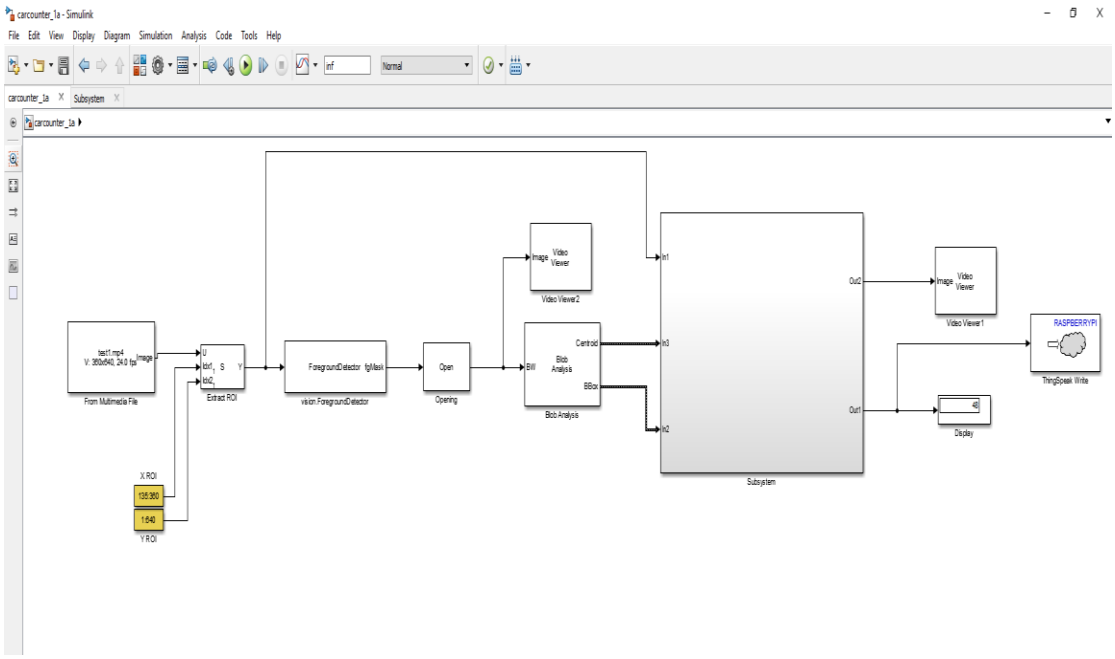
Region λεμε ενα αυποσύνολο pixels στο συνολο της εικονας

Blob λεμε 2 συνδεδεμενες περιοχες-regions

Καταμέτρηση αυτοκινήτων και ανάλυση κίνησης στο ThingSpeak από βίντεο στατικής κάμερας

Μετά από όλα τα παραπάνω βήματα που αναλύσαμε είμαστε πλέον έτοιμη να κατανοήσουμε τη διαδικασία ανάλυσης κ καταμέτρησης αντικειμένων και να πάμε ένα βήμα ακόμα πιο πέρα ανεβάζοντας τα στατιστικά μας στο thingspeak.

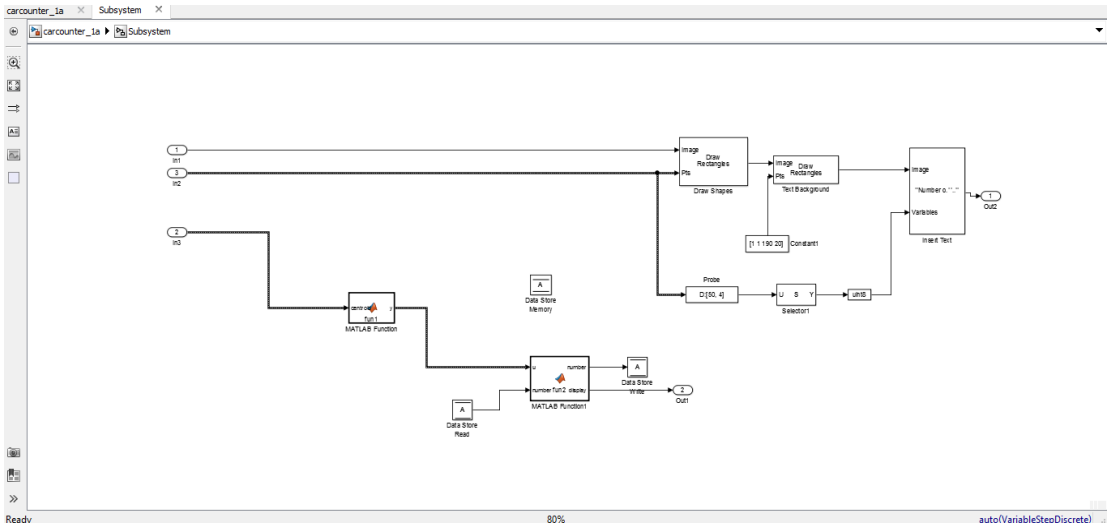
Το μοντέλο simulink το οποίο θα χρησιμοποιήσουμε για την επίτευξη του σκοπού αυτού είναι το παρακάτω:



8.1 Car counter model with video input

Το μοντέλο αυτό παίρνει είσοδο από ένα έτοιμο βίντεο το οποίο βρήκαμε στο ιντερνέτ από μία στατική κάμερα κυκλοφορίας , το περνάει από έναν επιλογέα ενδιαφέροντος περιοχής (Region Of Interest Selector) ή ROI και του δίνεται έτσι ένα συγκεκριμένο πλάτος και ύψος ώστε να περιορίσουμε το οπτικό πεδίο μας στο σημείο που μας ενδιαφέρει (μονή λωρίδα κυκλοφορίας). Στη συνέχεια το σήμα εισέρχεται στον Foreground Detector όπως είδαμε και πριν που είναι υπεύθυνος για την ανίχνευση των αντικειμένων ενδιαφέροντος εφαρμόζοντας Gaussian Mixture Model και στη συνέχεια εφαρμόζεται morphological opening το οποίο ευθύνεται για την βελτίωση των περιοχών υποσυνόλων των pixels ώστε να πάρουμε καλύτερα και μεγαλύτερης ακρίβειας αποτελέσματα. Αυτό βγαίνει σε μία έξοδο (Video viewer 2) Και ταυτόχρονα εφαρμόζεται blob analysis η οποία δημιουργεί τα πράσινα πλαίσια γύρω από τα αυτοκίνητα.

Όλα αυτά εισέρχονται σε ένα υποσύστημα Subsystem που αν το αναπτύξουμε πατώντας διπλό κλικ πάνω του μας εμφανίζεται το εξής σχηματικό:

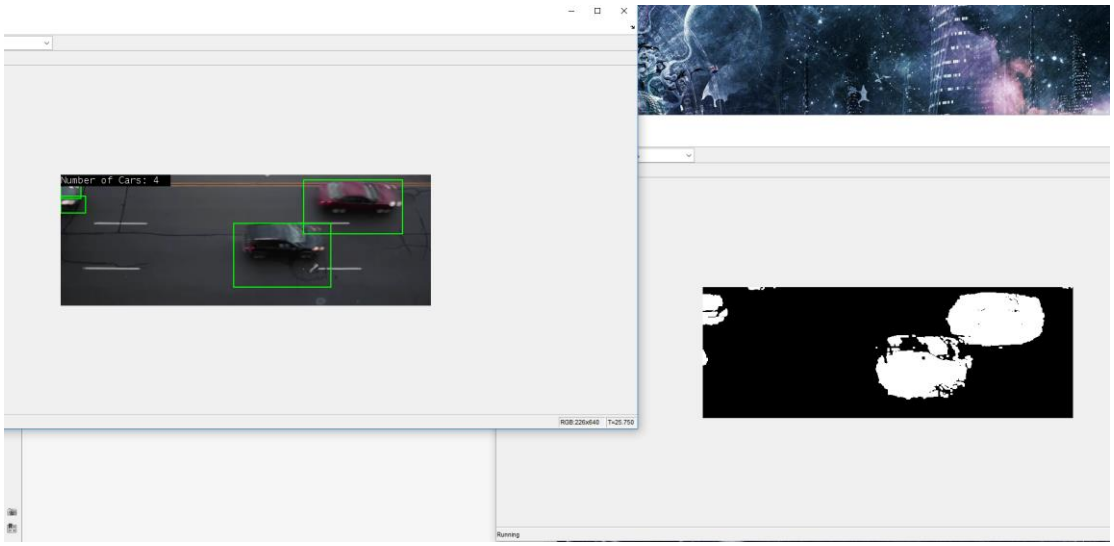


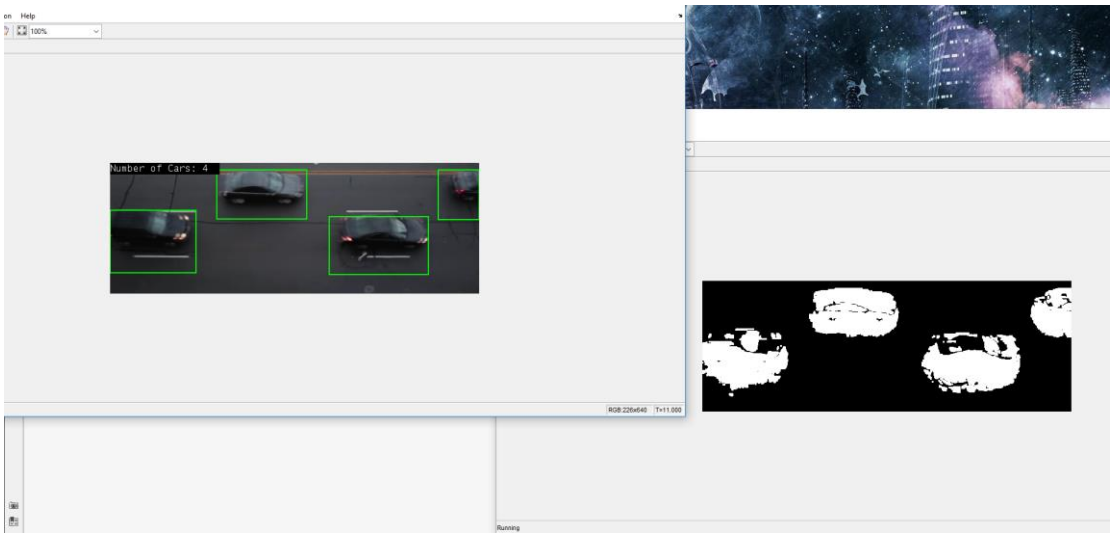
8.2 Car counter sub model with video input

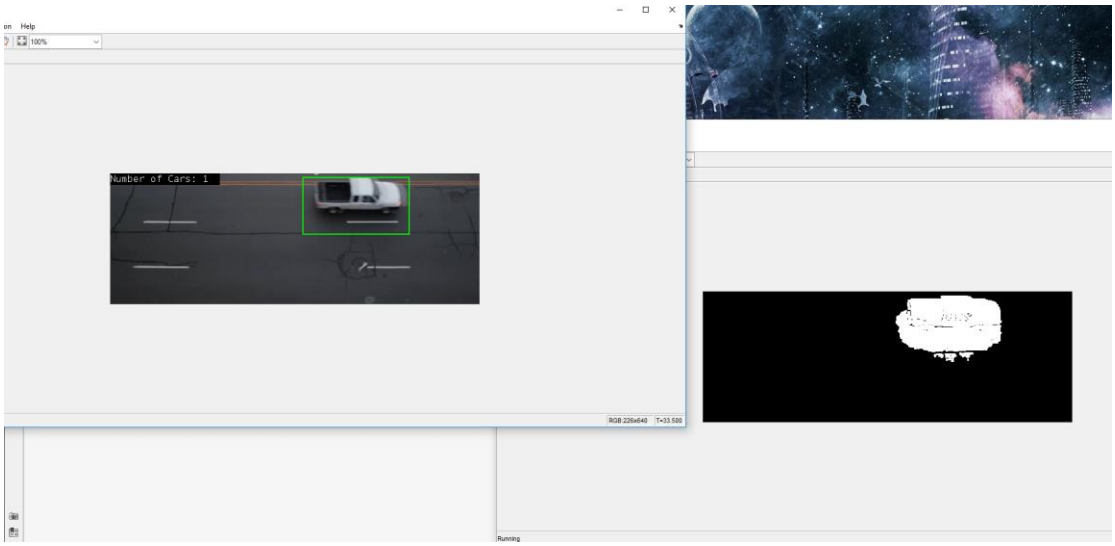
Στο κάτω μέρος του: αυτό το μοντέλο εκτελεί δύο απλές συναρτήσεις, την cancroids και την Matlab function που είναι υπεύθυνες για την δημιουργία πλαισίων και ενός καταμετρητή ο οποίος αυξάνεται κατά ένα κάθε φορά που ανιχνεύεται στην περιοχή ενδιαφέροντος ένα πλαίσιο. Το block Data store memory είναι ο αποθηκευτικός χώρος τον τιμών μας και τα A blocks Write/Read Αντίστοιχα διαβάζουν κ γράφουν στο block A μνήμης. Οι τιμές αποθηκεύονται καθώς το data store read διαβάζει και το data store write γράφει τις τιμές του πλήθους των αυτοκινήτων που ανιχνεύονται στέλνοντας τα σε μία έξοδο output 2.

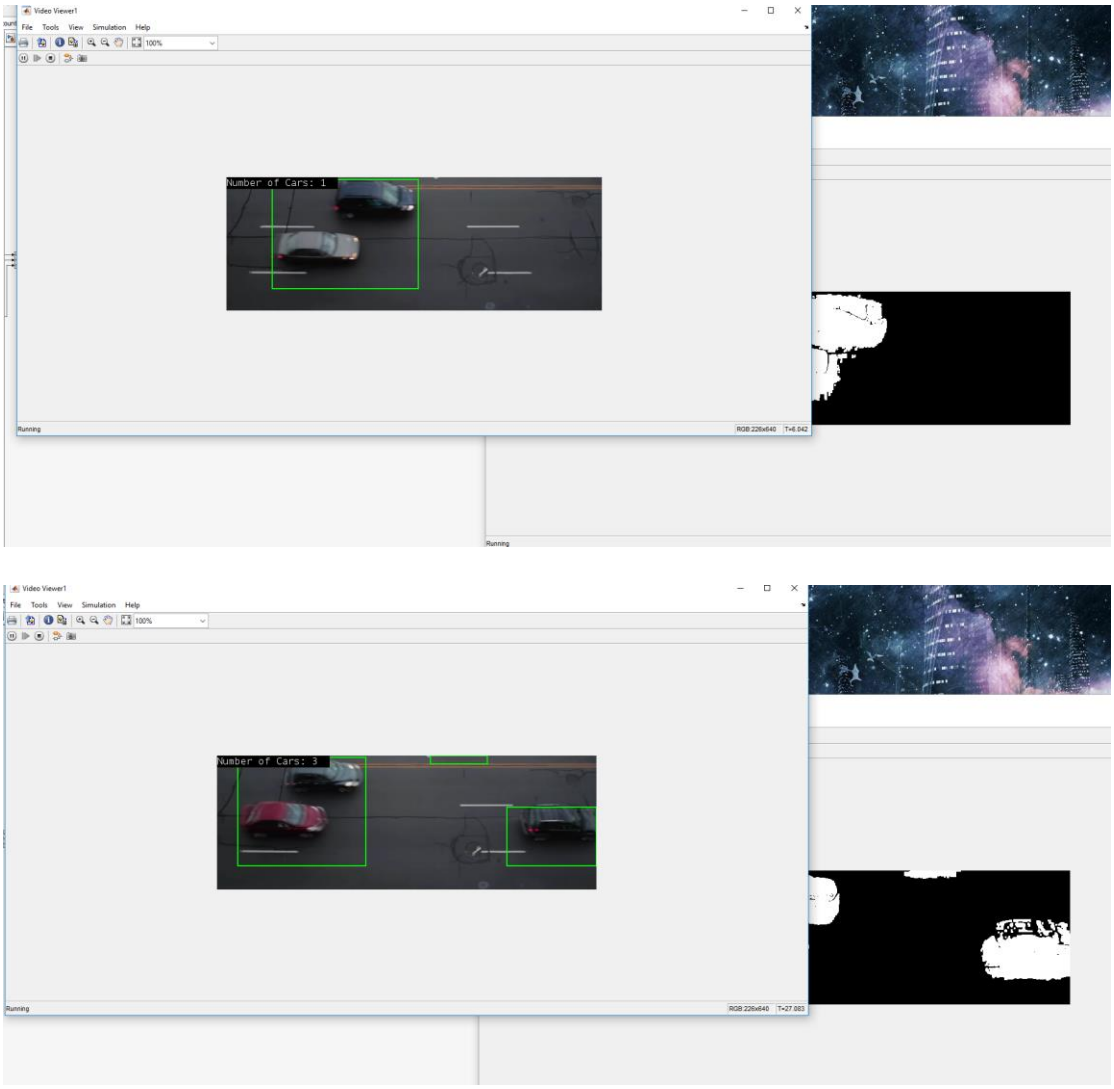
Στο πάνω μέρος του: το block draw shapes σχεδιάζει τα ορθογώνια πλαίσια ανίχνευσης για τα αντικείμενα ενδιαφέροντος καθώς παίρνει σαν είσοδο int2 ένα σήμα μεταβλητής από τα αποτελέσματα της blob analysis . Η Probe είναι υπεύθυνη για την δημιουργία ενός δυσδιάστατου δυναμικού πίνακα με μέγιστο μέγεθος 50χ4 ο οποίος αποθηκεύει τον αριθμό των πράσινων πλαισίων που βρίσκονται στην οθόνη συνολικά. Στην συνέχεια ο Selector κρατάει μόνο της στήλες του πίνακα που αντιστοιχούν στον αριθμό των πλαισίων τα μετατρέπει σε ακέραιο ώστε να μπορεί να σταλθεί στην έξοδο και στην οθόνη το αποτέλεσμα. Το μοντέλο Constant 1 δηλώνει την τοποθεσία του κουτιού το οποίο αναγράφει πόσα αυτοκίνητα υπάρχουν στην οθόνη ανά frame δηλώνοντας του ότι ξεκινάει από το pixel 1x ,1y και το κουτί θα έχει width 190 pixels και height 20 pixels. Τέλος η Text background παίρνει όλες τις μεταβλητές που προαναφέρθηκαν και τις δίνει όλες μαζί στο μοντέλο Insert Text όπου είναι αρμόδιο για το κείμενο που αναγράφεται πάνω αριστερά “number of cars” σε συνδυασμό με τον Integer που παίρνει από την uint8 βγάζει το κείμενο number of cars: ακολουθούμενο από τον αριθμό των αυτοκινήτων που βρίσκονται στην περιοχή ενδιαφέροντος εκείνη τη χρονική στιγμή.
Οι τελικές τιμές εξόδου βγαίνουν σε ένα video viewer και μια οθόνη όπως φαίνετε στο σχηματικό καθώς επίσης τα data analytics Που συλλέξαμε ανεβαίνουνε στο ThingSpeak.

Τα στιγμιότυπα που παίρνουμε από την εκτέλεση του κώδικα είναι τα ακόλουθα:









8.3-8.7 Screenshot results from simulink model with video input

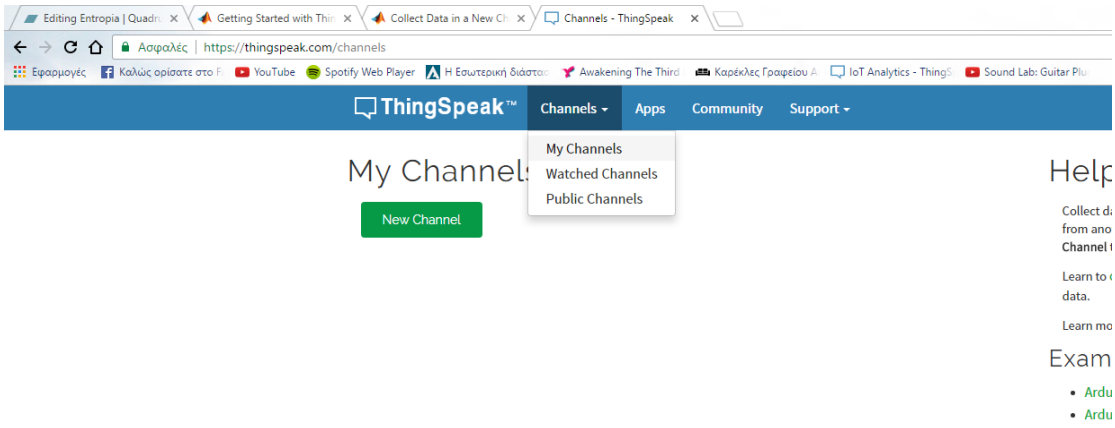
Όπως βλέπουμε τα αποτελέσματα του κώδικά μας είναι κατά προσέγγιση σωστά αλλά όχι 100% ακριβείας.

ThingSpeak Support Packages Toolbox

Για να μπορέσουμε να στείλουμε και να εμφανιστούν τα δεδομένα μας στα charts του thingspeak χρειαζόμαστε να έχουμε εγκατεστημένο το ThingSpeak Support Package , Toolbox. Για να το κάνουμε αυτό κατεβάζουμε το συγκεκριμένο toolbox από την Mathworks και το κάνουμε unzip στον φάκελο Matlab>Support Packages>Toolbox και αφού ανοίγουμε το Matlab επιλέγουμε τον συγκεκριμένο φάκελο.

<u>Ανάγνωση – Εγγραφή – Διαγραφή analytics από Thing Speak</u>

Για να συλλέξουμε δεδομένα σε ένα νέο κανάλι του thing speak αρχικά δημιουργούμε ένα νέο κανάλι εφόσον φτιάξουμε πρώτα έναν λογαριασμό στο thingspeak.



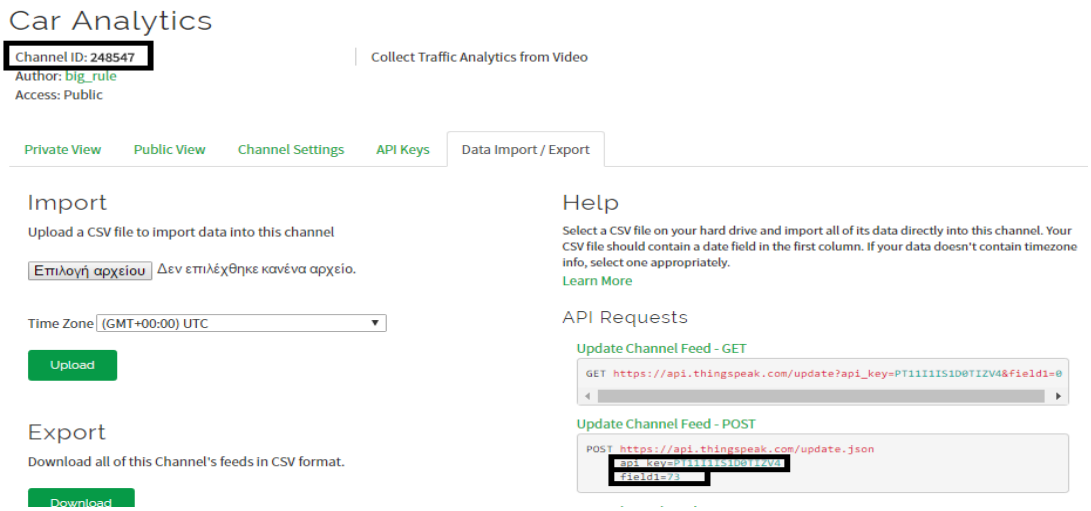
9.1 Thingspeak channel

Στην σελίδα Channels πατάμε New Channel και μαρκάρουμε τα checkboxes δίπλα από το πεδίο 1.Name: Car Analytics

Μετά την εκτέλεση του μοντέλου εκτελούμε στο Matlab την εντολή :

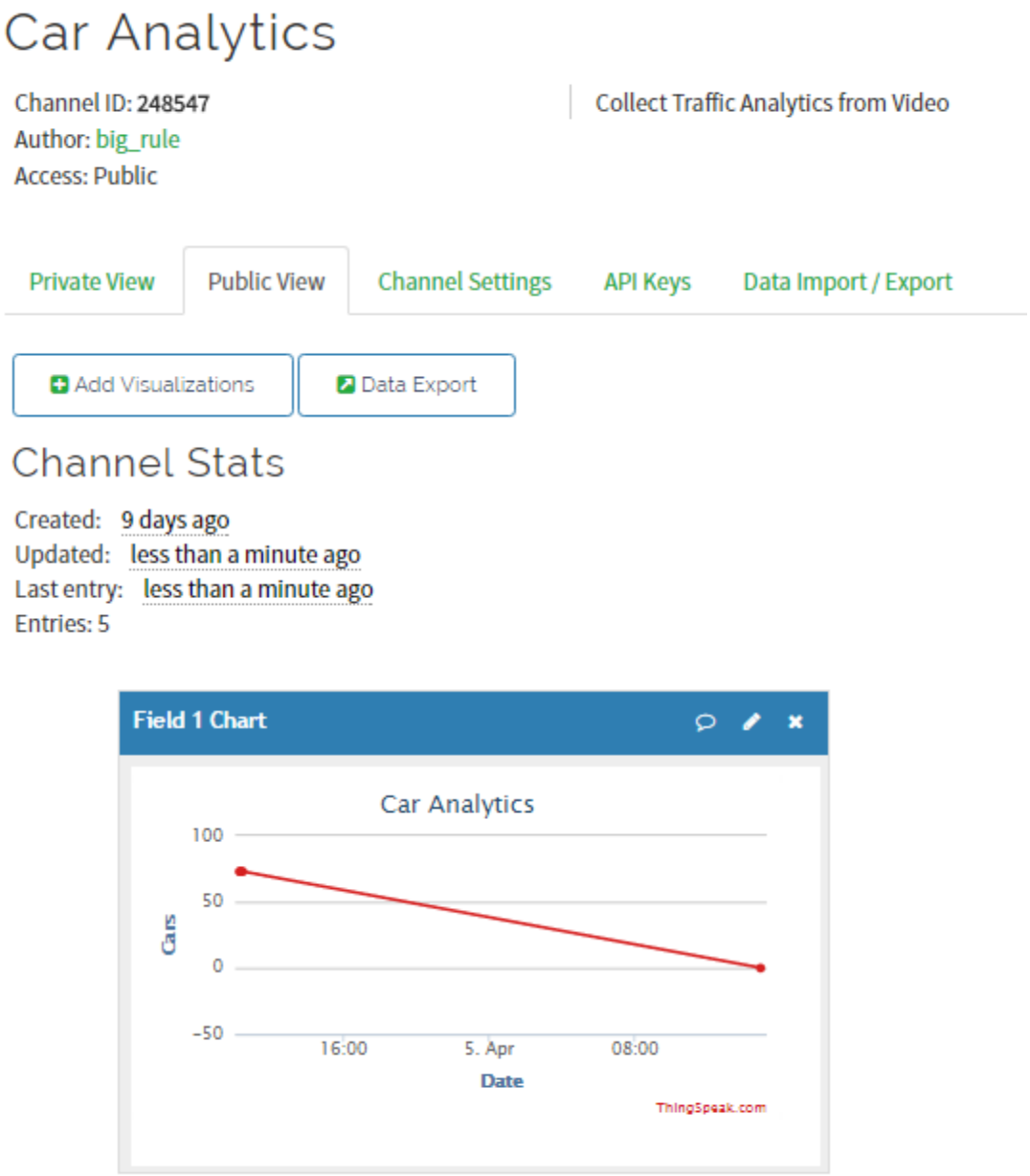
>> thingSpeakWrite(248547,73,'WriteKey','PT11I1IS1D0TIZV4');

όπου:
>> thingSpeakWrite(channel ID,Field,'WriteKey',’channel API Write Key’);



9.2 Thingspeak channel

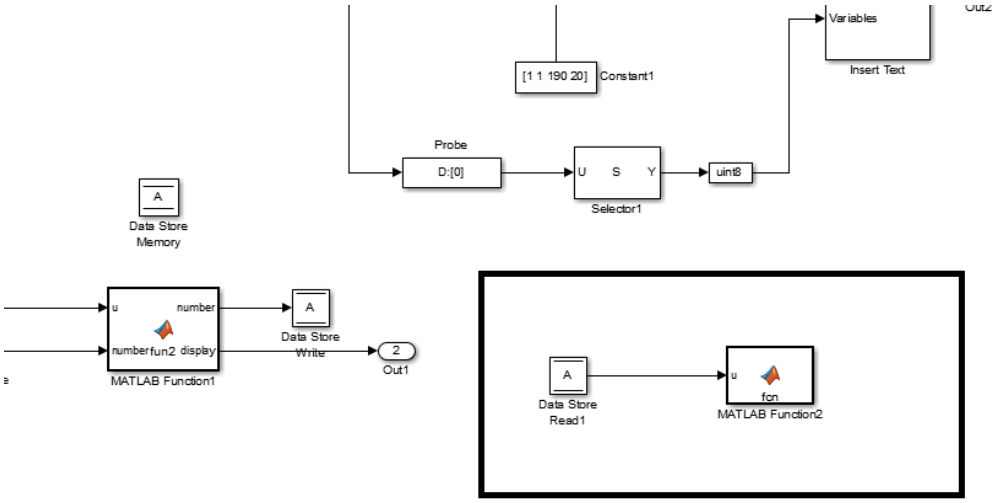
Με την εντολή αυτή στέλνουμε request στο thingspeak για να κάνει update και να περάσει τα νέα analytics που του στέλνει η εκτέλεση του μοντέλου μας.
Τα αποτελέσματα με τα statistics στο thing speak θα είναι τα ακόλουθα:



9.3 Thingspeak statistics

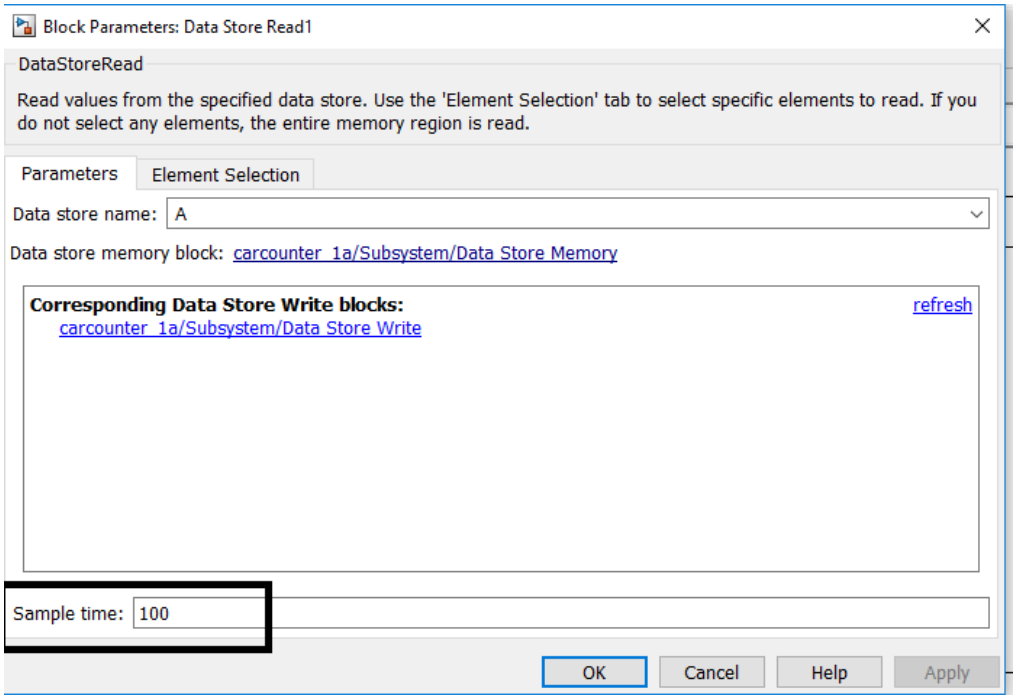
Για τους σκοπούς της εργασίας μας χρειαζόμαστε real time analytics capture οπότε δεν είναι δυνατόν να στέλνουμε τα δεδομένα μας μέσω εντολών. Θέλουμε να γίνεται αυτόματα. Για αυτό το λόγο φτιάχνουμε ένα μικρό και εύκολο μοντέλο το οποίο θα στέλνει post requests στο thingspeak αυτόματα κάθε 100 seconds.

Το model είναι εμφωλευμένο μέσα στο Sub model και είναι το εξής :

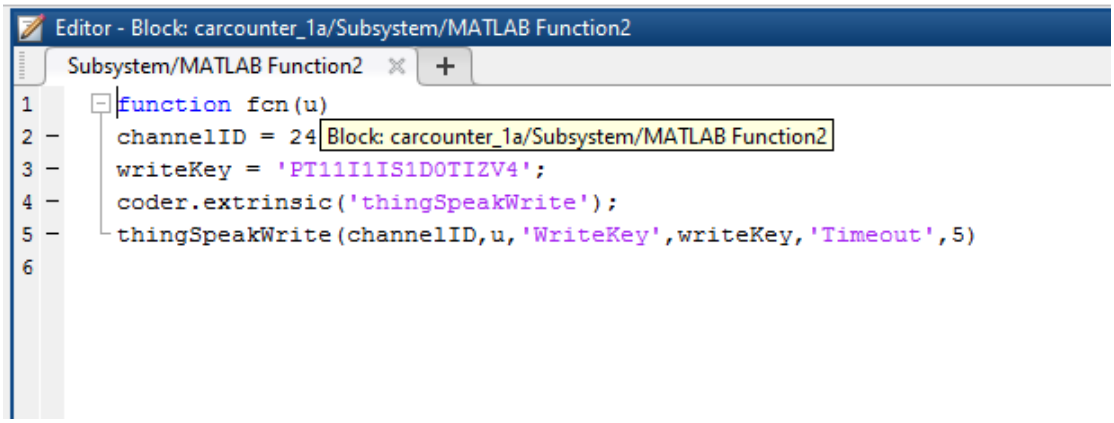


9.4 submodel of simulink car counter model

Το Data store Read 1 είναι υπεύθυνο για τον χρονισμό και την αποστολή των δεδομένων καθώς το MATLAB Function2 αποτελείτε από έναν απλό κώδικα που στέλνει τα analytics στο κανάλι Thingspeak Που έχει τα στοιχεία που του βάζουμε εμείς:

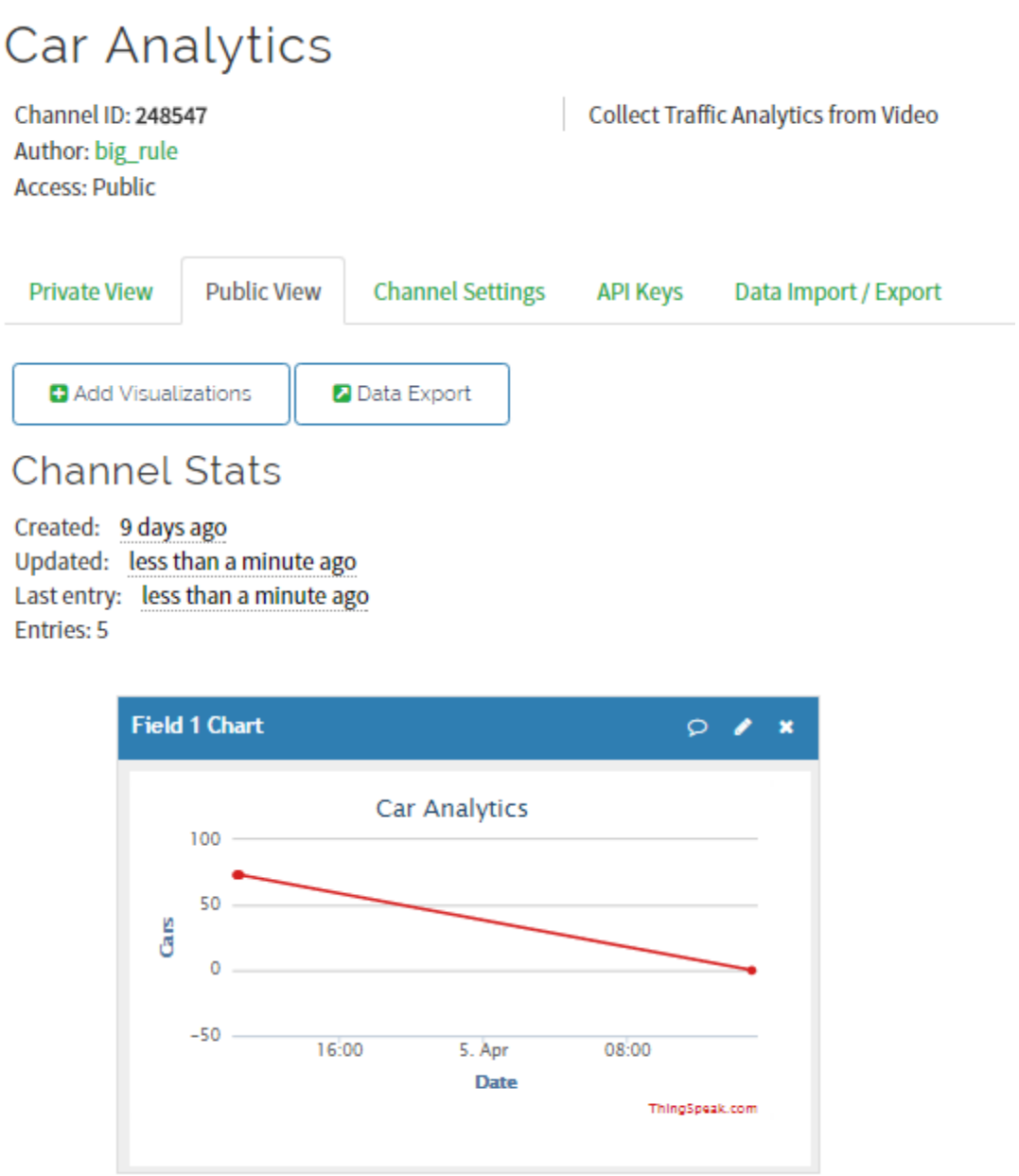


9.5 store data block



9.6 autoupdate algorithm

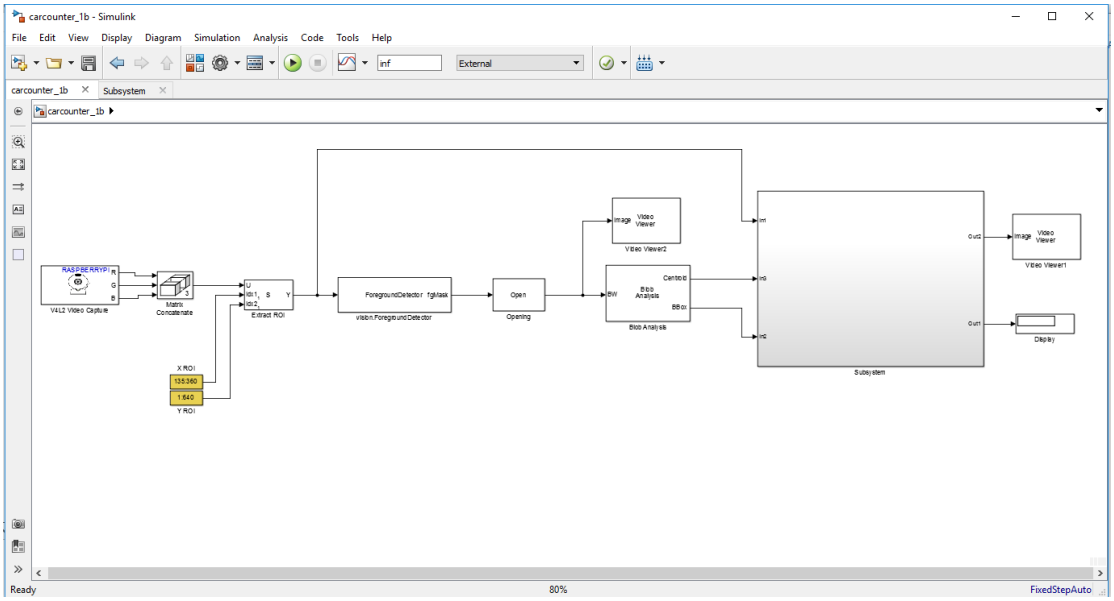
Με την εκτέλεση του μοντέλου μας περιμένοντας ένα λεπτό μπαίνουμε στο κανάλι που δημιουργήσαμε στο Thingspeak και βλέπουμε τα αποτελέσματα:



9.7 Thingspeak statistics

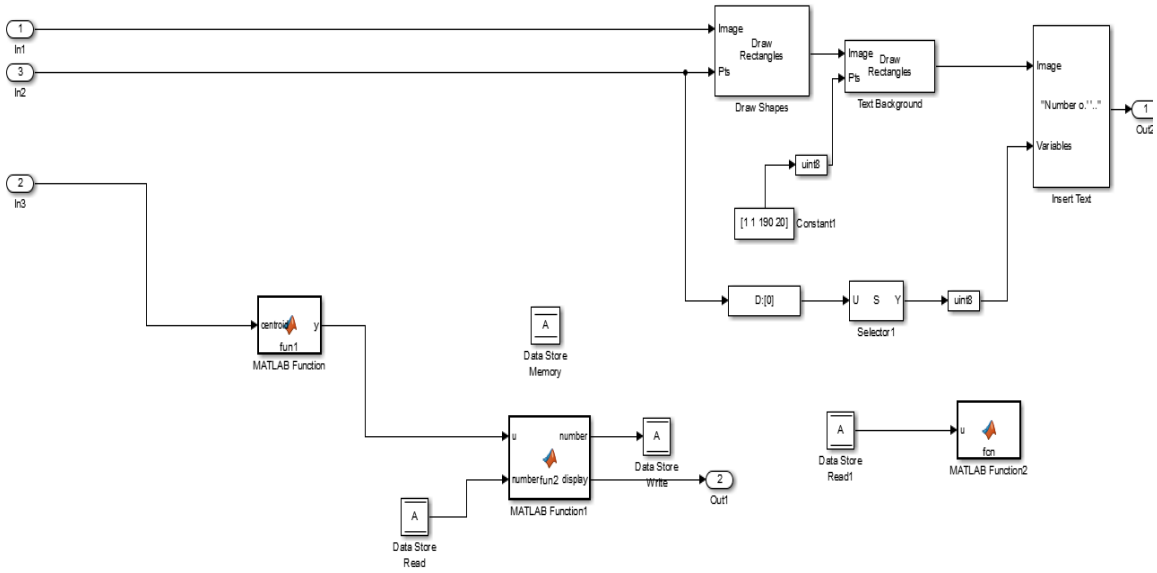
Καταμέτρηση αυτοκινήτων και ανάλυση κίνησης στο Thing Speak από video πραγματικού χρόνου στο Raspberry Pi με κάμερα.

Προηγούμενος ανιχνεύσαμε και μετρήσαμε επιτυχώς την κυκλοφορία ενός δρόμου με πηγή ένα έτοιμο βίντεο από μια στατική κάμερα δρόμου κυκλοφορίας. Σε αυτό το παράδειγμα θα εφαρμόσουμε ακριβώς την ίδια διαδικασία μόνο που τώρα θα γίνει σε πραγματικό χρόνο από την κάμερα του Raspberry Pi.

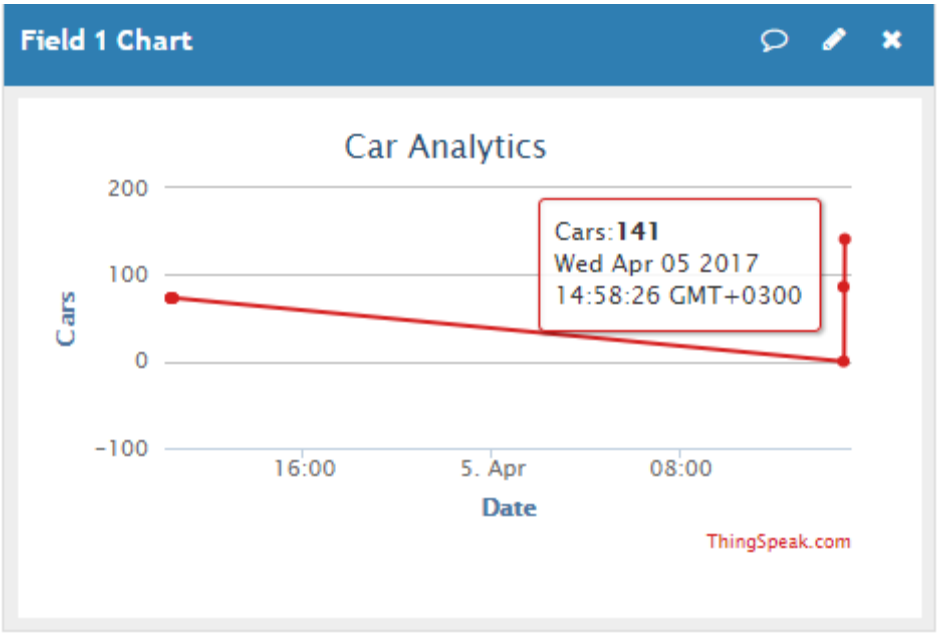


10.1 simulink car counter with camera input

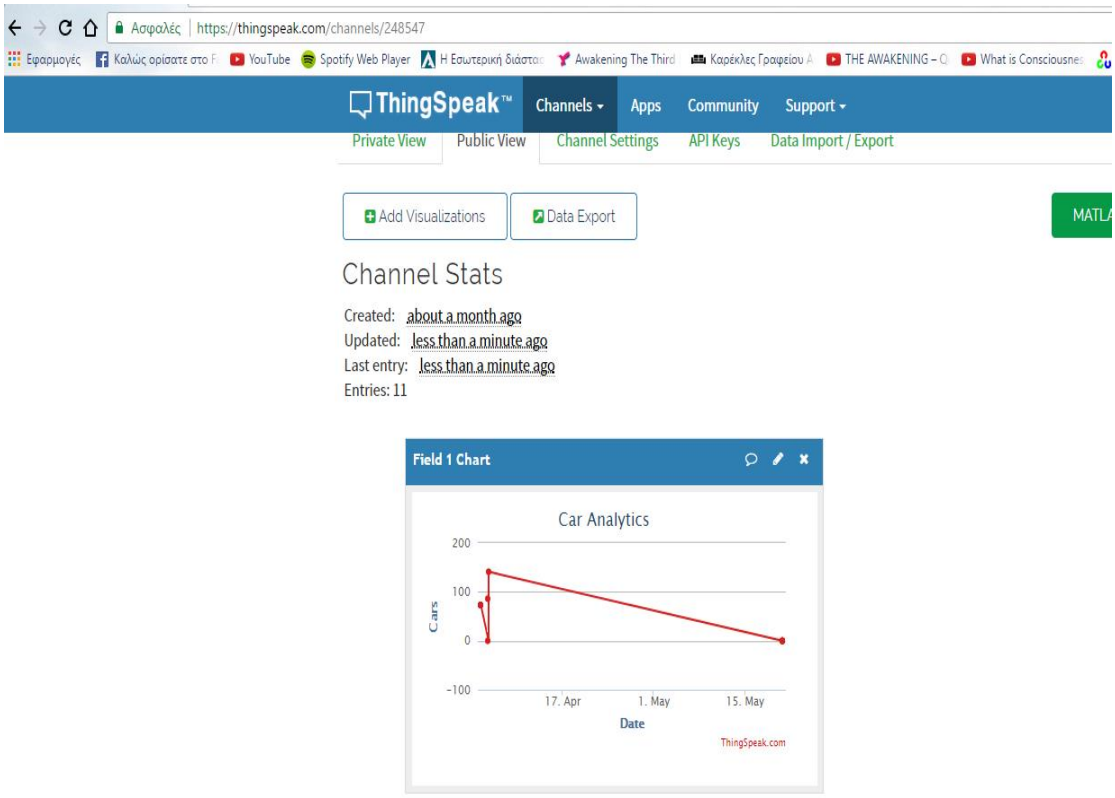
Όπως βλέπουμε το σχηματικό είναι ακριβώς το ίδιο με μόνη διαφορά την είσοδο που στο συγκεκριμένο παράδειγμα είναι η κάμερα του raspberry pi. Επίσης προσθέσαμε στο sub model ένα uint8 στο constant1 διότι έβλεπε το προηγούμενο μπλοκ σαν πραγματικό αριθμό ενώ ήθελε ακέραιο. Πολλές φορές χρειάζεται να κάνουμε προσαρμογή των αριθμητικών τύπων. Επίσης προσθέσαμε ένα matrix concatenate στην αρχή μετά την κάμερα μας γιατί θέλουμε το R-G-B σήμα (3 πίνακες ξεχωριστά-ένας για κάθε χρώμα) να γίνει ένα τρισδιάστατο σήμα (multidimensional). Εκτελώντας το σχηματικό μας σε External mode έχοντας συνδεδεμένο το raspberry τοπικά με τον υπολογιστή μας συλλέγουμε τα στοιχεία και πηγαίνοντας στο κανάλι μας του Thing Speak βλέπουμε τα νέα στατιστικά που έχουμε συλλέξει αυτή τη φορά από έναν δρόμο της Αθήνας σε πραγματικό χρόνο (Real Time Capture).

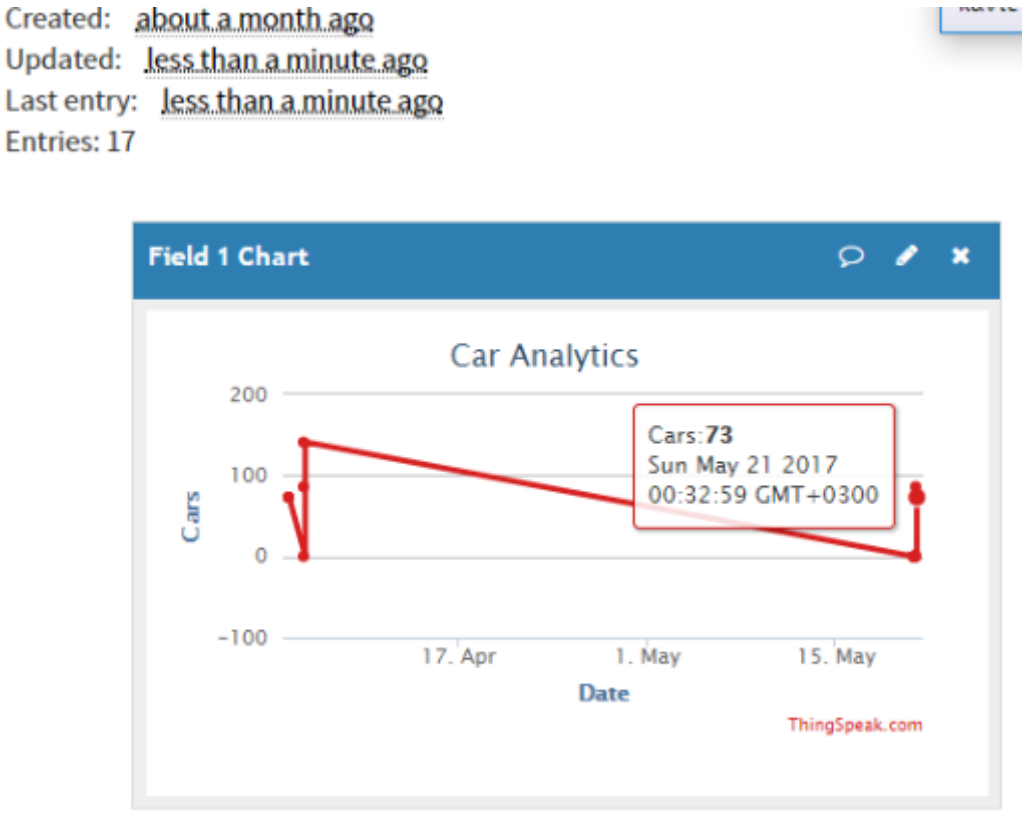


10.2 simulink car counter submodel with camera input

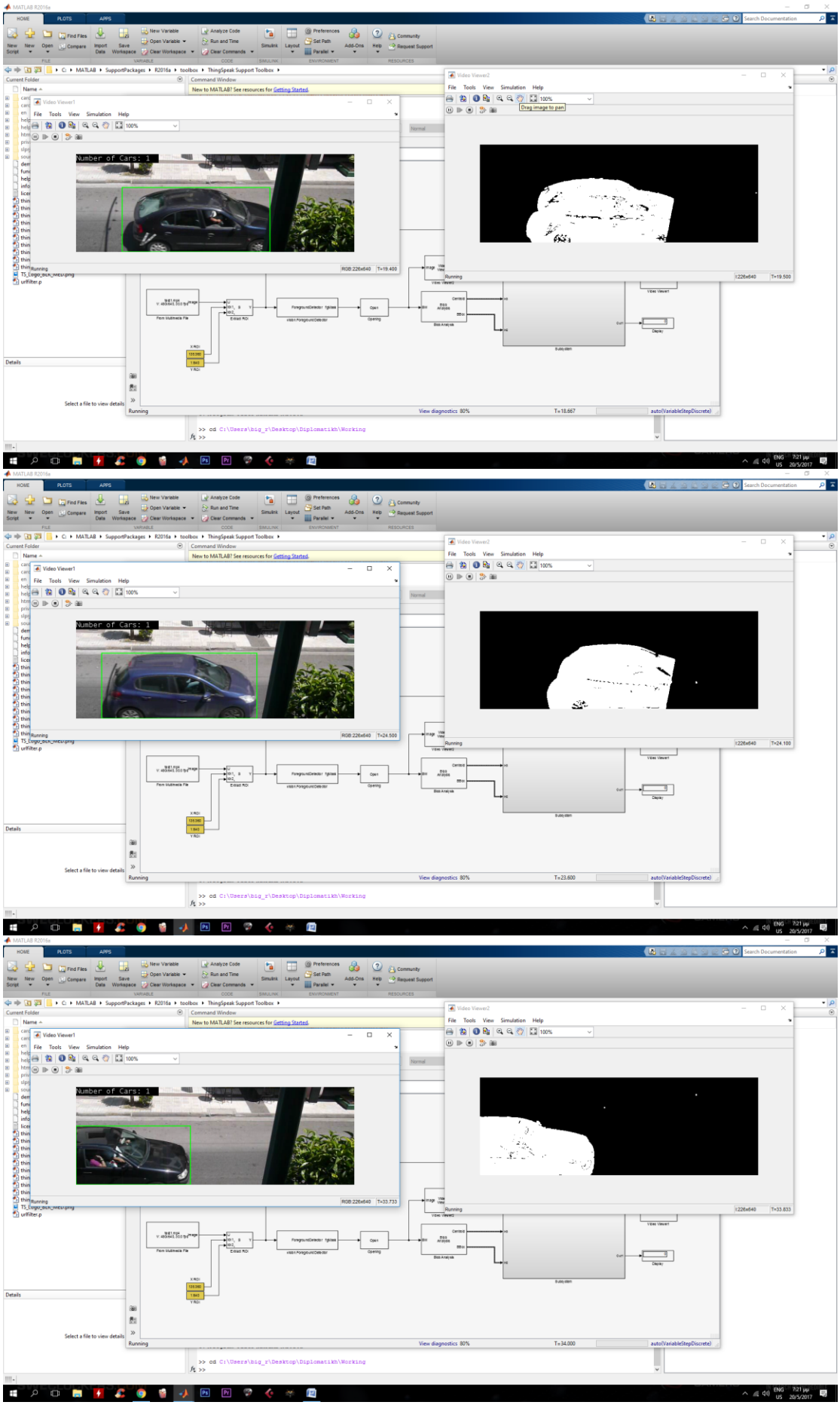


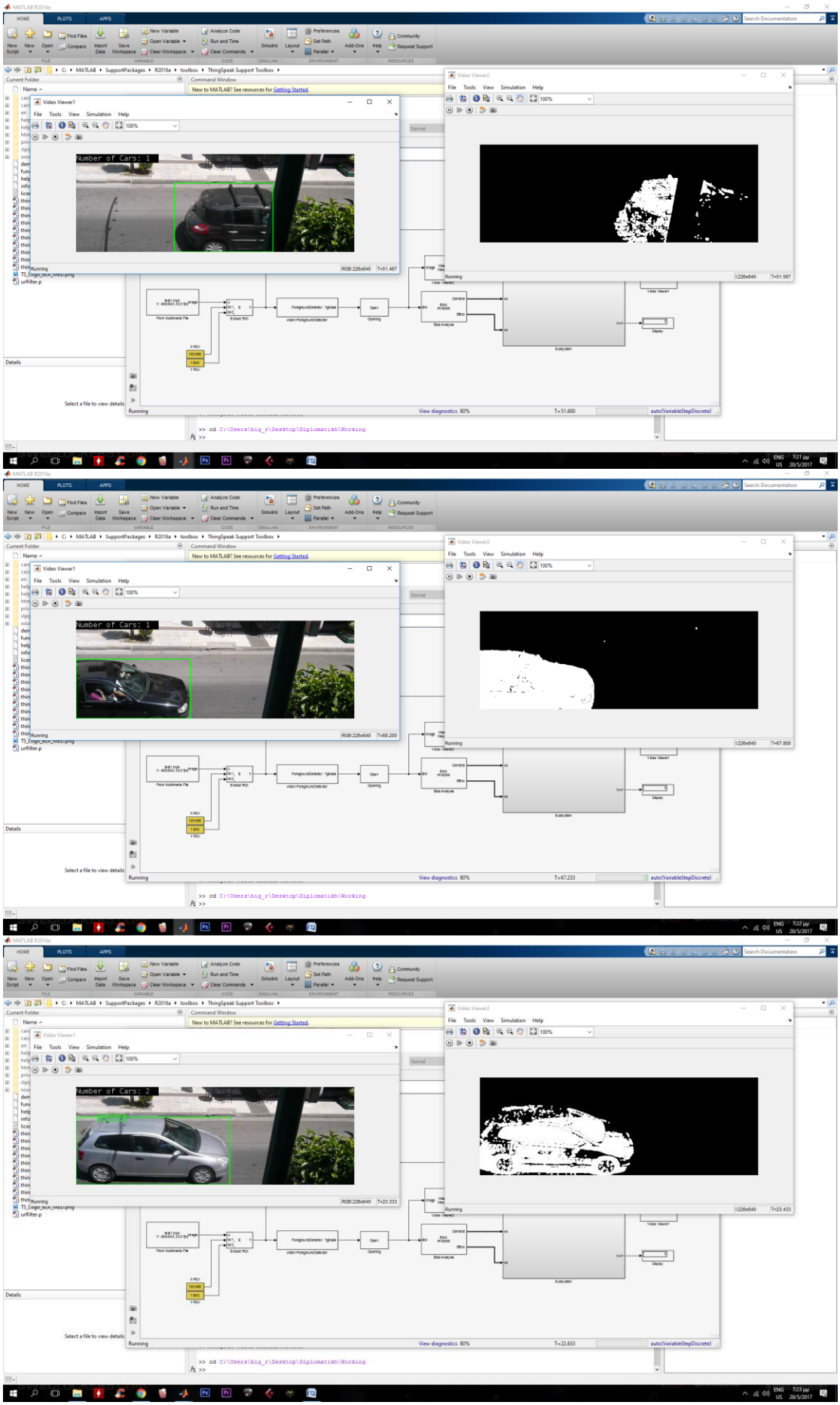
10.3 thingspeak channel analytics

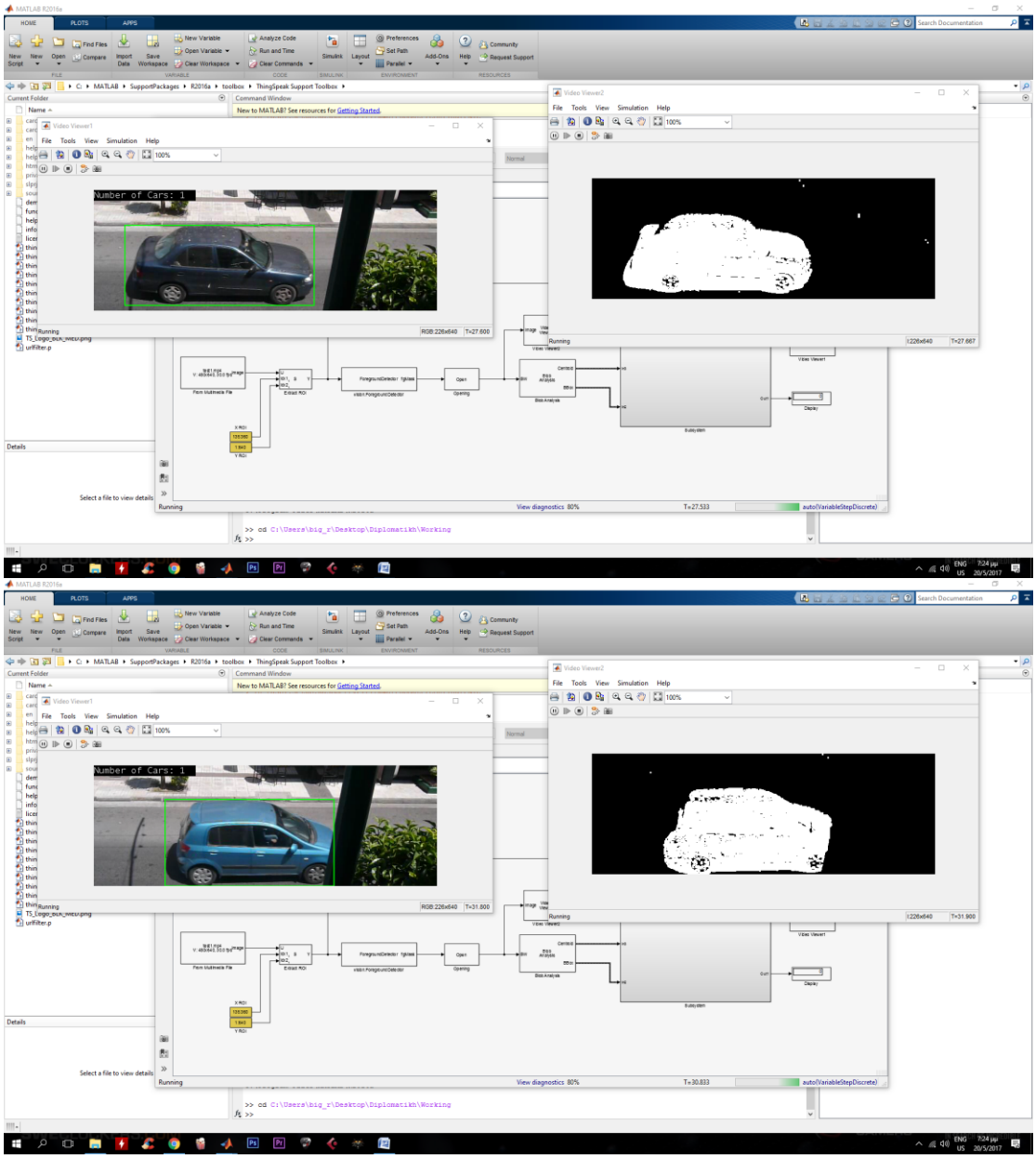




10.4-10,5 thingspeak channel analytics

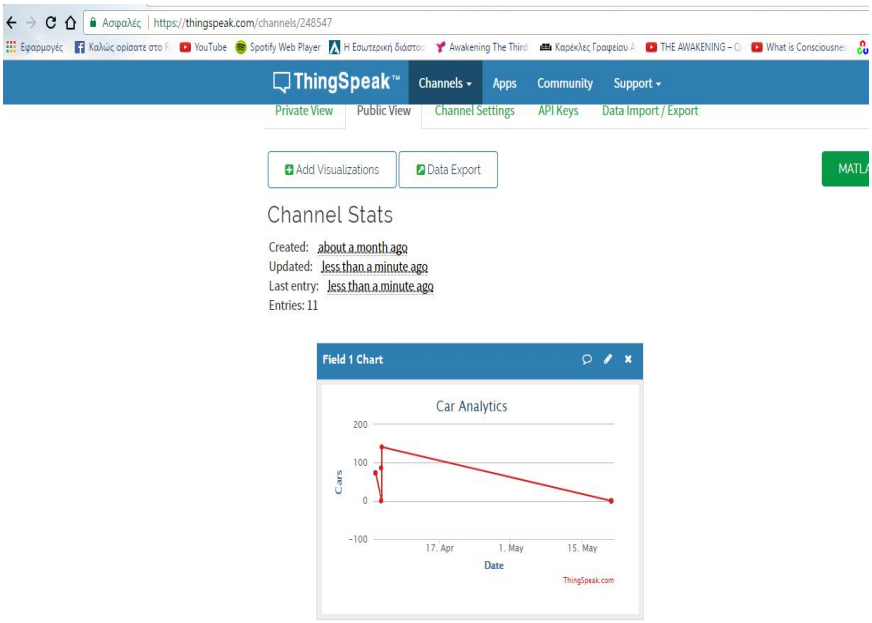

Number of Cars: 1
Number of Cars: 1
Number of Cars: 1


Number of Cars: 1
Number of Cars: 1
Number of Cars: 2



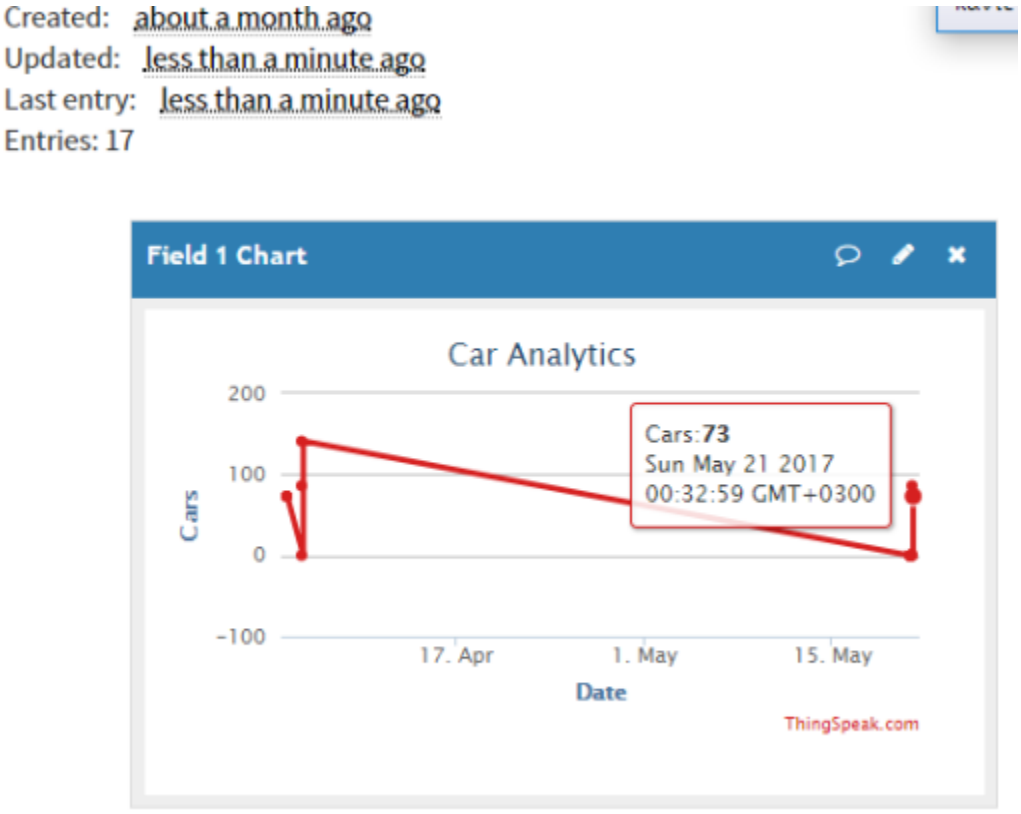
10.6-10.13 screenshot of the final results

Μετά από 3 μήνες συνεχόμενων ελέγχων της κίνησης καταλήξαμε σταδιακά σε αυτά τα διαγράμματα

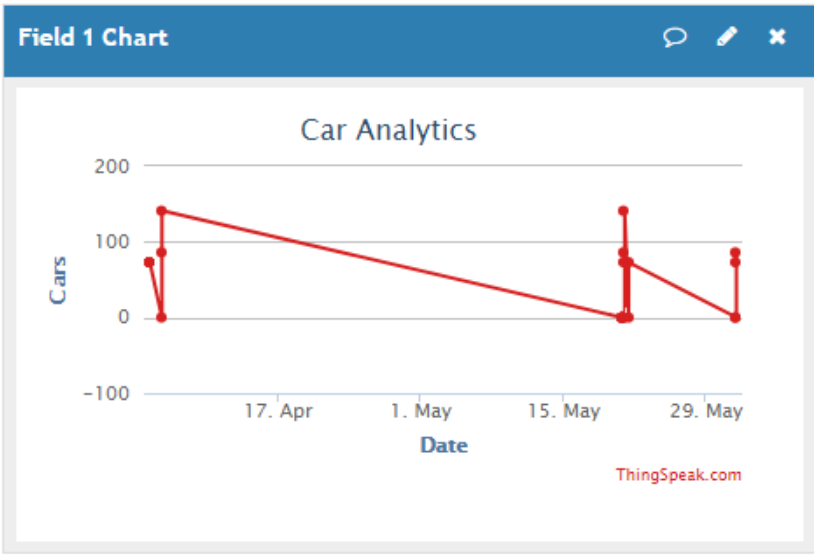


11.1 thingspeak channel analytics

Οι μετρήσεις γίνανε όλες μεταξύ 12:00-13:00 μ.μ σε ώρα ακμής της κίνησης στην κεντρική λεωφόρο Μακρυγιάννη του Μοσχάτου κατά τους μήνες Απρίλιο και Μάιο.

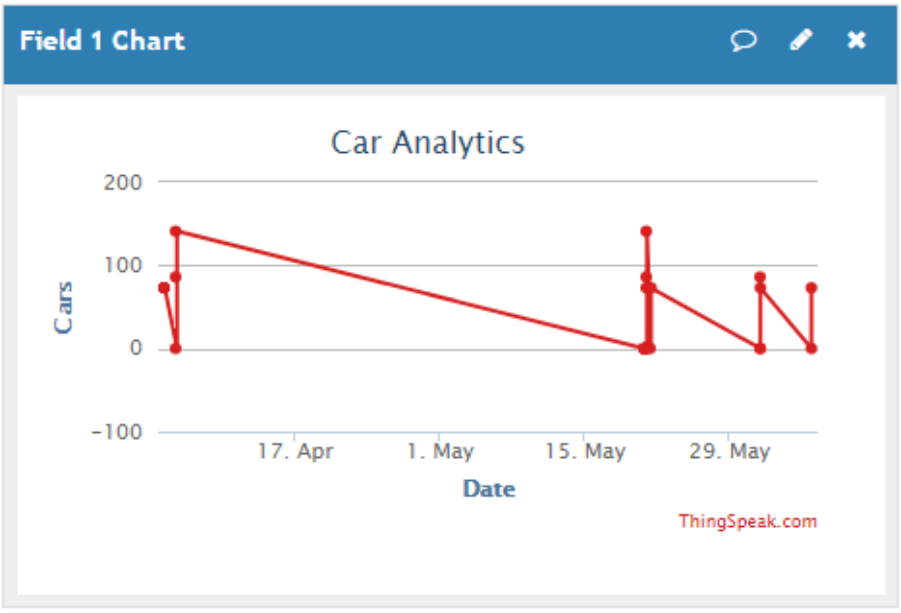


11.2 thingspeak channel analytics



11.3 thingspeak channel analytics

Created: 2 months ago
Updated: less than a minute ago
Last entry: less than a minute ago
Entries: 26



11.4 thingspeak channel analytics

Μέρος δεύτερο: Multi-object Motion Detection με python και Debian

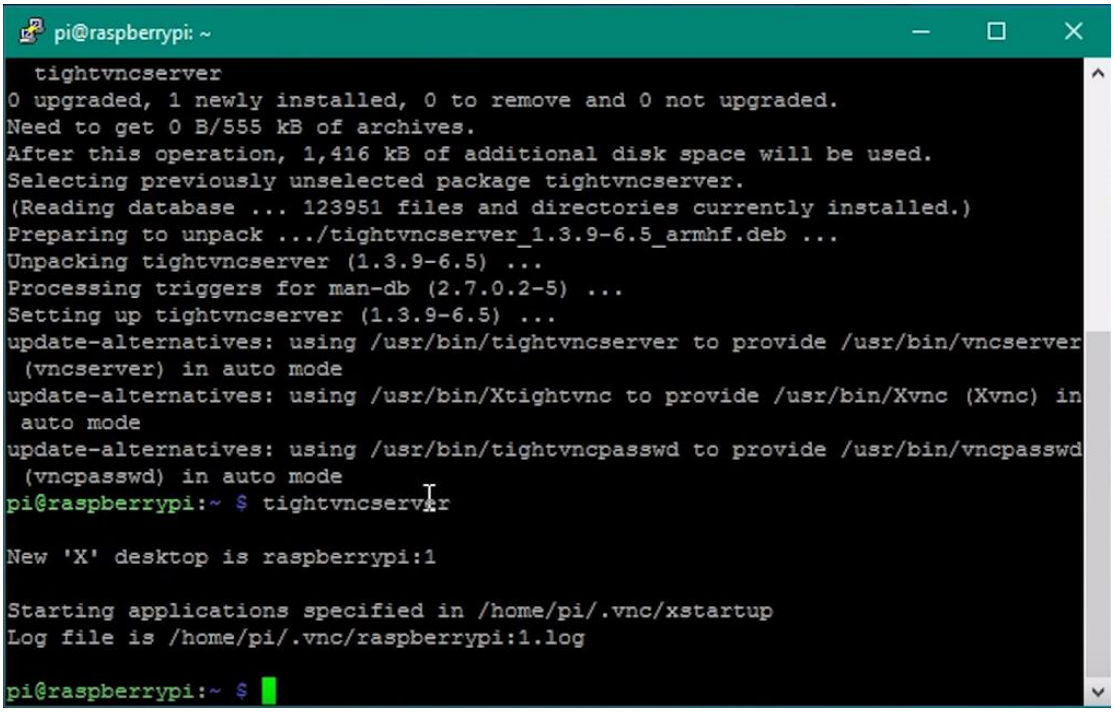
Σε αυτό το σημείο θα επαναλάβουμε τον ίδιο εργασιακό σκοπό αλλά αυτή τη φορά χωρίς matlab και simulink αλλά με Python και Linux.

Αρχικά εγκαθιστούμε στο Raspi μας την τελευταία έκδοση λειτουργικού του Debian. Όταν αυτό ολοκληρωθεί συνδέουμε στο modem μας με καλώδιο Ethernet το raspberry pi. Κατεβάζουμε τις εφαρμογές pUTTY και tightvncserver στον υπολογιστή των windows.

Ανοίγοντας την εφαρμογή pUTTY βάζουμε το i.p address του raspi και πατάμε connect για να συνδεθούμε. Ένας τερματικός θα ανοίξει και θα μας ζητήσει κωδικούς.Βάζουμε Pi για username και Raspberry για κωδικό. Στη συνέχεια πατάμε την ακόλουθη εντολή

>>sudo apt-get install tightvncserver

Αυτή η εντολή θα εγκαταστήσει τα απαραίτητα πακέτα που θα χρειαστεί η συσκευή μας για να συνδεθεί με το tightvnc viewer

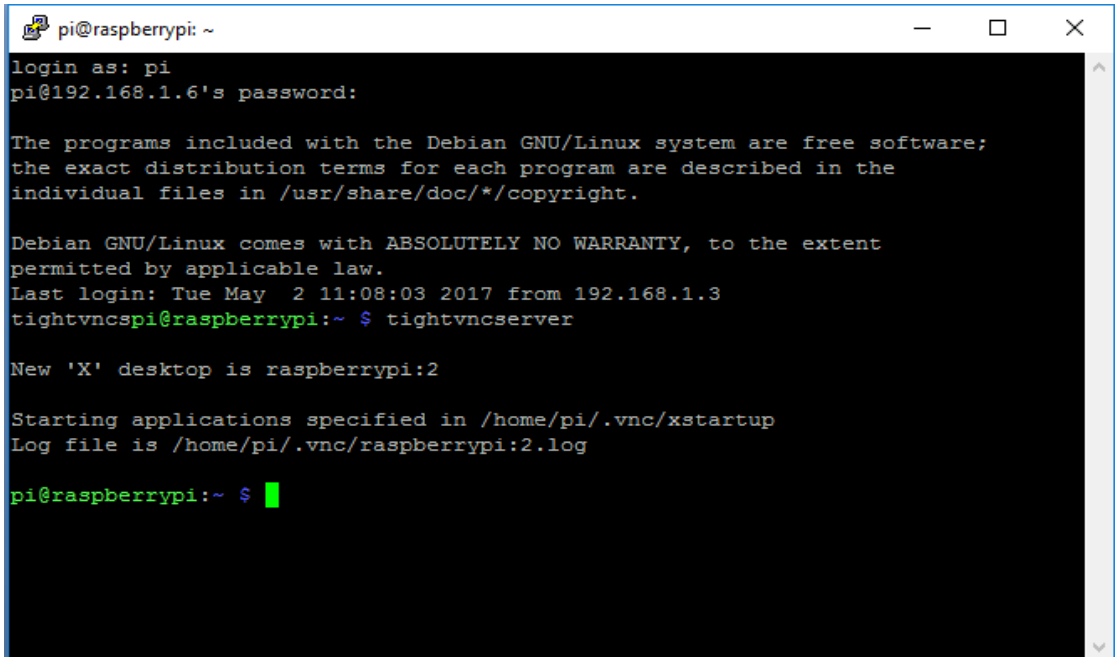


12.1 using putty to establish pi connection

Όταν ολοκληρωθεί η διαδικασία πατάμε την εντολή:

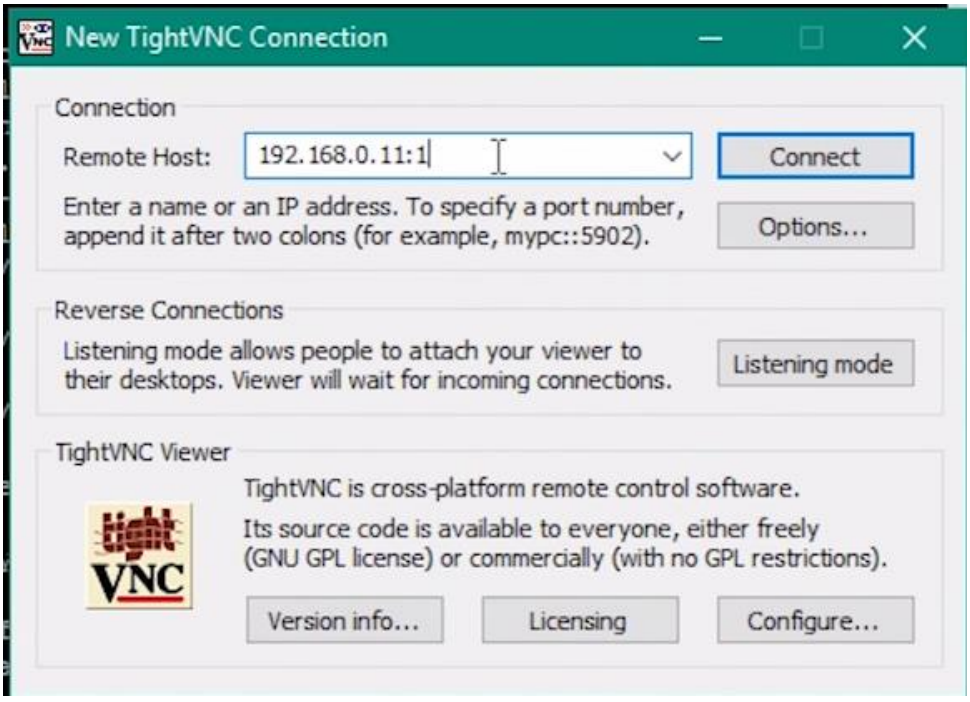
>>tightvncserver

Και παίρνουμε την ακόλουθη οθόνη:



12.2 using putty to establish pi connection

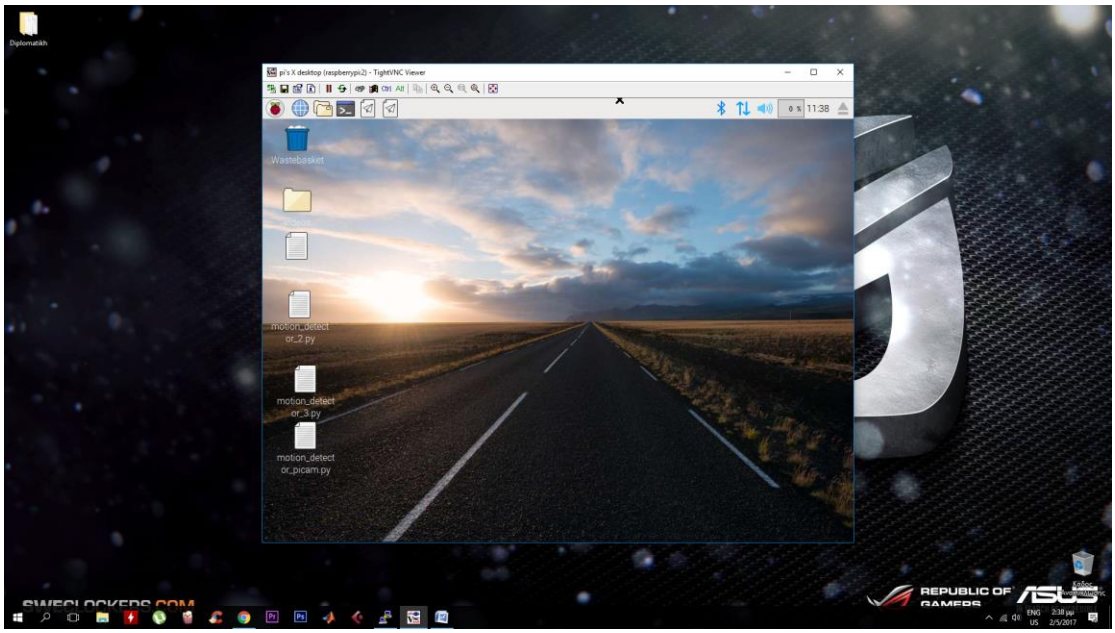
Είμαστε πλέον έτοιμοι να ανοίξουμε το tightvnc viewer και να κάνουμε σύνδεση remotely με την συσκευή μας.



12.3 using putty to establish pi connection

Πληκτρολογούμε την i.p του raspi ακολουθόυμενη από : και τον αριθμό του Destop που μας εμφάνισε στον τερματικό μετά την εκτέλεση της εντολής **tightvncserver.**

Πατώντας connect έχουμε πλέον συνδεθεί στην συσκευή μας ασύρματα μέσω modem.



12.4 Final Results

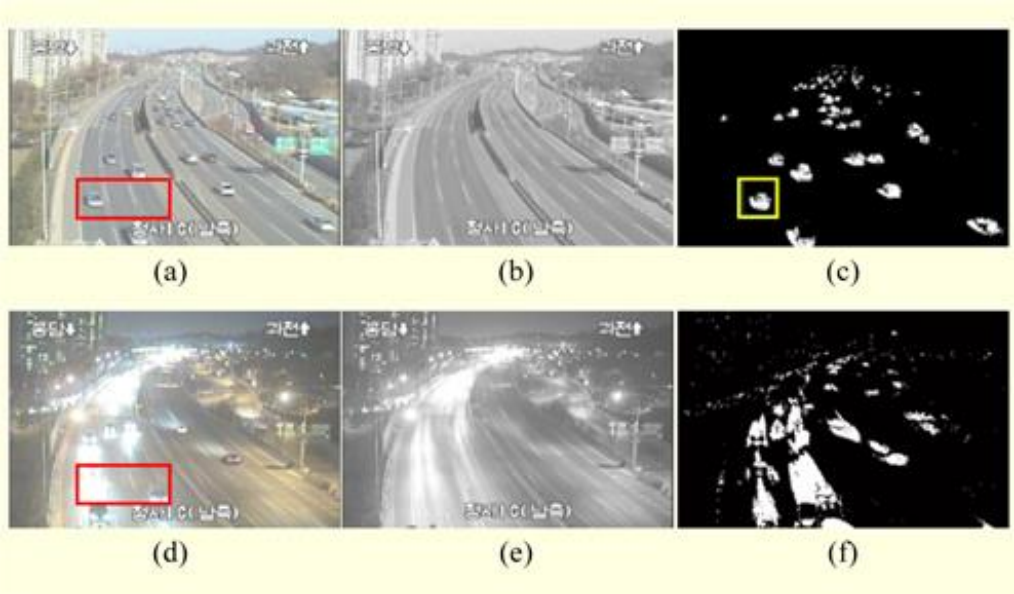
Foreground and Background

Οι αλγόριθμοι ανίχνευσης κίνησης βασίζονται στην υπόθεση ότι το background που παρακολουθούμε στο παρασκήνιο είναι στατικό και ανιχνεύεται κίνηση όταν υπάρχει ουσιώδης αλλαγή σε αυτό. Στην πράξη αυτό δεν είναι εντελώς σωστό γιατί υπάρχουν και άλλοι παράγοντες που επηρεάζουν το background όπως φωτισμός, σκίαση και άλλα efe. Η κάμερα πρέπει να είναι στατική ώστε να αποφθεχθούν όλες αυτές οι αρνητικές συγκυρίες.

Μερικοί μέθοδοι δίνουν διαχωρισμό των frames στο foreground και το background έτσι ώστε να γίνει διαφοροποίηση μεταξύ κίνησης και άλλων effe που ίσως προκαλέσουν δυσλειτουργία στον αλγόριθμο. Οι αλγόριθμοι αυτοί χρησιμοποιούν Gaussian Mixture Model ή της Bayesian method παρόλα αυτά όμως αυτές οι μέθοδοι κοστίζουν μεγάλη ποσότητα υπολογιστικής ισχύς.

Αν η κάμερα είναι σε σταθερό σημείο τότε μια απλή προσέγγιση είναι να αφαιρεθεί το background του πρώτο frame από όλα τα μεταγενέστερα frames.

Ένα πιο εξελιγμένο σχηματικό θα ήταν να γίνει ανανέωση του background frame όταν δεν υπάρχει καθόλου κίνηση και ταυτόχρονα όταν το background frame reference είναι μεγαλύτερο από το προκαθορισμένο threshold.



13.1 Foreground-background detection

$I(i, j, t)$ $I_b(i, j, t)$

Βασική προσέγγιση για αφαίρεση της foreground μάσκας:
$|I(i, j, t) - I_b(i, j, t)| > T$

-Το background frame είναι το πρώτο frame στην διαδικασία σύλληψης

T=25

T=100

T=200

13.2 sampling time of detection

-Το background είναι το μέσο των προηγούμενων n frames

$$I_b(i,j,t) = \frac{1}{n}\sum_{k=1}^{n} I(i,j,t-k)$$

I_b for n=10

Foreground mask

I_b for n=50

Foreground mask

13.3 sampling results of detection

-Αντί για το μέσο θα μπορούσε να ήταν ο τρέχον μέσος όρος

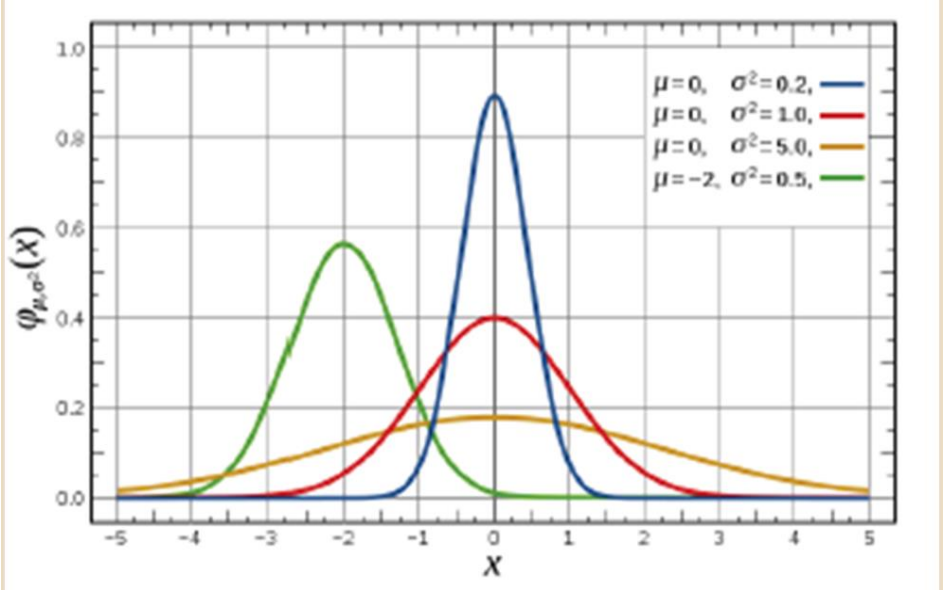
$$I_b(i,j,t) = (1-a)I_b(i,j,t-1) + aI(i,j,t)$$

python motion_detector.py --video videos/example_01.mp4

Mixture of Gaussians (MoG) or Gaussian Mixture Models (GMM)

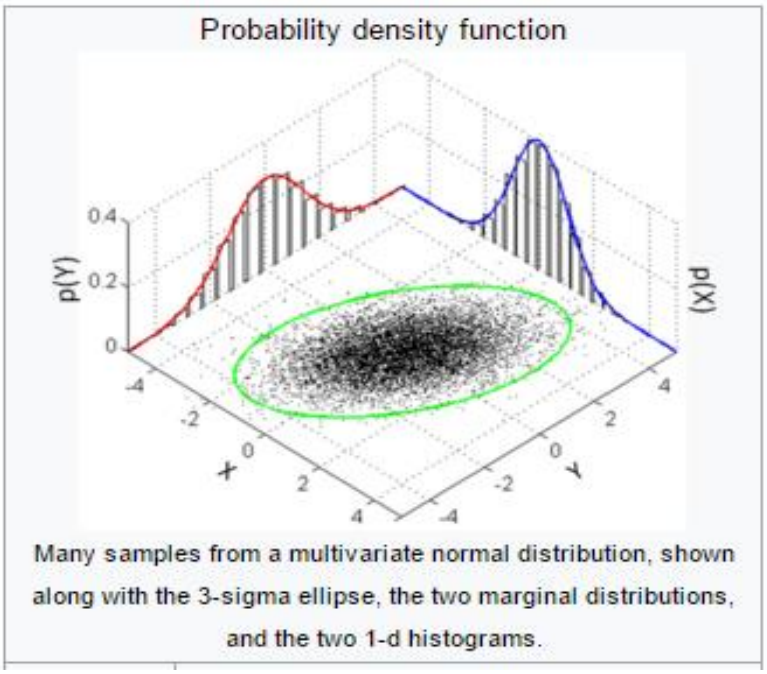
Normal or Gaussian Distribution – Univariate – One dimensional

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \qquad -\infty < x < \infty$$



14.1 Normal or Gaussian Distribution – Multivariate – Multidimensional

$$f(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$



14.2 3d Gaussian Distribution

p = dimensions

μ = mean

Σ = Covariance of random variables

Ο αλγόριθμος διαφοροποίησης με Gaussian Mixture-based Background/foreground βγήκε για πρώτη φορά σε paper από τον P. Kadew TraKuPong και R. Bowden το 2001. Μοντελοποιεί κάθε pixel του background από ένα μείγμα Gaussian διανομών (K=3 με 5). Η σημασία των μειγμάτων αντιπροσωπεύουν τις χρονικές αναλογίες των χρωμάτων που παραμένουν στο παρασκήνιο. Τα πιθανά background χρώματα είναι αυτά που παραμένουν περισσότερο και είναι πιο στατικά

Ο αλγόριθμος του GMM

Σε οποιαδήποτε χρονική στιγμή t είναι γνωστό το ιστορικό των τιμών αυτού του pixel. Αυτός ίσως θεωρηθεί ως ένα προσαρμοστικό μείγμα Gaussians γιατί υπάρχει αλλαγή σε κάθε χρονική στιγμή και πρέπει αν ακολουθεί την αλλαγή των συνθηκών φωτισμού

Αυτό το ιστορικό έχει μοντελοποιηθεί από ένα μείγμα Κ Γκαουσιανών διανομών. Για παράδειγμα ένα pixel στο (0,0) :

$$\{X_1, \ldots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \mathcal{N}(\mathbf{X}_t | \mu_{i,t}, \mathbf{\Sigma}_{i,t})$$

Όπου Ν είναι ο αριθμός της πολυπαραγοντικής τιμής διανομής, w η βαρύτητα κάθε τιμής.

Σε κάθε επανάληψη τα μείγματα αξιολογούνται και αν ένα pixel ταιριάζει σε κπαοιο από τα Gaussians πιθανότατα ανήκει στο background.
Ο μέσος όρος και η διακύμανση αυτού του Gaussian ενημερώνονται λαμβάνοντας υπόψη τη νέα τιμή του εικονοστοιχείου

Τα εικονοστοιχεία που δεν είναι φόντο, ταξινομούνται ως pixel προσκηνίου. Τα εικονοστοιχεία του προσκηνίου ομαδοποιούνται με ανάλυση συνδεδεμένου στοιχείου.

Στο OpenCV υπάρχει μια διαδικασία createBacgroundSubtractorMOG2() ώστε να δημιουργεί ένα αντικείμενο background. Έχει κάποιες προεραιτικές παραμέτρους όπως το μήκος του ιστορικού, ο αριθμός των Gaussian μειγμάτων, Thresholds και άλλα. Όλα έχουν κάποιες default τιμές. Έπειτα μέσα στον βρόχο του video χρησιμοποιεί την backgroundsubtractor.apply() μέθοδο για να πάρει την foreground mask.

```
import numpy as np

import cv2

cap = cv2.VideoCapture('vtest.avi')

fgbg = cv2.createBackgroundSubtractorMOG()

while(1):

        ret, frame = cap.read()

        fgmask = fgbg.apply(frame)

        cv2.imshow('frame',fgmask)

        k = cv2.waitKey(30) & 0xff

        if k == 27:

        break

cap.release()

cv2.destroyAllWindows()
```

Καταμέτρηση αυτοκινήτων με ανίχνευση Foreground Detection (με Python)

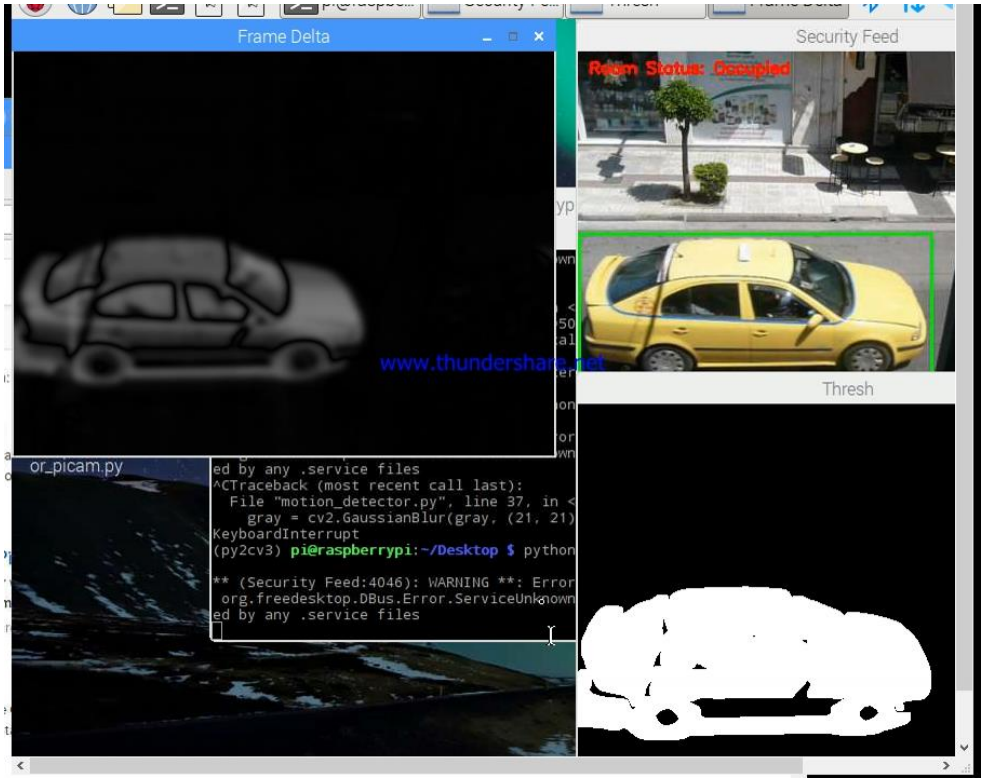
Στο παρακάτω πείραμα θα εκτελέσουμε τα ίδια βήματα με πρίν για την ανίχνευση οχημάτων. Ο κώδικας μας θα γίνει σε Python και compiled από τον τερματικό του Debian ώπου κατά την εκτέλεση εν τέλει θα χρησιμοποιεί foreground detection για την ανίχνευση οχημάτων σε έναν δρόμο και θα υποδεικνύει τον αριθμό τον αμαξιών που βρίσκονται στο οπτικό πεδίο της κάμερας κάθε φορά. Ο κώδικας μας σε Python που θα αναλύσουμε αργότερα είναι ο παρακάτω:

```
# python motion_detector.py --video videos/example_01.mp4

# εισαγωγή των πακέτων που θα χρειαστούμε κυρίως τα opencv2

import argparse

import datetime

import imutils

import time

import cv2


# κατασκευή αναλυτή παραμέτρων και ανάλυση των 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"])
```

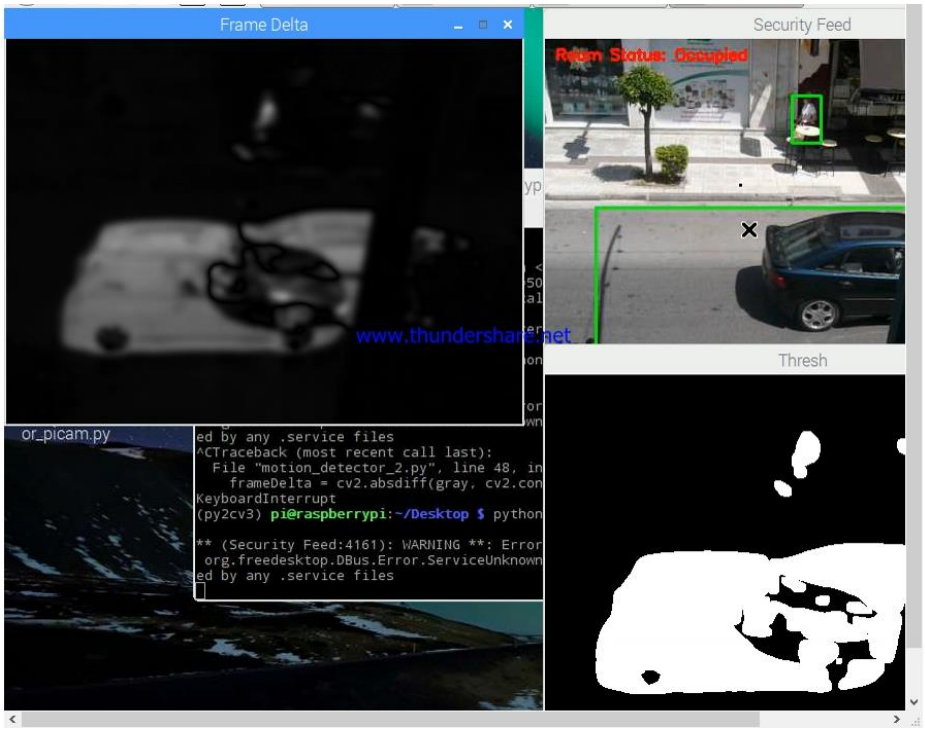
```
# αρχικοποίηση πρώτου frame του video

firstFrame = None


        # επανάληψη στα frames του video

        while True:

                # εύρεση και αρχικοποίηση του πρώτου frame σαν occupied/unocupied

                # text

                (grabbed, frame) = camera.read()

                text = "Unoccupied"


                # αν δεν μπορεί να αρχικοποιηθεί το πρώτο frame φτάσαμε στο τέλος του video


                if not grabbed:

                        break


                # αλλαγή μεγέθους του frame,μετατροπή σε grayscale και εφαρμογή blur

                frame = imutils.resize(frame, width=500)

                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # convert from RGB to grayscale

                gray = cv2.GaussianBlur(gray, (21, 21), 0) #blur


                # αν δεν υπάρχει το πρώτο frame αρχικοποίησε το

                if firstFrame is None:

                        firstFrame = gray

                        continue


                # υπολογισμός της απόλυτης διαφοράς μεταξύ του τρέχοντος και του αρχικού
                frame

                frameDelta = cv2.absdiff(firstFrame, gray) #video background isolation from objects
                #of interest
```

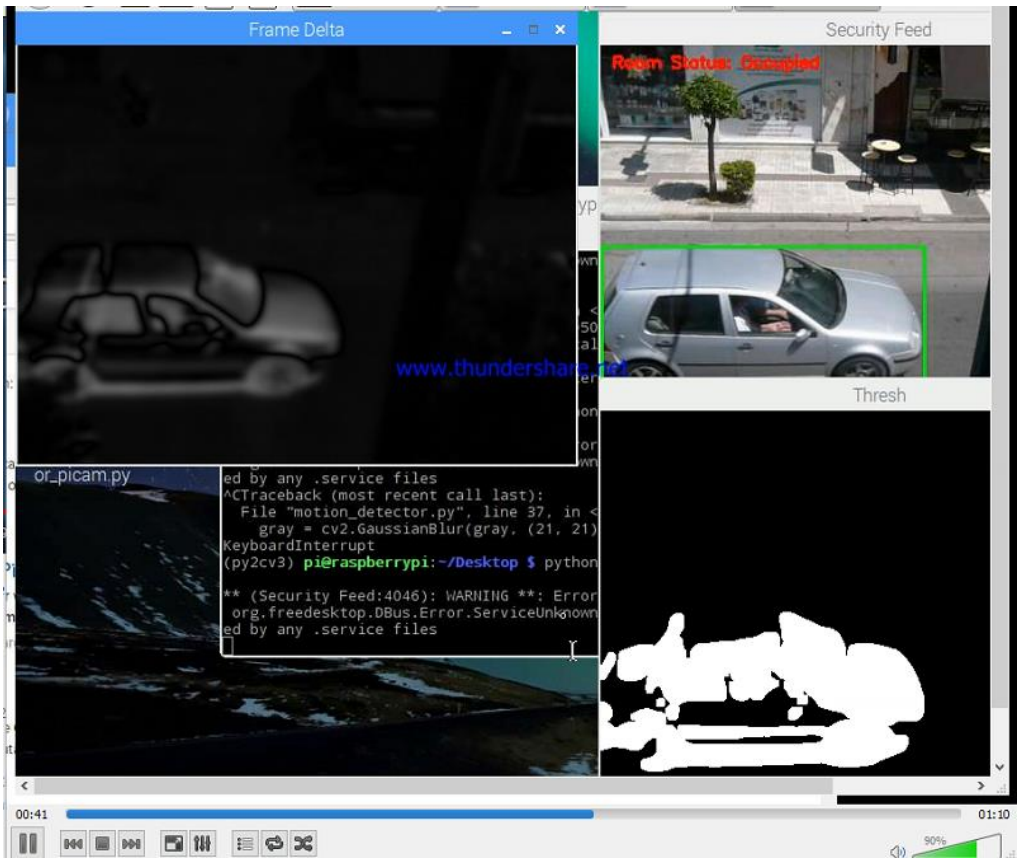
```
#Αν η τιμή του pixel είναι μεγαλύτερη από την τιμή του threshold,γίνεται εκχώρηση μιας τιμής αλλιώς ορίζεται μια άλλη τιμή. Το πρώτο argument είναι η εικόνα πηγής (κλίμακα γκρι) και το δεύτερο όρισμα είναι η τιμή threshold για την ταξινόμηση των τιμών pixel. MaxVal είναι η τιμή που πρέπει να δωθεί εάν η τιμή του pixel είναι μεγαλύτερη ή μικρότερη από την τιμή threshold.

thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]


        # διαστολή της εικόνας που έχει υποστεί threshold για να γεμίσουν τα κενά και να βρεθούν τα περιγράμματα

        thresh = cv2.dilate(thresh, None, iterations=2) #find colors

        (_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,

        cv2.CHAIN_APPROX_SIMPLE) # εύρεση περιγραμμάτων


        # επανάληψη για τα διαγράμματα

        for c in cnts:

                # αγνόηση πολύ μικρών διαγραμμάτων

                if cv2.contourArea(c) < args["min_area"]:

                        continue


                #Υπολογισμός των ορίων για τα περιγράμματα, σχεδιασμός στο frame και ανανέωση του 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

# εκκαθάριση της κάμερας και κλείσιμο όλων των παραθύρων
camera.release()
cv2.destroyAllWindows()
```



14.1 results of foreground detection on p ython

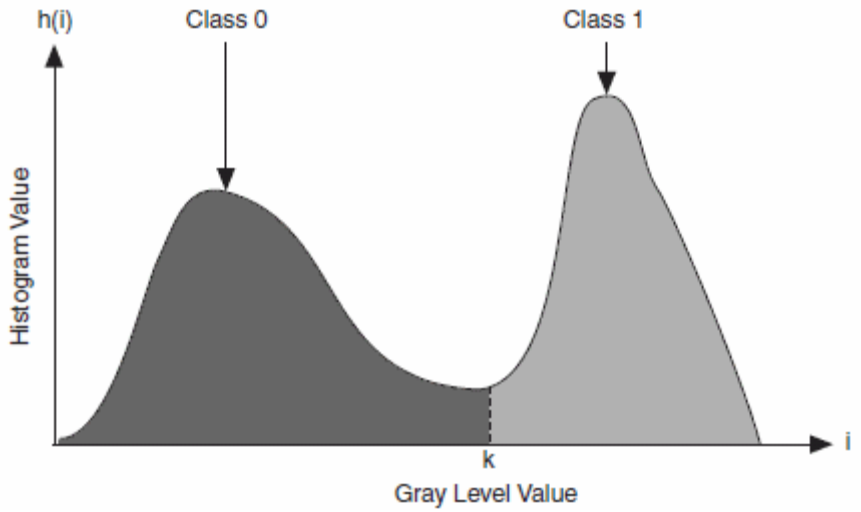


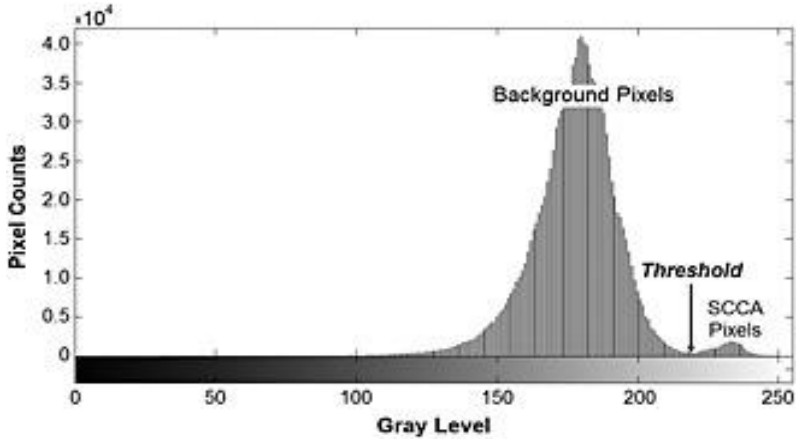


14.2-14.3 results of foreground detection on python

Όπως βλέπουμε ο παρακάτω αλγόριθμος παίρνει ένα video από μια πηγή και εφαρμόζει foreground detection.μετατρέπει το background σε grayscale,λαμβάνει το αρχικό pixel και στη συνέχεια συγκρίνει τα pixels των επόμενων frames με το αρχικό. Αν η τιμή του pixel είναι μεγαλύτερη από την τιμή του threshold,γίνεται εκχώρηση μιας τιμής αλλιώς ορίζεται μια άλλη τιμή. Το πρώτο argument είναι η εικόνα πηγής (κλίμακα γκρι) και το δεύτερο όρισμα είναι η τιμή threshold για την ταξινόμηση των τιμών pixel. MaxVal είναι η τιμή που πρέπει να δοθεί εάν η τιμή του pixel είναι μεγαλύτερη ή μικρότερη από την τιμή threshold . Στη συνέχεια υπολογίζει και σχεδιάζει τα πράσινα περιγράμματα και τα όρια αυτών.

Στο παρακάτω διάγραμμα φαίνεται ο διαχωρισμός των pixels ανάλογα με το συγκεκριμένο threshold Που του έχουμε δώσει. Τα pixels με τιμή κάτω από threshold ταξινομούνται στην Class 0 ενώ τα pixels με τιμή ίση ή ανώτερη του threshold ταξινομούνται στην Class1.





14.4-14.5 Gray Level Value Diagram in Moving Average Background

Καταμέτρηση αυτοκινήτων με ανίχνευση Moving Average Background (Foreground Detection σε Python)

Όπως είδαμε πρίν καταφέραμε να κάνουμε ανίχνευση των αυτοκινήτν με τον τρόπο Background Detection αλλά όπως και στο Matlab έτσι και εδώ αυτός ο τρόπος δεν είναι ο βέλτιστος ή έστω αρκετά ικανοποιητικός για την συλλογή των δεδομένων μας. Παρακάτω θα δούμε την καταμέτρηση και ανίχνευση αντικειμένων με την εφαρμογή του Moving Average Background γνωστό και ως Foreground Detection ή Background Subtraction.

Για να ανακεφαλαιώσουμε πριν προχωρήσουμε στον κώδικα, η τεχνική foreground detection εφαρμόζεται στην επεξεργασία εικόνας και στις εφαρμογές computer vision ώπου το πρώτο πλάνο της εικόνας εξάγεται για περαιτέρω επεξεργασία. Στο πρώτο πλάνο επίσης βρίσκονται τα πεδία του ενδιαφέροντός μας (αυτοκίνητα). Μετά από το στάδιο της επεξεργασίας της εικόνας (αφαίρεση θορύβου,morphological opening κτλ) χρειάζεται να κάνουμε ταυτοποίηση του αντικειμένου.

Χρησιμοποιώντας απλή μαθηματική λογική μπορούμε να τμηματοποιήσουμε τα αντικείμενα μας χρησιομοποιότνας την τεχνική αφαίρεσης παρασκηνίου δηλαδη για κάθε pixel στην I(t) παίρνουμε την τιμή του pixel που υποδηλώνει το P[I(t)] και το αφαιρούμε με το αντίστοιχο pixel στην ίδια θέση στο background που έχει υποδηλωθεί σαν P[B].

$$P[F(t)]=P[I(t)] – P[B]$$

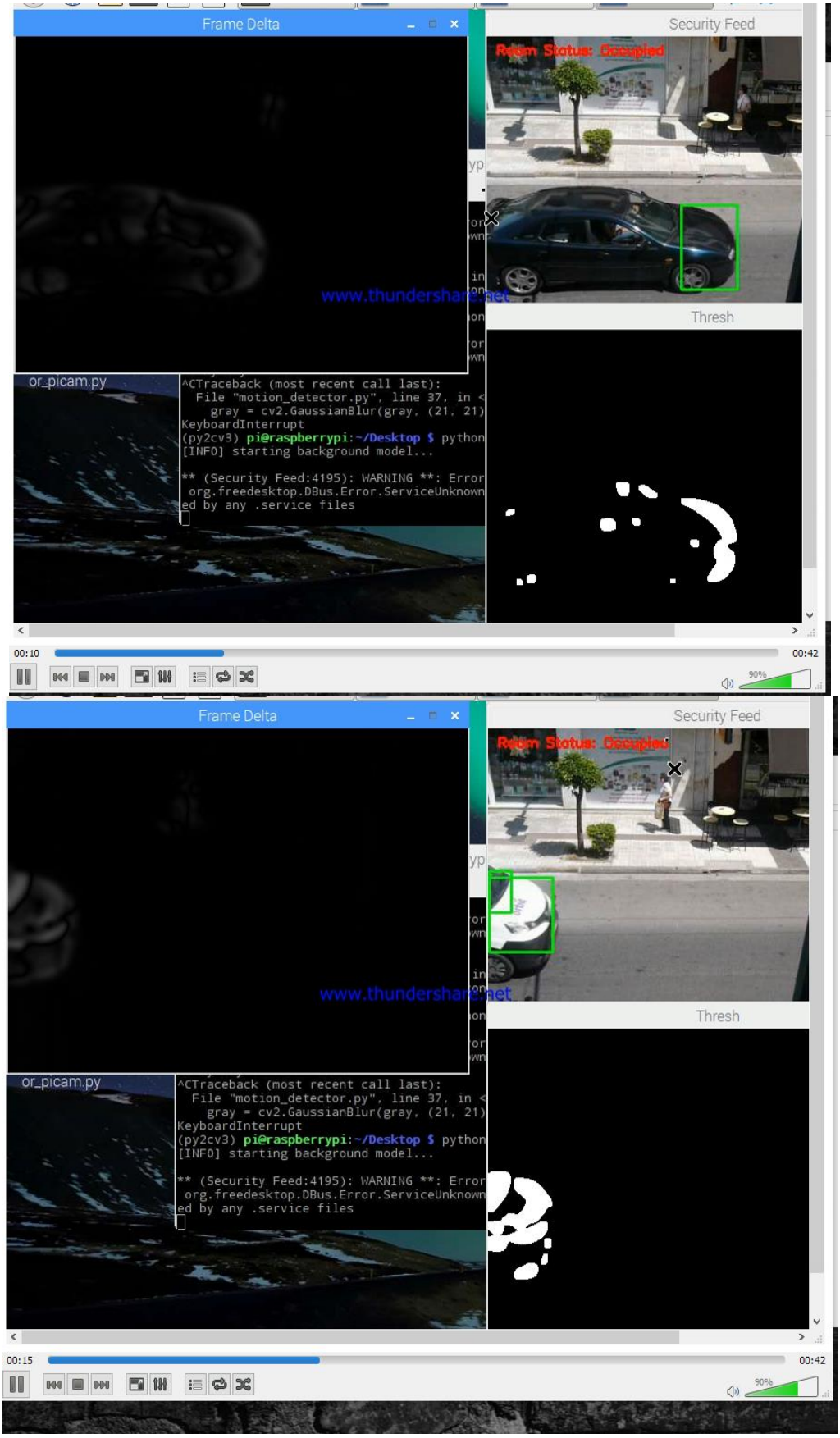
Το παρασκήνιο θεωρούμε ότι βρίσκεται συνέχεια σε έναν χρόνο t. Αυτή η διαφορά στην εικόνα θα δείξει πιο έντονα τις τοποθεσίς των pixels που έχουν αλλάξει μεταξύ των δύο συγκρινόμενων frames.

Ο κώδικας σε Python είναι ο ακόλουθος:

```
# python motion_detector_2.py --video videos/example_2.mp4

# 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


#Συσσώρευση του μέσου όρου μεταξύ του τρέχοντος frame και των προηγούμενων frames.Έπειτα υπολόγισε την διαφορά μεταξύ του τρέχοντος frame και του τρέχοντος μέσου όρου

cv2.accumulateWeighted(gray, avg, 0.5)

frameDelta = cv2.absdiff(gray, cv2.convertScaleAbs(avg))


# βάλε threshold στην τρέχουσα εικόνα, διαστολή του threshold ώστε να γεμίσει τα κενά της εικόνας, και μετά βρές τα περιγράμματα της thresholded εικόνας

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
```

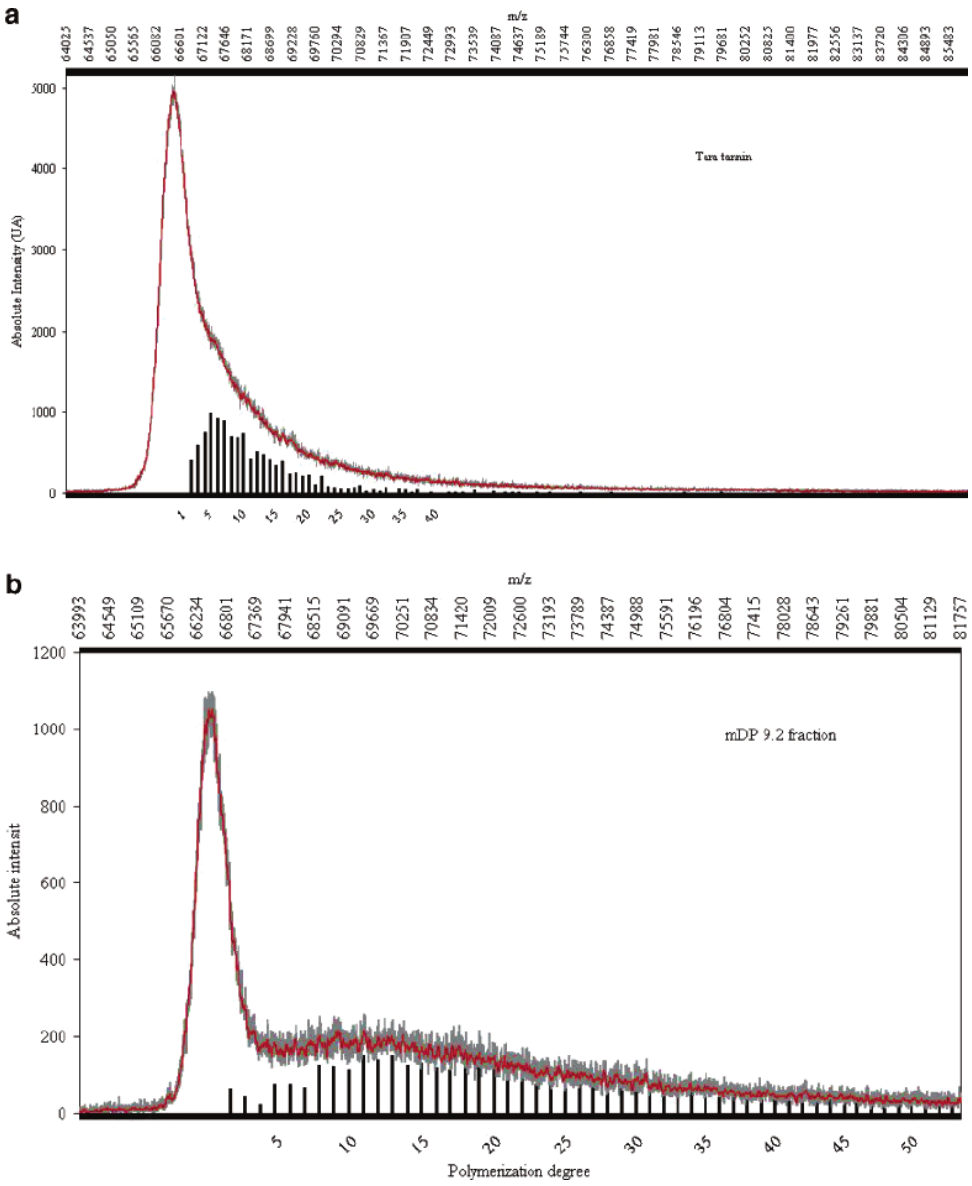
```
# compute the bounding box for the contour, draw it on the frame, and update the text
(x, y, w, h) = cv2.boundingRect(c)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
text = "Occupied"
# draw the text and timestamp on the frame
cv2.putText(frame, "Room Status: {}".format(text), (10, 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
```



15.1 results moving average background detection on python

Σε αυτόν τον κώδικα ακολουθούμε μια παρόμοια διαδικασία με πριν αλλά αυτή την φορά αντί να συγκρίνουμε τα frames με το πρώτο αρχικοποιημένο frame βρίσκουμε τον μέσο όρο μεταξύ του τρέχοντος frame και των προηγούμενων frames και υπολογίζουμε την διαφορά μεταξύ του τρέχοντος frame και του τρέχοντος μέσου όρου. Αυτή η τεχνική Moving Average είναι η βελτιστοποιημένη τεχνική της πρώτης. Τα αποτελέσματα είναι σαφώς πιο ικανοποιητικά αλλά όχι ολοκληρωτικά.

Στο παρακάτω διάγραμμα βλέπουμε μια χαρακτηριστική καμπύλη moving average background. Παρατηρούμε ότι οι τιμές στο σχεδιάγραμμα b έχουν αλλάξει με βάση των τιμών που προέκυψαν από τον υπολογισμό του μέσου όρου των pixels



15.2-15.3 Moving Average background graph

Καταμέτρηση αυτοκινήτων με Gaussian Mixture Model (σε Python)

Τέλος η τελευταία μας τεχνική είναι η γνωστή GMM ή Gaussian Mixture Model ώπου θεωρείτε και η βέλτιστη.

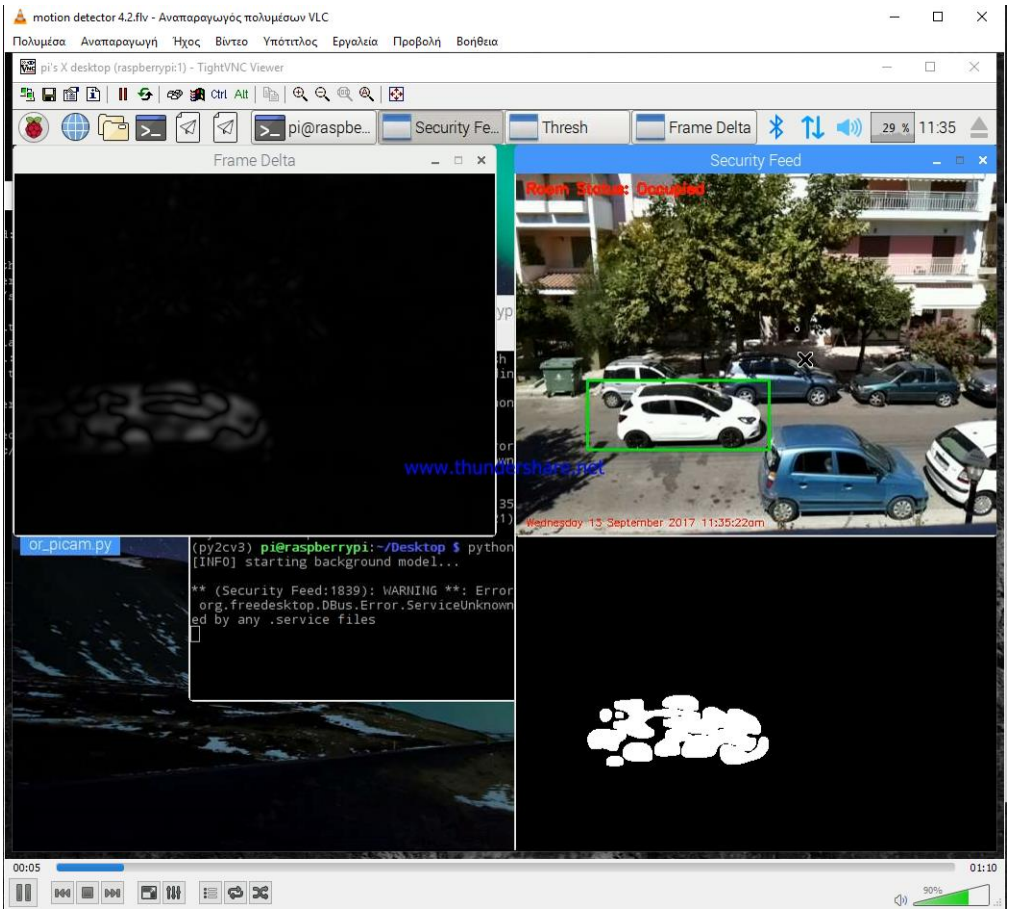
Ο κώδικας σε Python που αντιστοιχεί σε αυτή την μέθοδο είναι η εξής:

```
# python motion_detector_3.py --video videos/video_01.avi

# 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


# εφαρμογή GMM

fgmask=fgbg.apply(frame)


# κατάργηση θορύβου με την μέθοδο διάβρωσης-διαστολής και μετά βρες τα πλαίσια

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]
```

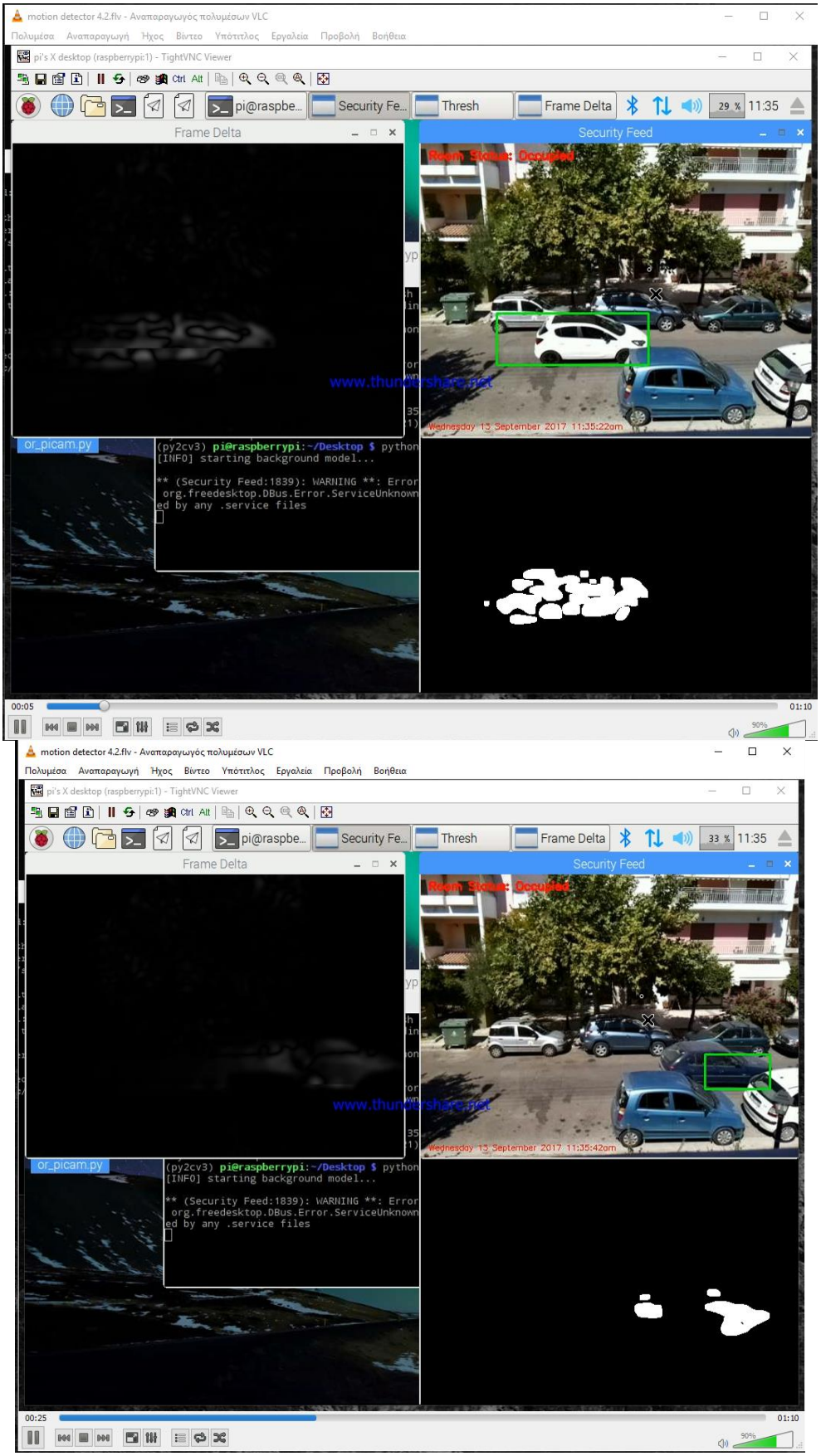
```
if cv2.contourArea(cnt) > args["min_area"]:
# compute the bounding box for the contour, draw it on the frame,and update the text
(x, y, w, h) = cv2.boundingRect(cnt)
cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
text = "Occupied"

# 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()
```

Όπως παρατηρούμε η εφαρμογή της τεχνικής Gaussian Mixture Model είναι η βέλτιστη από όλες τις προηγούμενες.



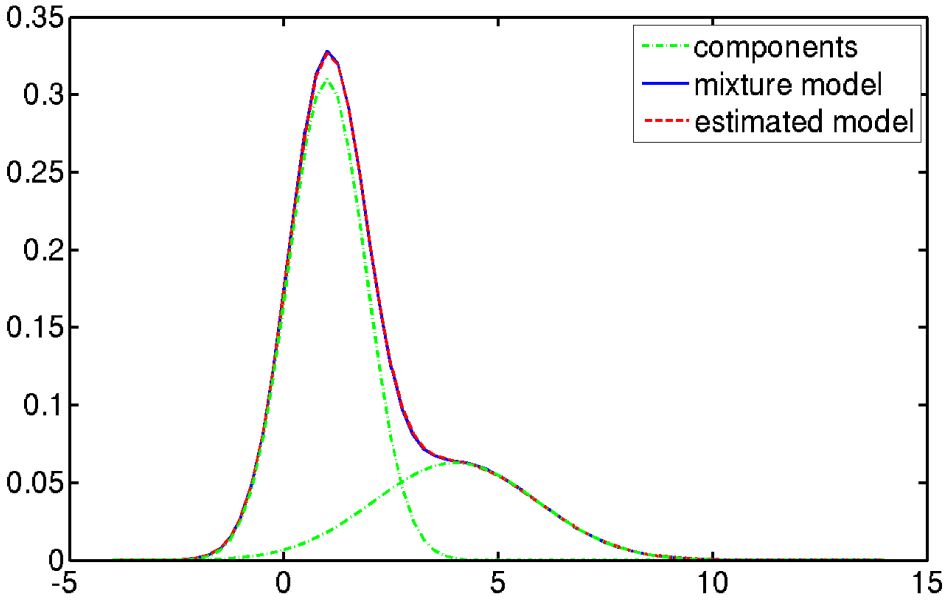
16.1 results of Gaussian mixture model in Python

Αυτό το παράδειγμα έτρεξε ζωντανά κάνοντας sampling από το μπαλκόνι του σπιτιού σε πραγματικό χρόνο καταμέτρησης οχημάτων στις 2 το μεσημέρι ώρα Ελλάδας.



16.2-16.3 results of Gaussian mixture model in Python

Παρακάτω δίδεται μια γραφική παράσταση επεξεργασίας εικόνας pixels που έχουν υποστεί Gaussian mixture Model



16.4 Gaussian Mixture Model Pixel Analysis

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. https://www.mathworks.com/help/vision/examples/detecting-cars-using-gaussian-mixture-models.html - Detecting Cars Using Gaussian Mixture Models

2. https://www.mathworks.com/help/vision/examples/tracking-cars-using-foreground-detection.html - Detecting Moving Objects with foreground detection

3. https://www.mathworks.com/help/supportpkg/raspberrypiio/examples/working-with-raspberry-pi-camera-board.html - Working with Raspi Camera

4. https://www.mathworks.com/help/vision/examples/motion-based-multiple-object-tracking.html - Multi Object Detection

5. https://www.youtube.com/watch?v=HP0x5iH8g6k – Using matlab and Raspberry Pi camera board

6. https://github.com/opencv/opencv/tree/master/include/opencv2 - Computer vision open cv2 python

7. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html - Image analyzing with open cv

8. http://www.pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/ - Image difference with opencv and python

9. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html - Getting started with videos and python

10. https://www.youtube.com/watch?v=xkvahkqnN1c – Vehicle video counter and classification

11. https://realpython.com/blog/python/face-recognition-with-python/ - face recognition with python

12. https://github.com/senko/python-video-converter/blob/master/converter/formats.py - python video converter