Πανεπιστήμιο Δυτικής Αττικής

Μεταπτυχιακό Πρόγραμμα Σπουδών

Εφαρμοσμένα Πληροφοριακά Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Κίνησης Αντικειμένων με Matlab/Simulink

Ιωάννης Γαλανάκης AIS-0134

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΑΠΡΙΛΙΟΣ 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Κίνησης αντικειμένων με Matlab/Simulink

Ιωάννης Γαλανάκης
Α.Μ. ΑΙS-0134

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 ……AIS-0134…….. 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 για την πολλαπλή ανάλυση και ανίχνευση αντικειμένων συγκεκριμένου ενδιαφέροντος και την βελτίωση των αλγορίθμων με εφαρμογή kalman filter

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 Matlab 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 for multi-object detection.

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 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

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.

Κεφάλαια

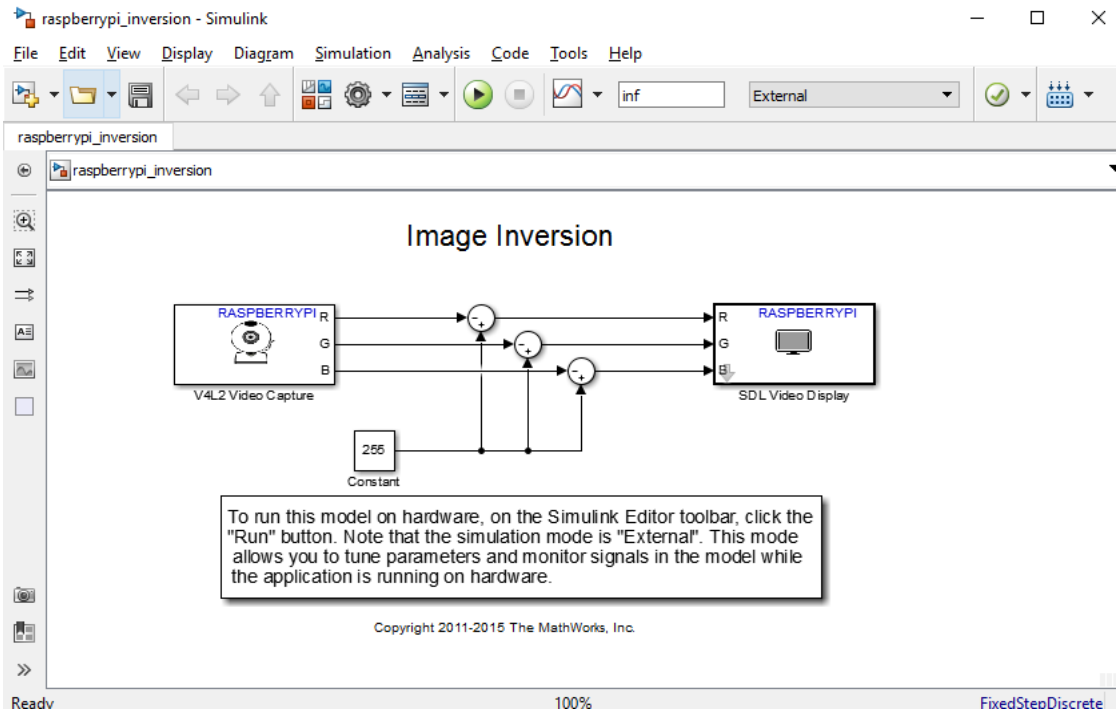
ΜΕΡΟΣ ΠΡΩΤΟ:Εκτελώντας ένα απλό Image Inversion στο Simulink

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:

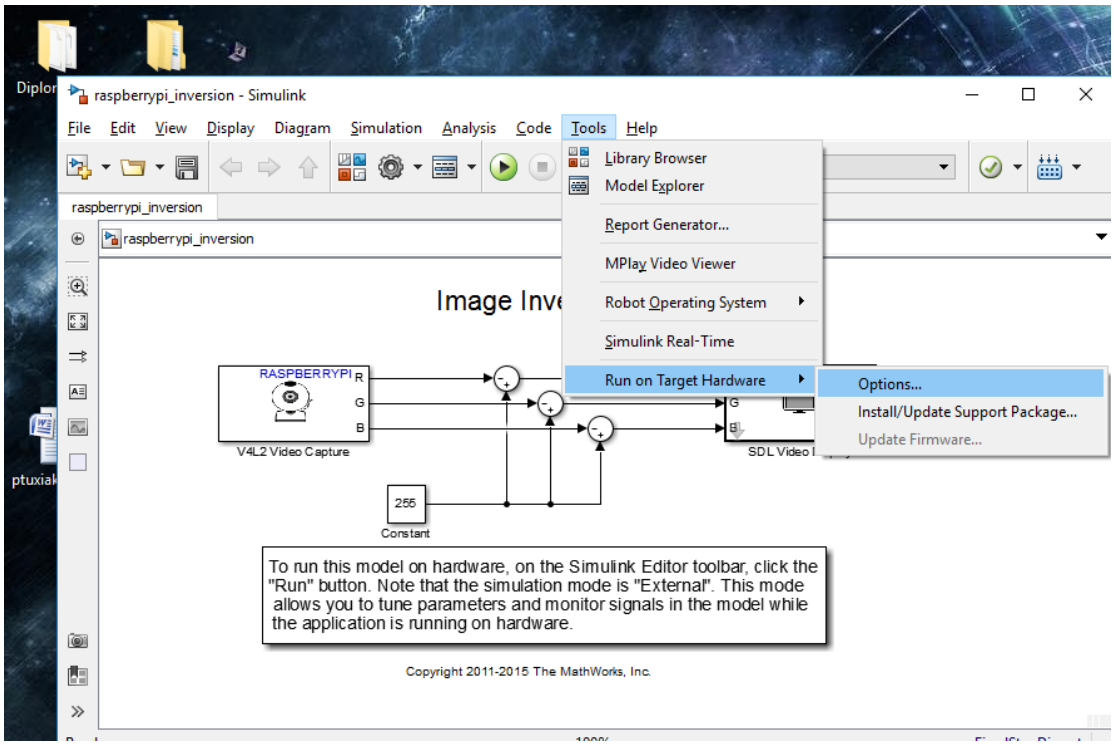


4.1 Image Inversion

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.

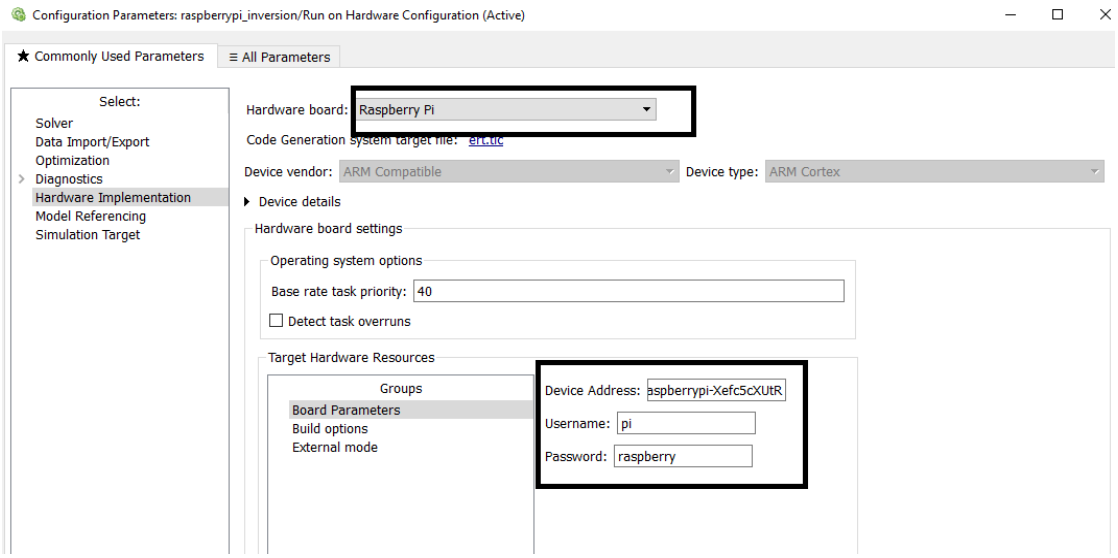
Πρόκειται για μια τεχνική επεξεργασίας εικόνας, όπου οι φωτεινές περιοχές μετατρέπονται σε σκοτεινές, και οι σκοτεινές περιοχές μετατρέπονται σε φωτεινές. Με άλλα λόγια, μετά την αναστροφή της εικόνας το μαύρο γίνεται άσπρο και το άσπρο γίνεται μαύρο. Μια ανεστραμμένη ασπρόμαυρη εικόνα (Inverted Image)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

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

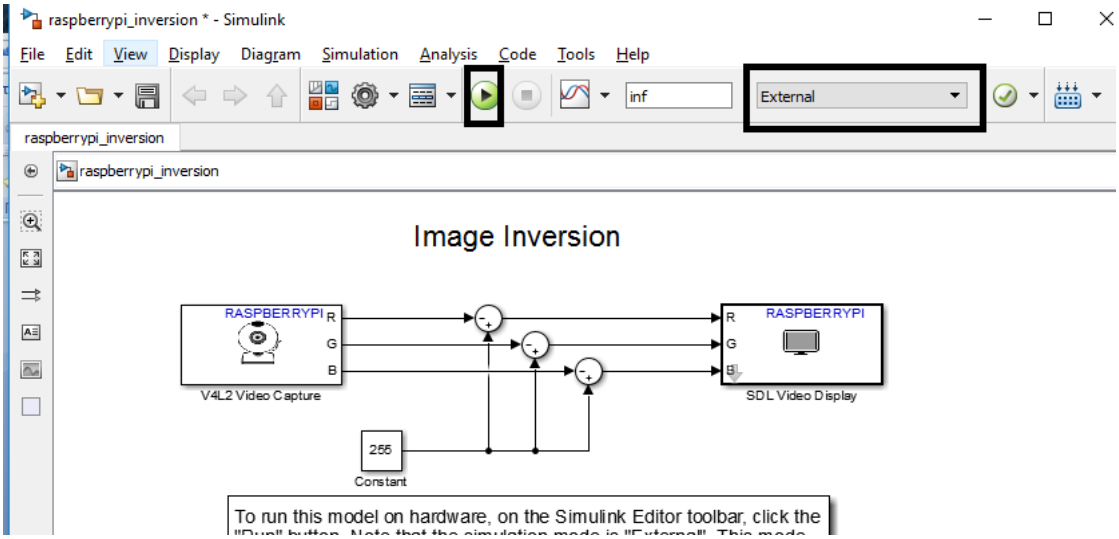


4.2 Image Inversion

Και επιβεβαιώνουμε ότι όλα τα παιδία είναι σωστά:

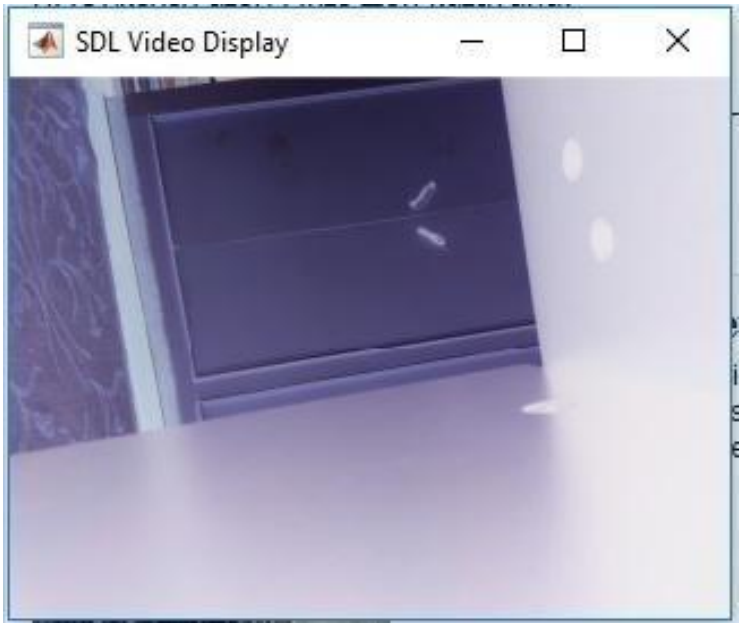


Επίσης φροντίζουμε να έχουμε επιλεγμένη την επιλογή External και τρέχουμε την προσομοίωση:



4.3 Image Inversion

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

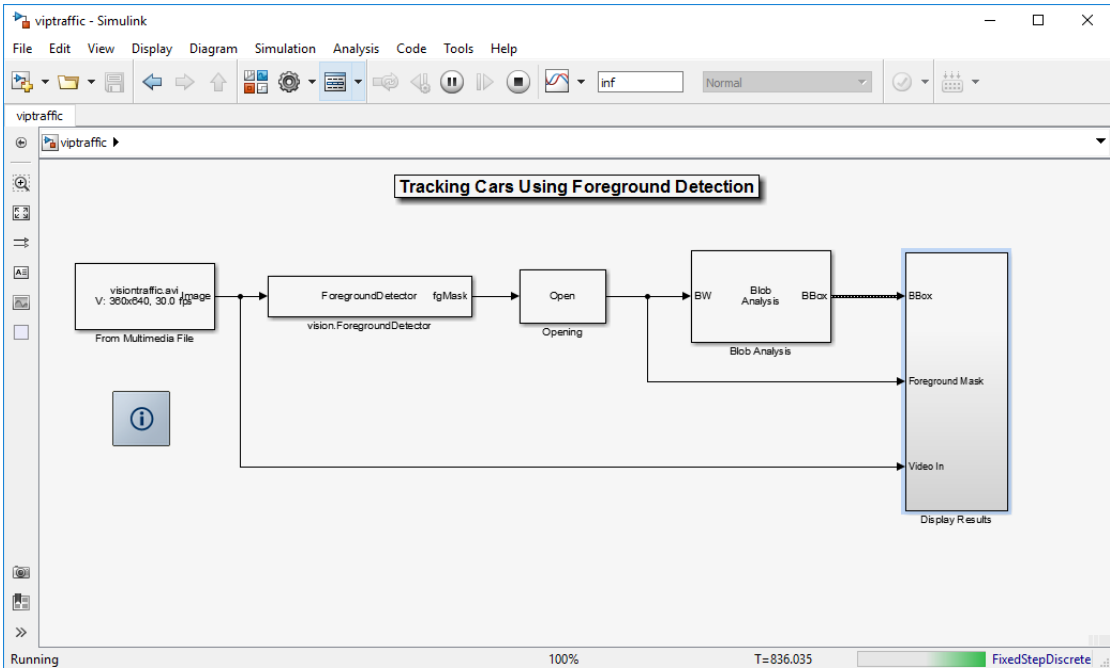


4.4 Results of Image Inversion

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

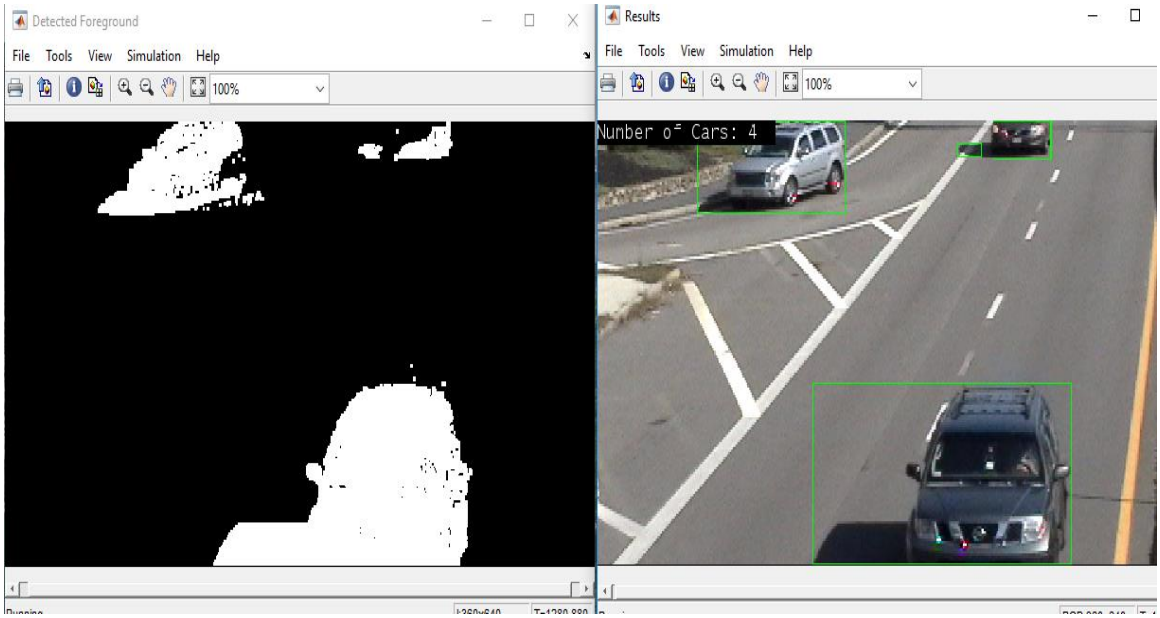
Καταμέτρηση αυτοκινήτων με ανίχνευση Foreground Detection

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

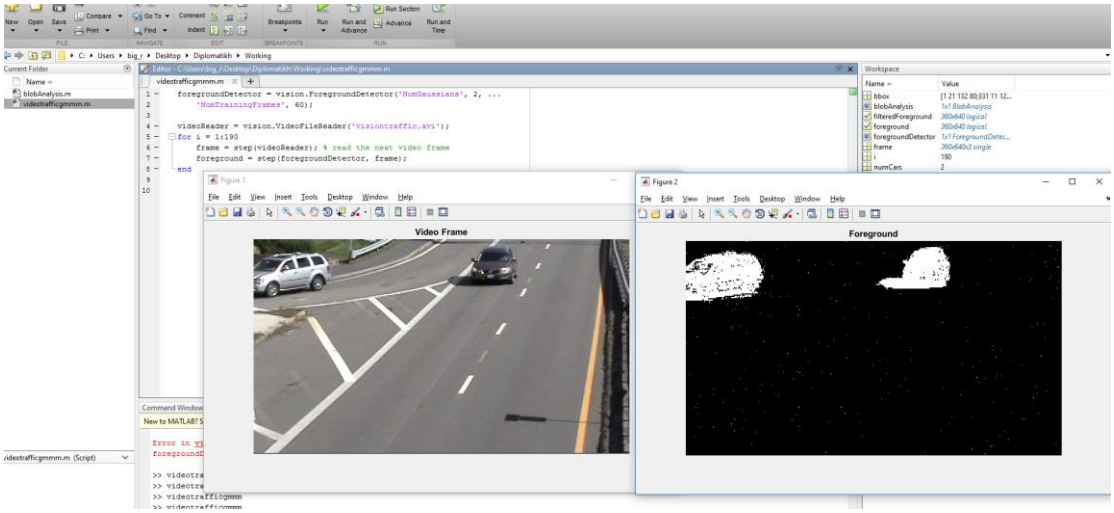
Όπως είδαμε στο παράδειγμα πίσω το μοντέλο μας στο Simulink χρησιμοποιεί Foreground Detection για να ανιχνεύσει τα αυτοκίνητα στο παρασκήνιο. Το τελικό αποτέλεσμα του κώδικα που βρίσκεται πίσω από αυτό είναι πολύπλοκο για αυτό θα το χωρίσουμε σε μικρά μέρη.

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');
```

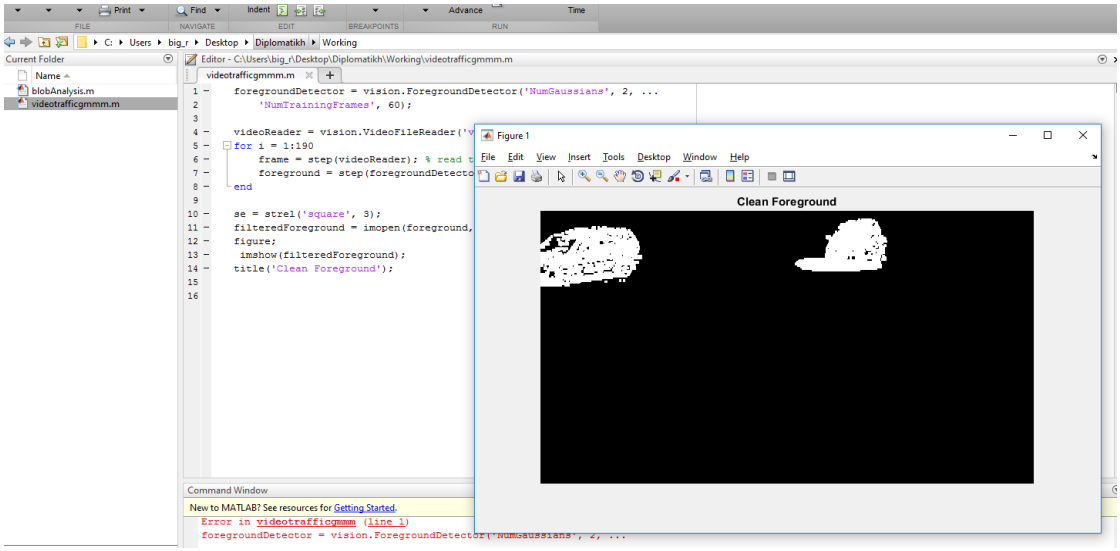


7.2 Simulink Foreground Model with GMM results

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:



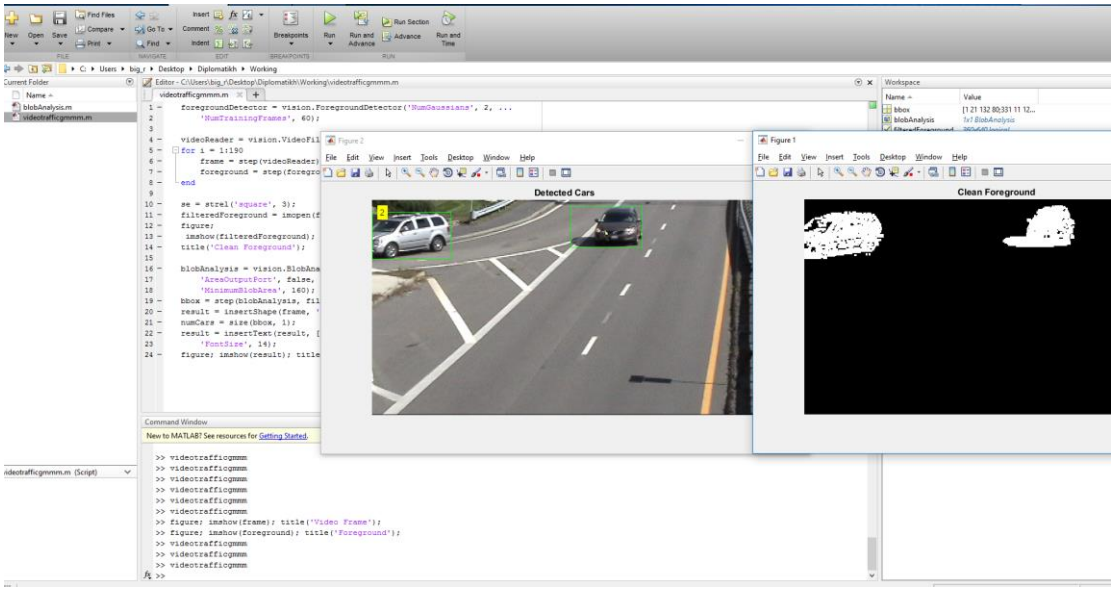
7.3 Simulink Foreground Model with GMM results

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');
```

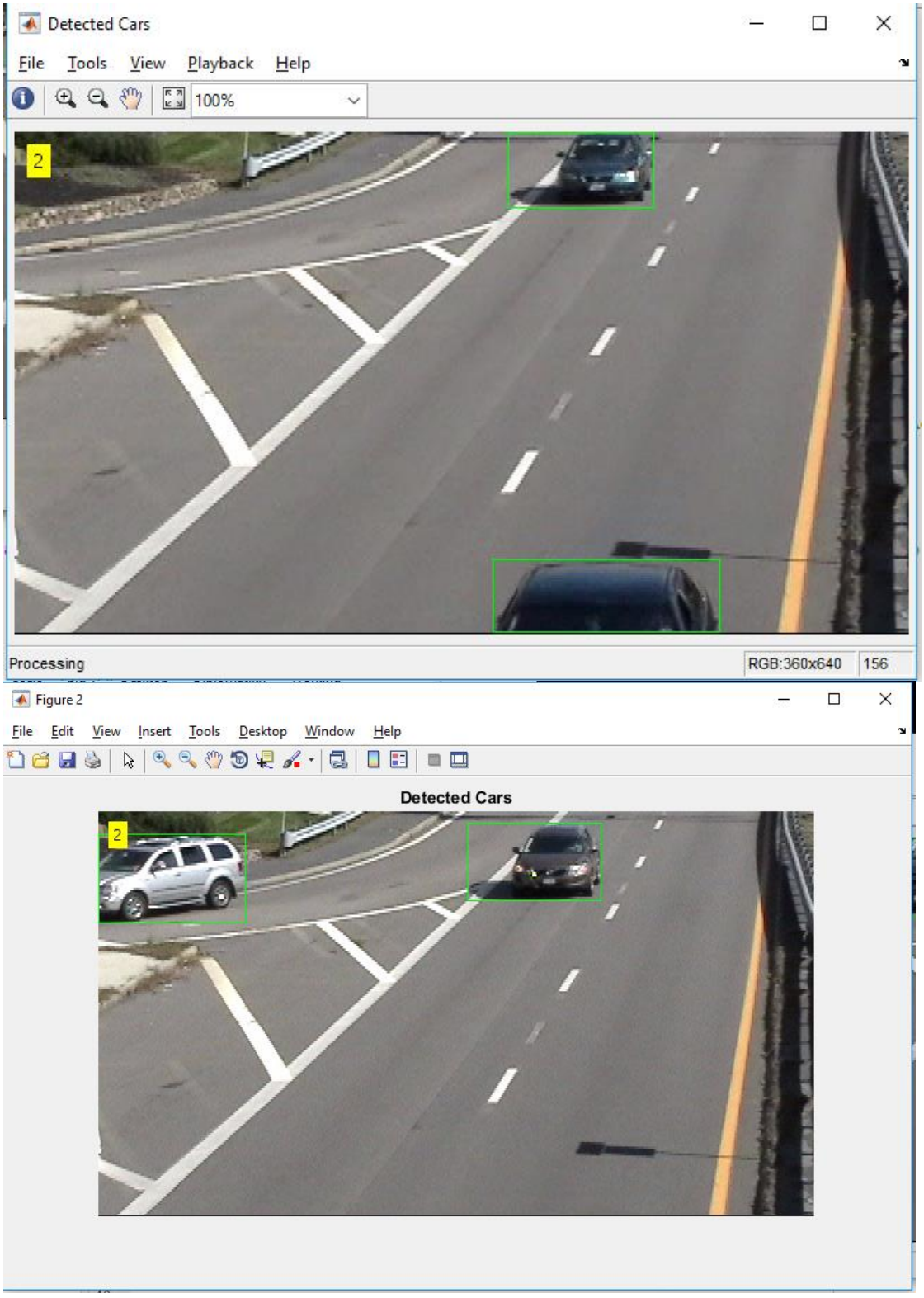


7.4 Simulink Foreground Results with GMM and green highlight

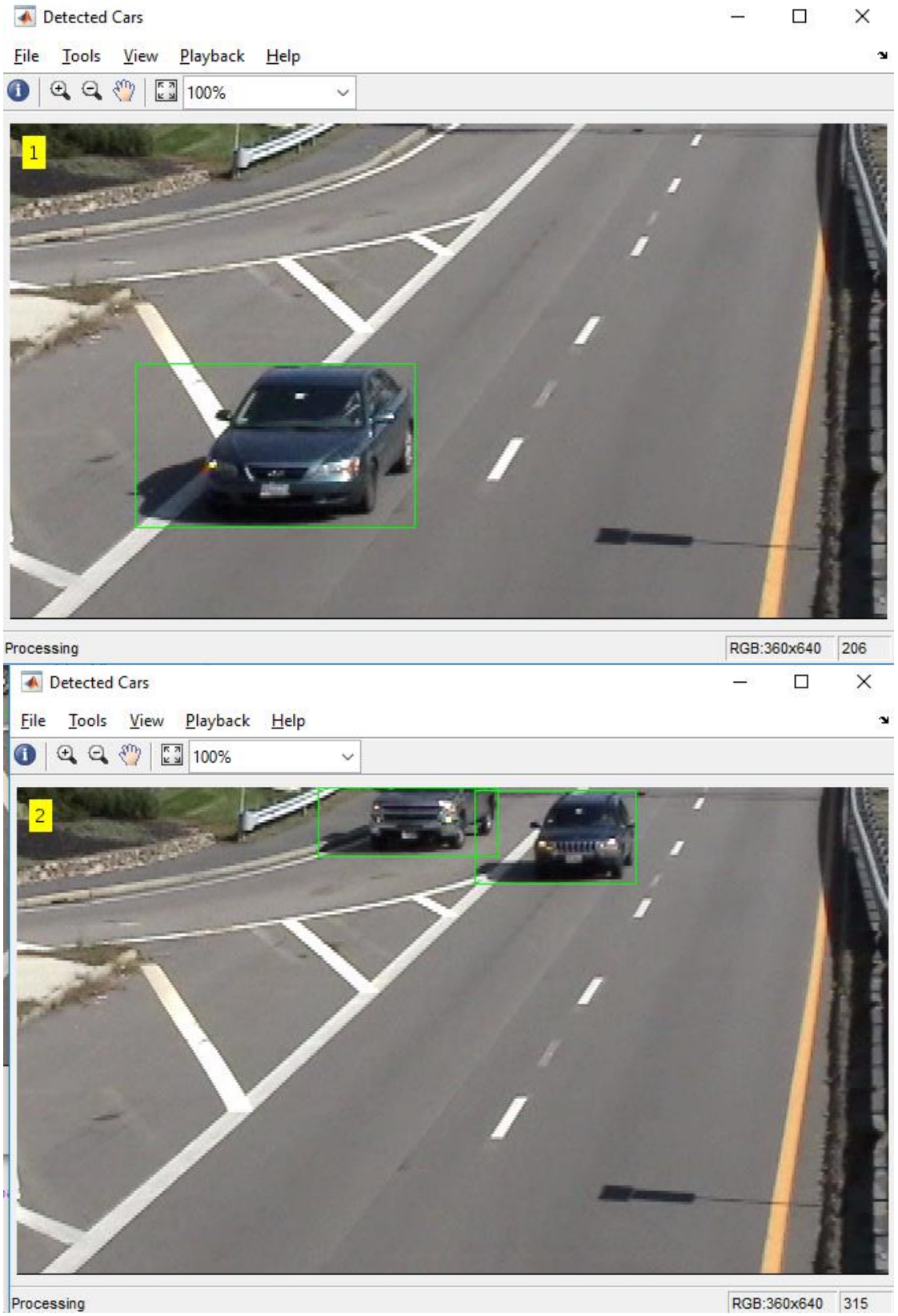
Τέλος για να ολοκληρώσουμε την διαδικασία πρέπει να επεξεργαστούμε και τα υπόλοιπα Video Frames:

```
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

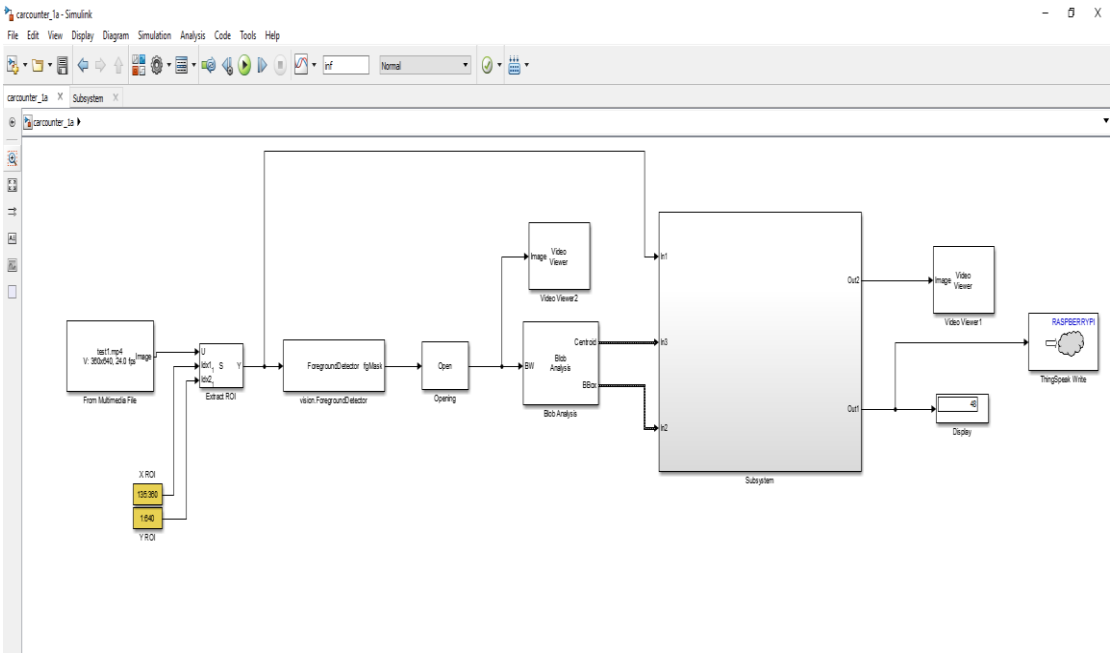
Region λεμε ενα αυποσύνολο pixels στο συνολο της εικονας

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.

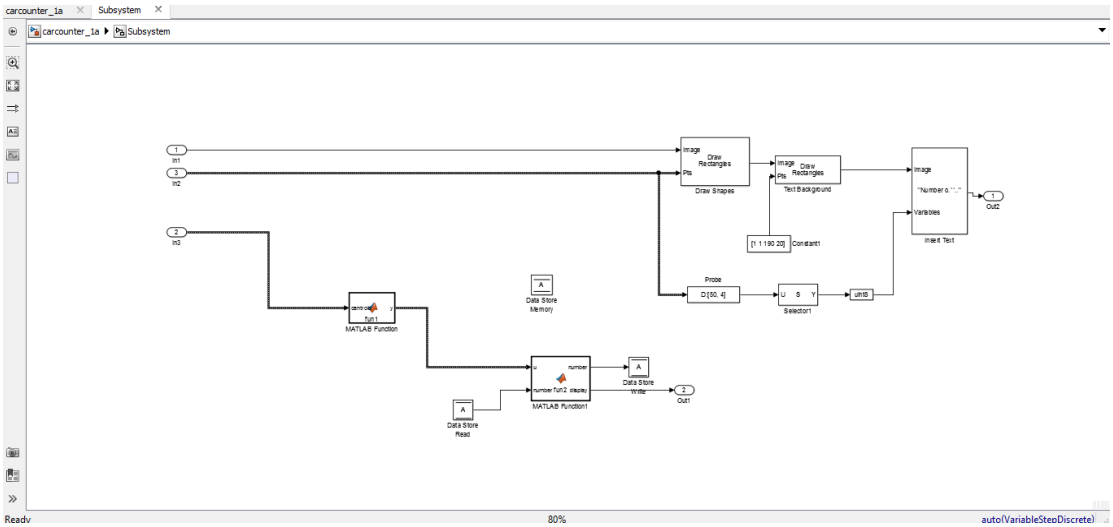
Το μοντέλο simulink το οποίο θα χρησιμοποιήσουμε για την επίτευξη του σκοπού αυτού είναι το παρακάτω:



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:

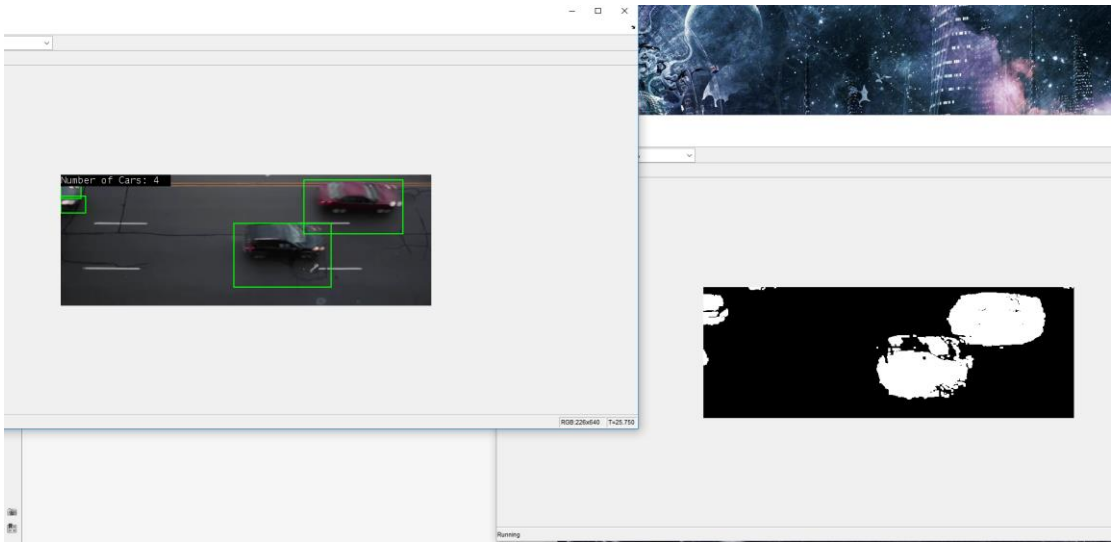


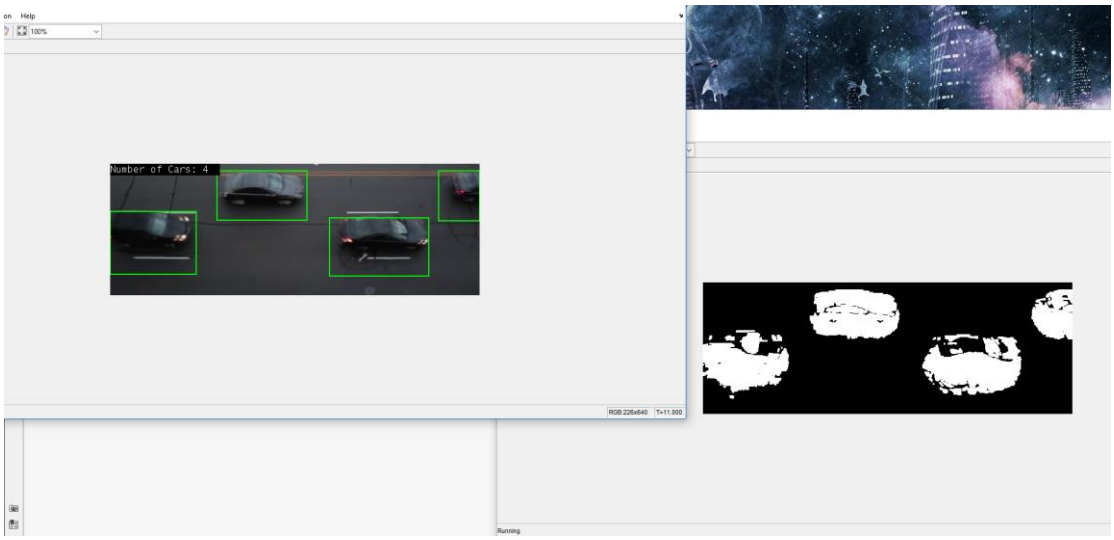
8.2 Car counter sub model with video input

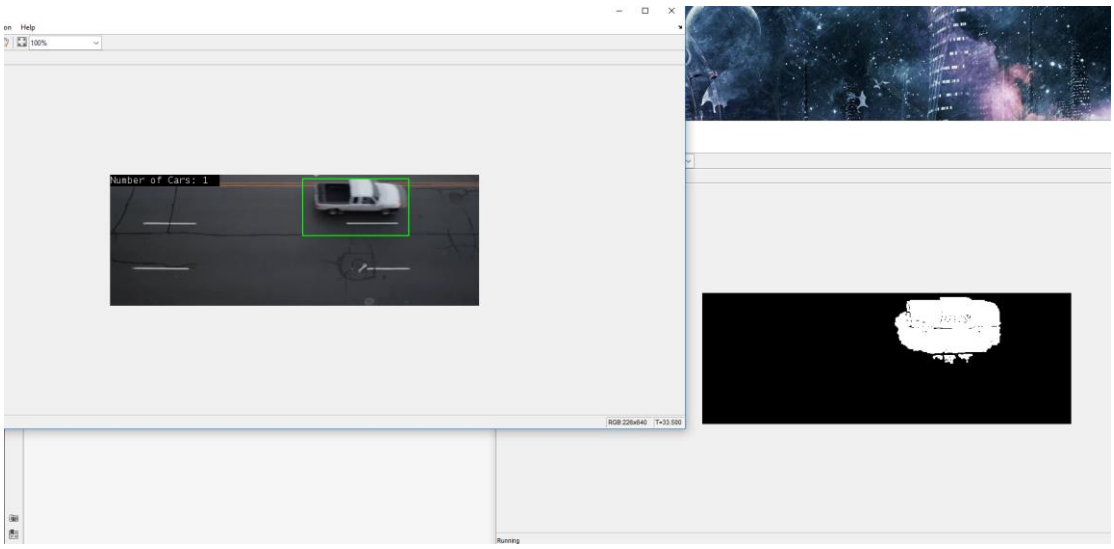
Στο κάτω μέρος του: αυτό το μοντέλο εκτελεί δύο απλές συναρτήσεις, την cancroids και την Matlab function που είναι υπεύθυνες για την δημιουργία πλαισίων και ενός καταμετρητή ο οποίος αυξάνεται κατά ένα κάθε φορά που ανιχνεύεται στην περιοχή ενδιαφέροντος ένα πλαίσιο. Το block Data store memory είναι 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.

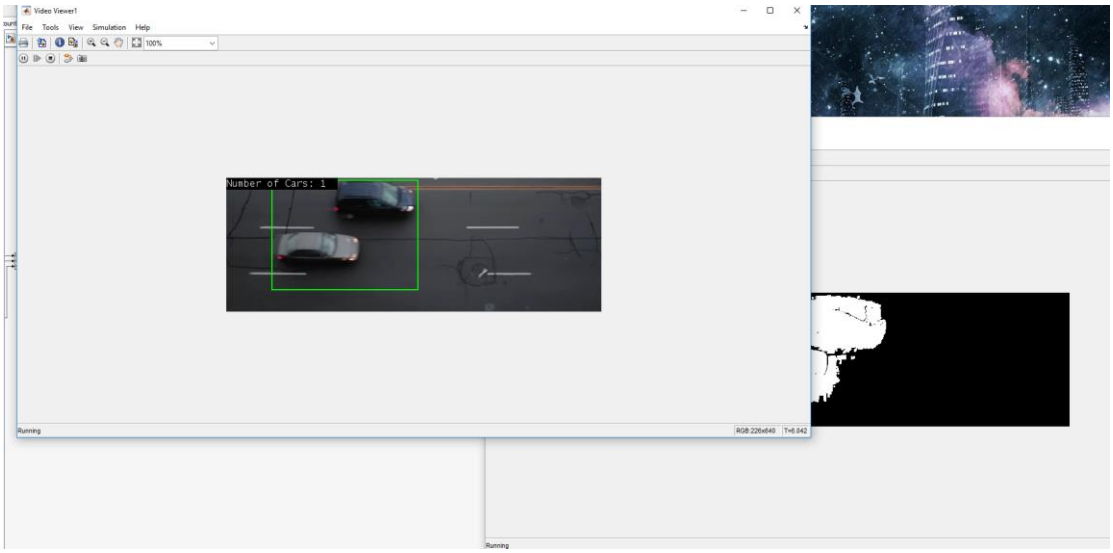
Στο πάνω μέρος του: το block draw shapes σχεδιάζει τα ορθογώνια πλαίσια ανίχνευσης για τα αντικείμενα ενδιαφέροντος καθώς παίρνει σαν είσοδο int2 ένα σήμα μεταβλητής από τα αποτελέσματα της blob analysis . Η Probe είναι υπεύθυνη για την δημιουργία ενός δυσδιάστατου δυναμικού πίνακα με μέγιστο μέγεθος 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.

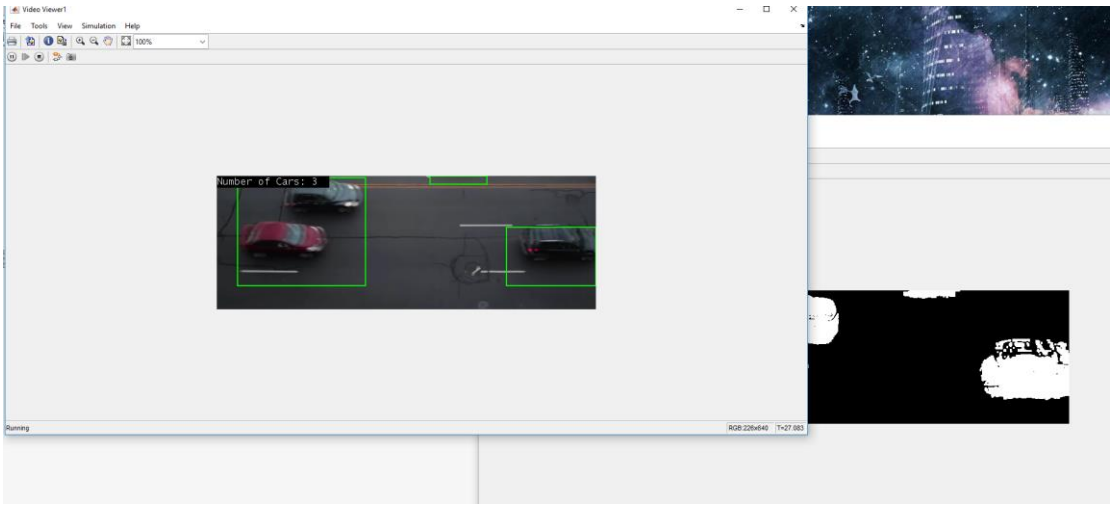
Τα στιγμιότυπα που παίρνουμε από την εκτέλεση του κώδικα είναι τα ακόλουθα:











8.3-8.7 Screenshot results from simulink model with video input

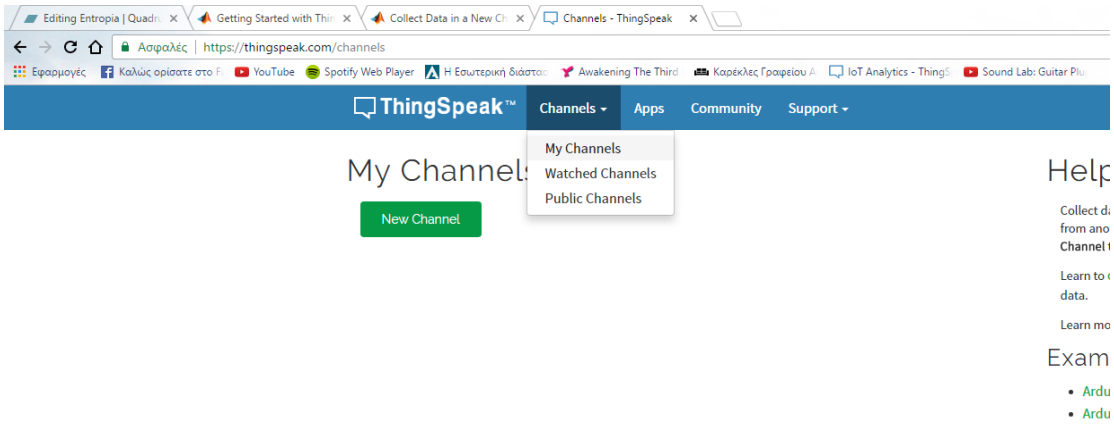
Όπως βλέπουμε τα αποτελέσματα του κώδικά μας είναι κατά προσέγγιση σωστά αλλά όχι 100% ακριβείας.

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.



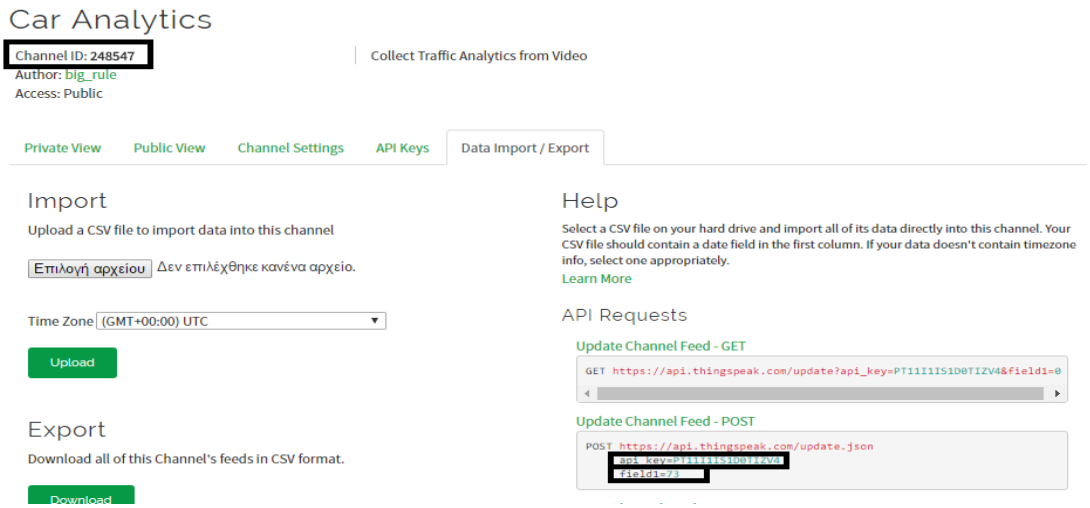
9.1 Thingspeak channel

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 :

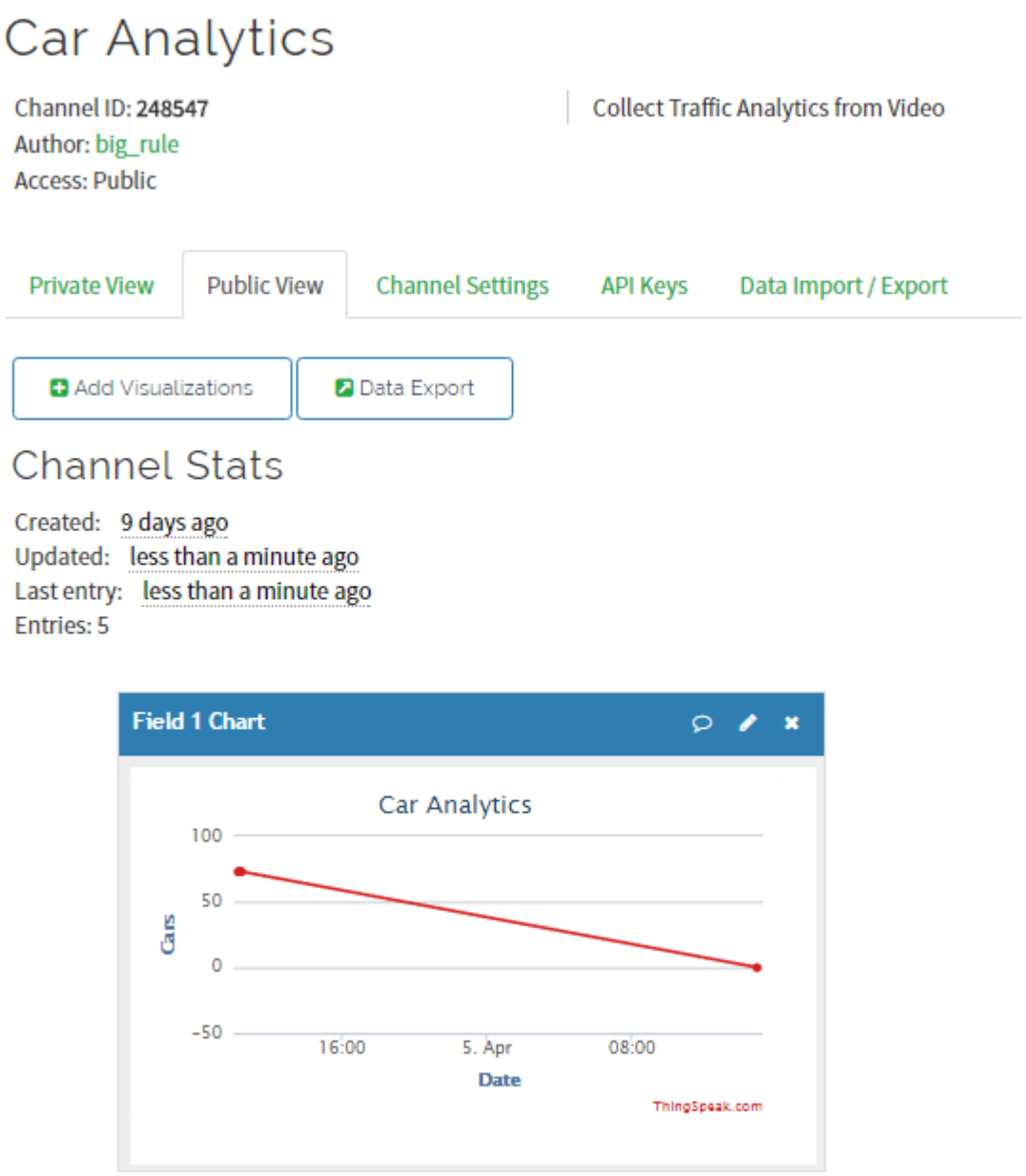
>> thingSpeakWrite(248547,73,'WriteKey','PT11I1IS1D0TIZV4');

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



9.2 Thingspeak channel

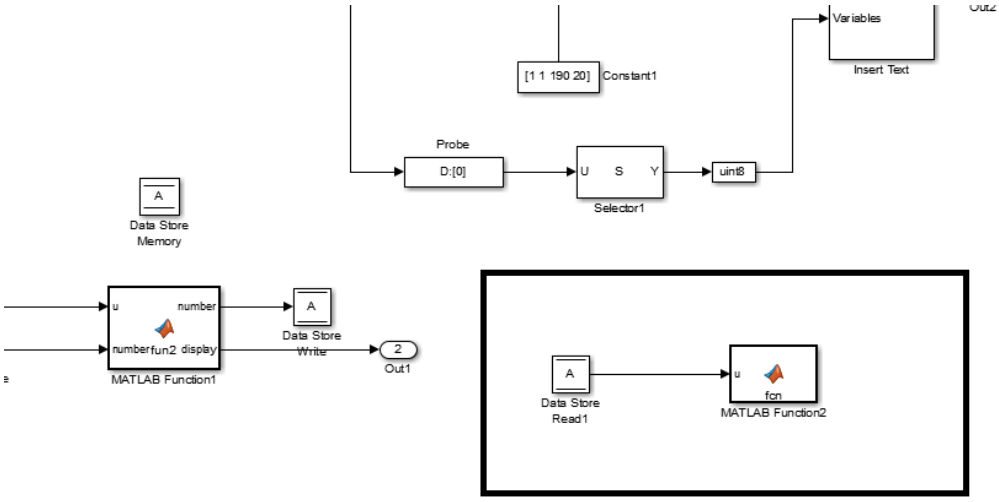
Με την εντολή αυτή στέλνουμε request στο thingspeak για να κάνει update και να περάσει τα νέα analytics που του στέλνει η εκτέλεση του μοντέλου μας. Τα αποτελέσματα με τα statistics στο thing speak θα είναι τα ακόλουθα:



9.3 Thingspeak statistics

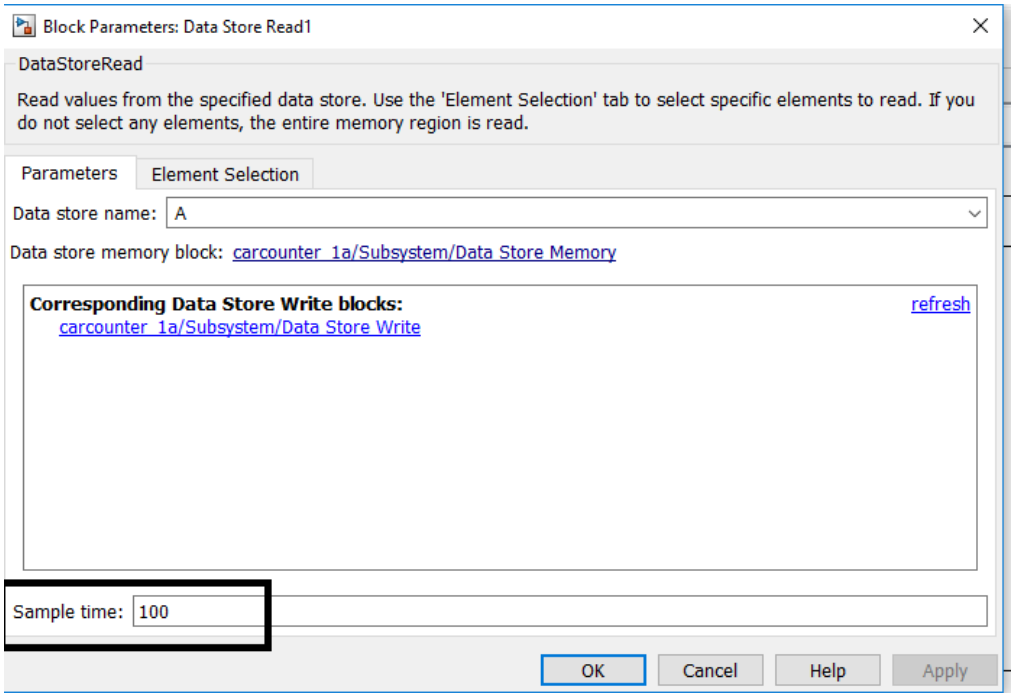
Για τους σκοπούς της εργασίας μας χρειαζόμαστε real time analytics capture οπότε δεν είναι δυνατόν να στέλνουμε τα δεδομένα μας μέσω εντολών. Θέλουμε να γίνεται αυτόματα. Για αυτό το λόγο φτιάχνουμε ένα μικρό και εύκολο μοντέλο το οποίο θα στέλνει post requests στο thingspeak αυτόματα κάθε 100 seconds.

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

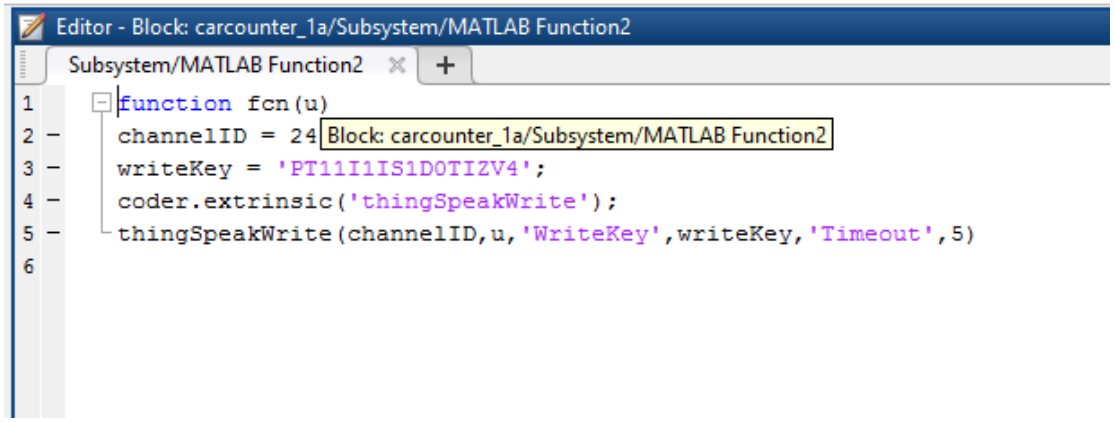


9.4 submodel of simulink car counter model

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:

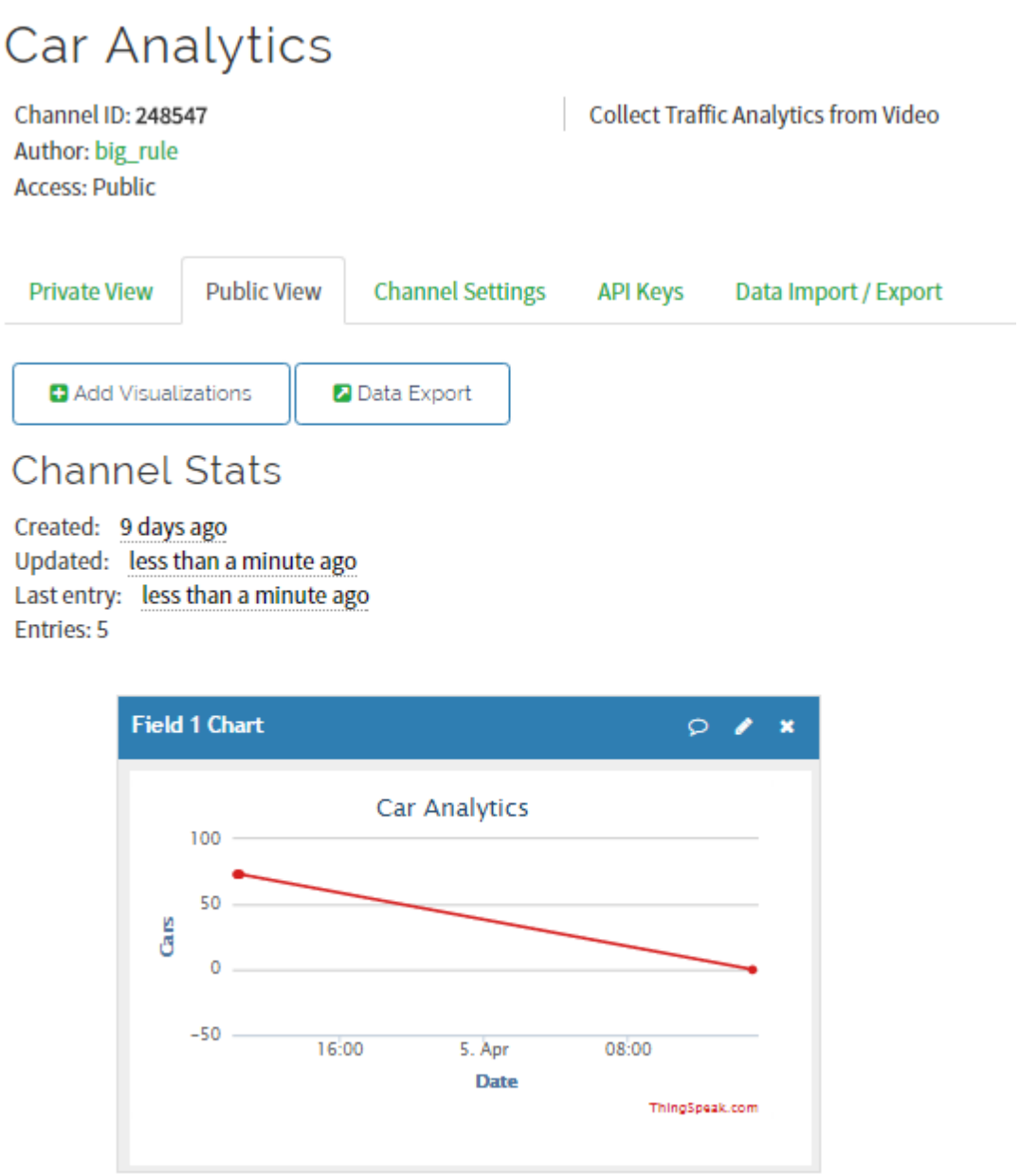


9.5 store data block



9.6 autoupdate algorithm

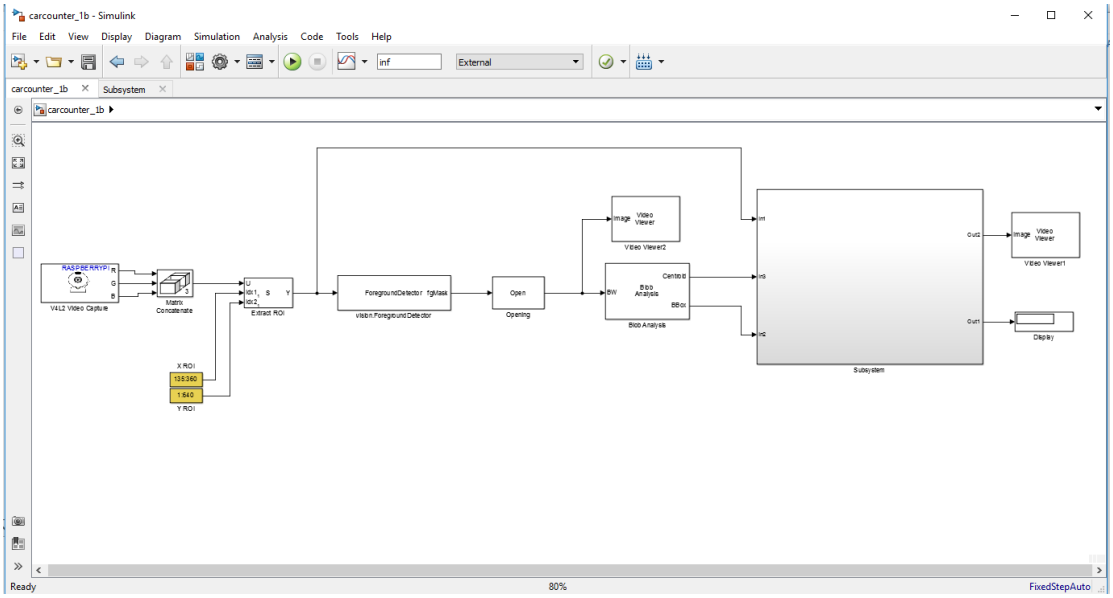
Με την εκτέλεση του μοντέλου μας περιμένοντας ένα λεπτό μπαίνουμε στο κανάλι που δημιουργήσαμε στο Thingspeak και βλέπουμε τα αποτελέσματα:



9.7 Thingspeak statistics

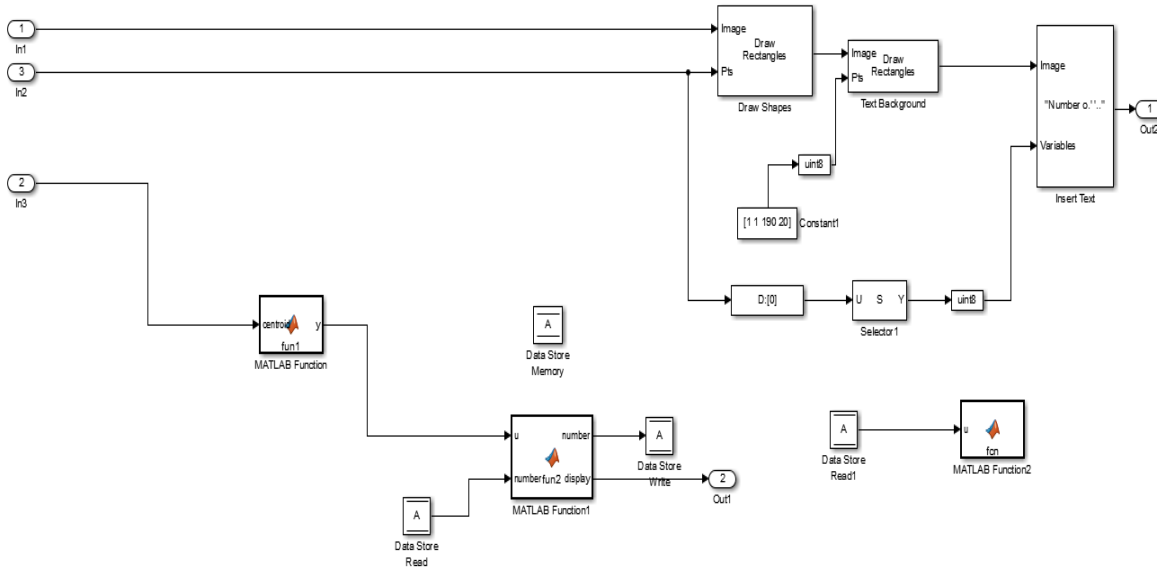
Καταμέτρηση αυτοκινήτων και ανάλυση κίνησης στο Thing Speak από video πραγματικού χρόνου στο Raspberry Pi με κάμερα.

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.

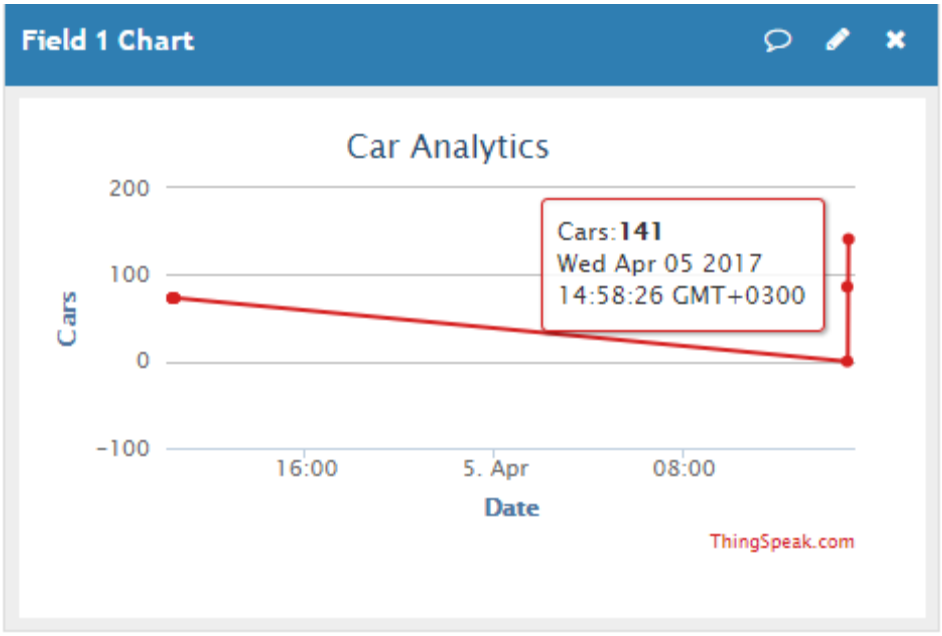


10.1 simulink car counter with camera input

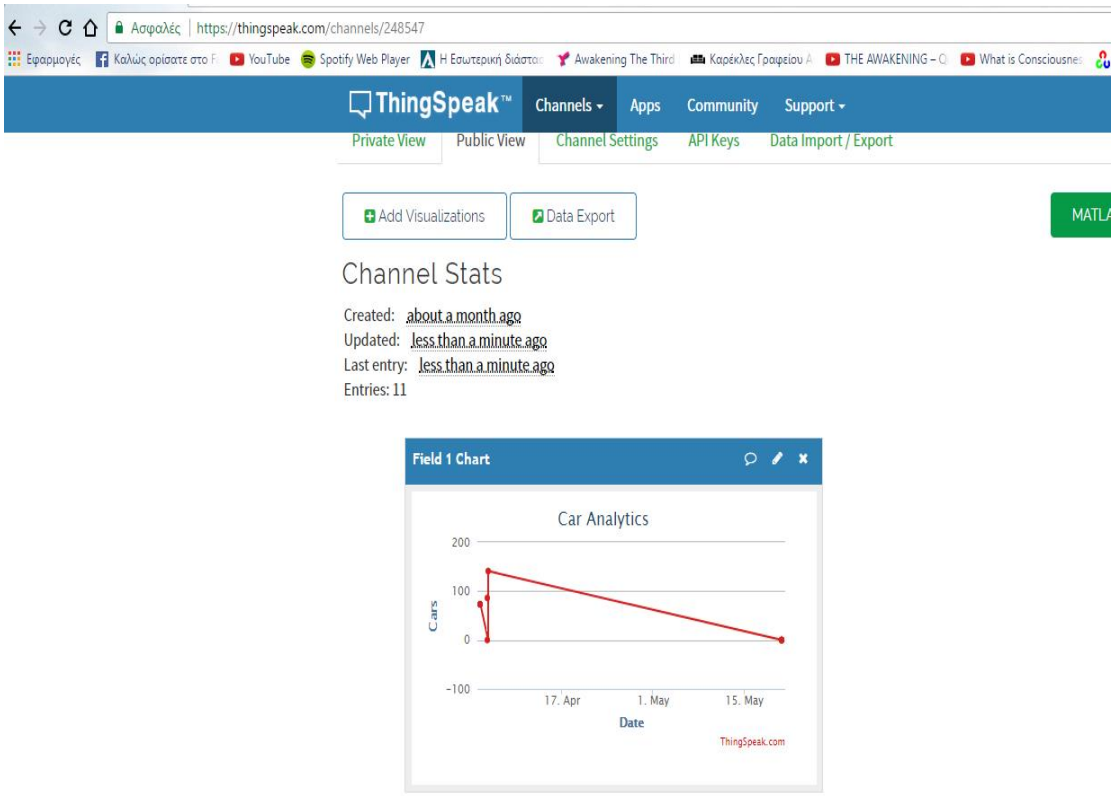
Όπως βλέπουμε το σχηματικό είναι ακριβώς το ίδιο με μόνη διαφορά την είσοδο που στο συγκεκριμένο παράδειγμα είναι η κάμερα του raspberry pi. Επίσης προσθέσαμε στο sub model ένα uint8 στο constant1 διότι έβλεπε το προηγούμενο μπλοκ σαν πραγματικό αριθμό ενώ ήθελε ακέραιο. Πολλές φορές χρειάζεται να κάνουμε 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).

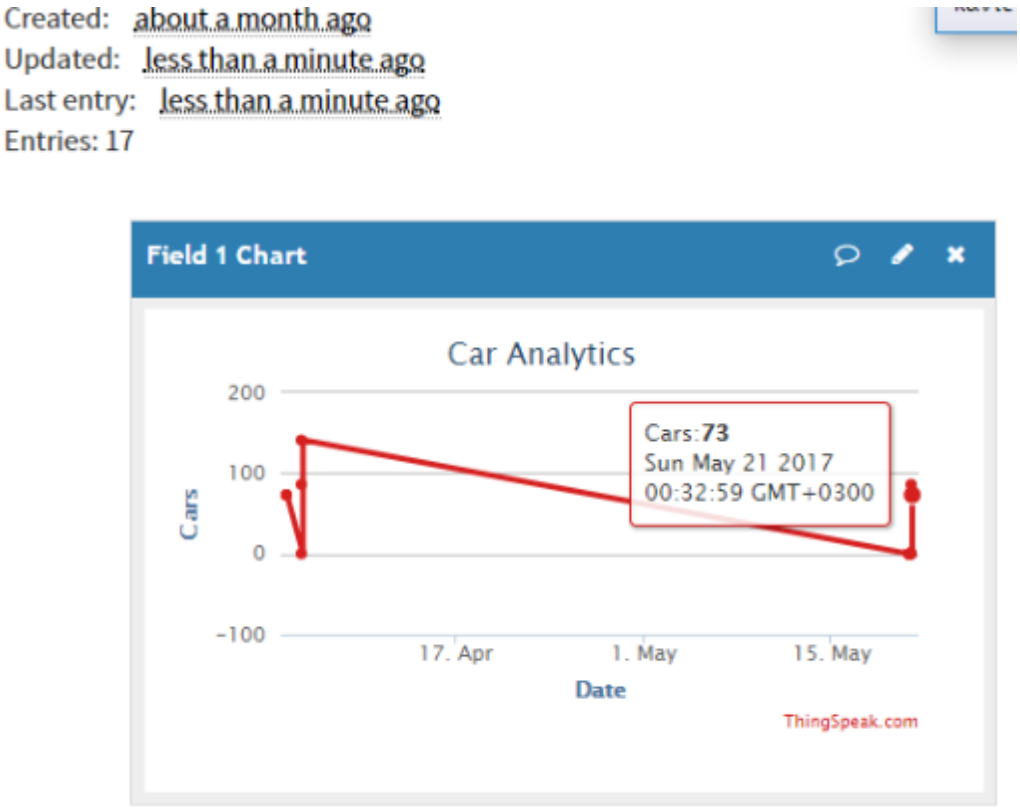


10.2 simulink car counter submodel with camera input

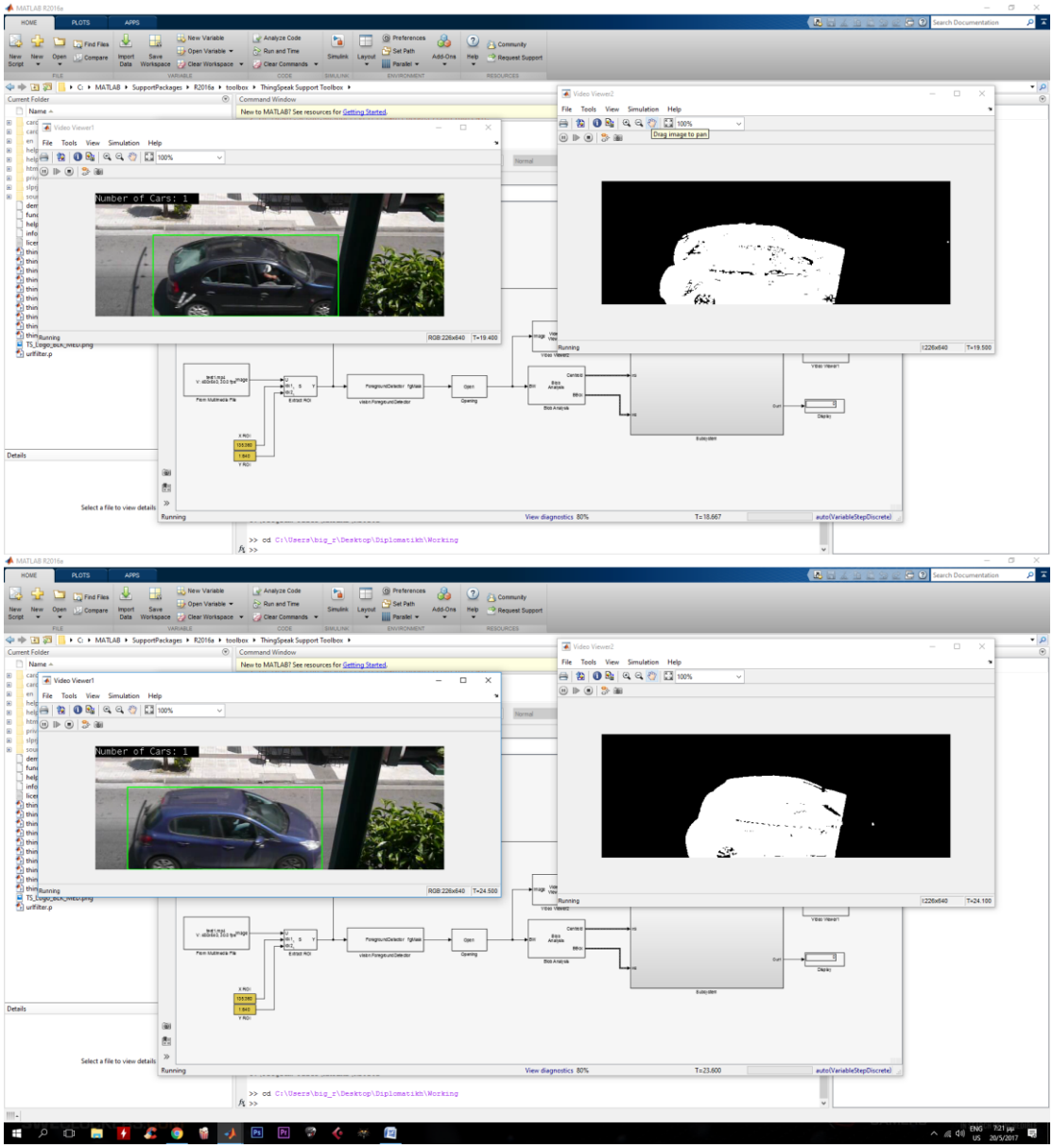


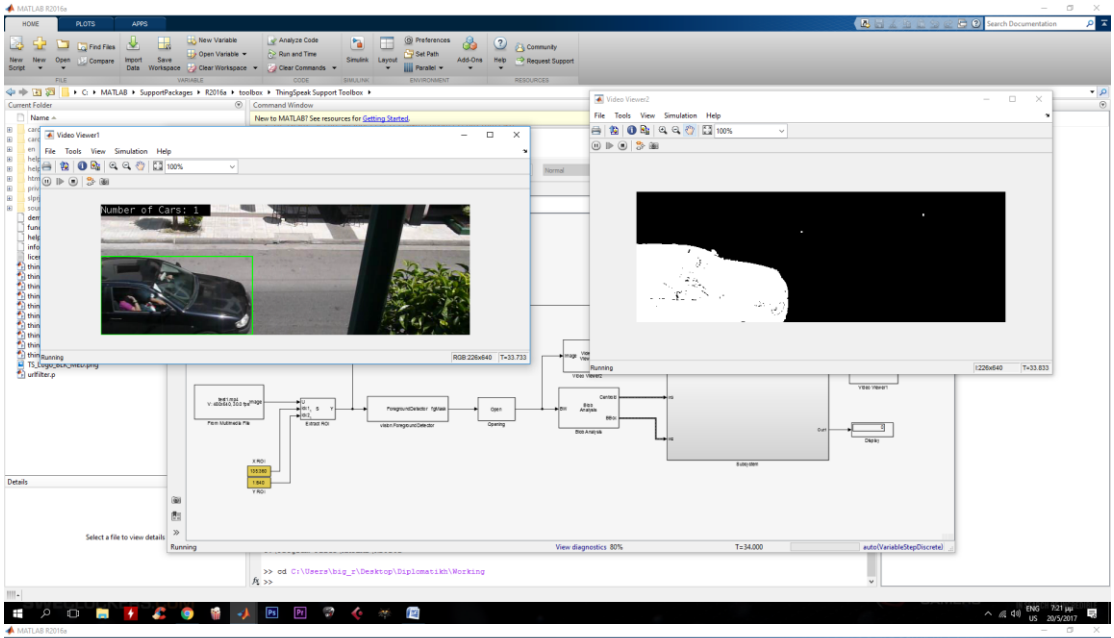
10.3 thingspeak channel analytics

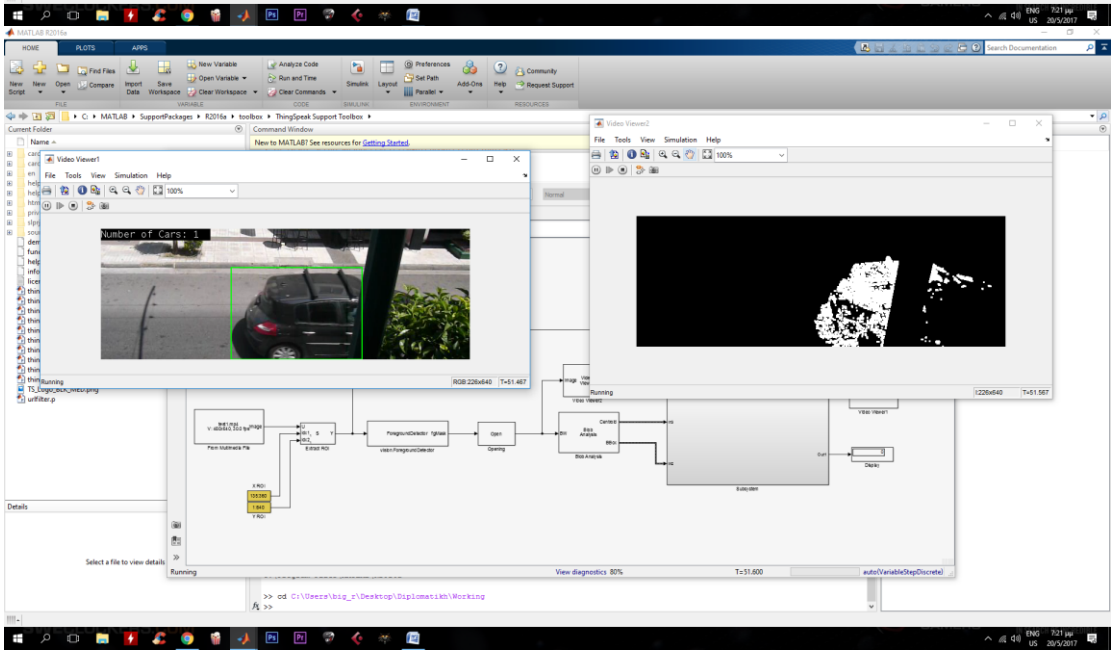


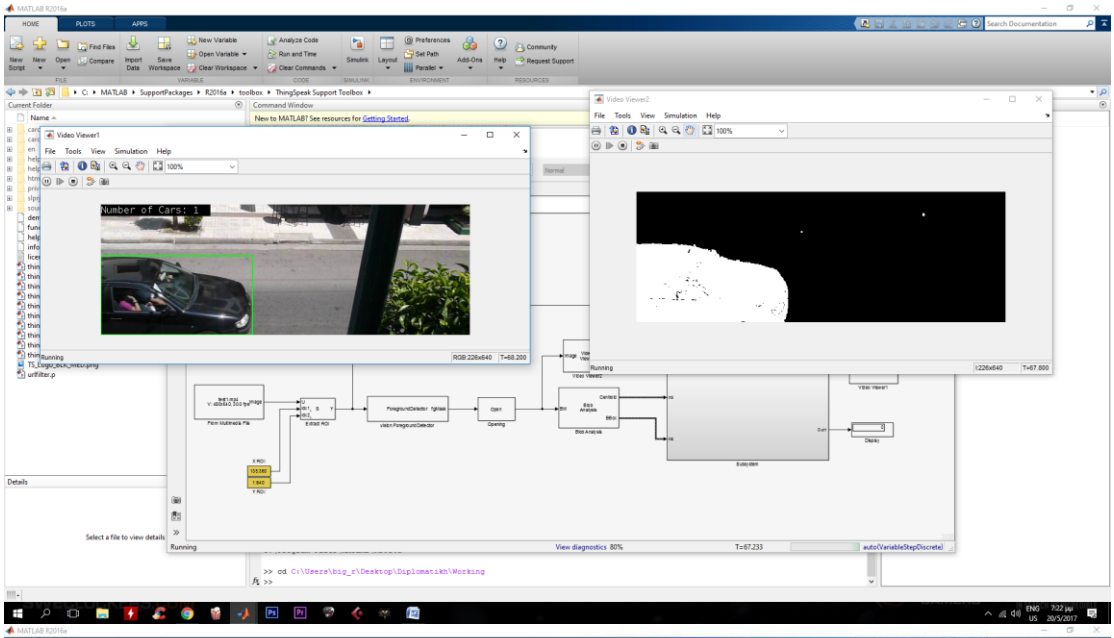


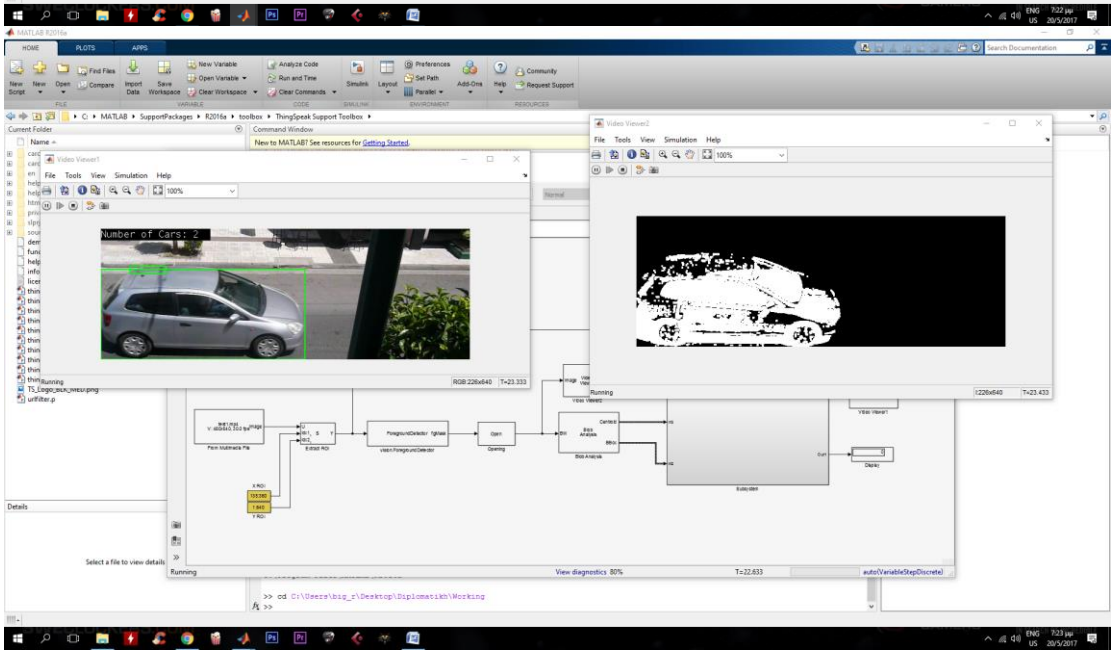
10.4-10,5 thingspeak channel analytics

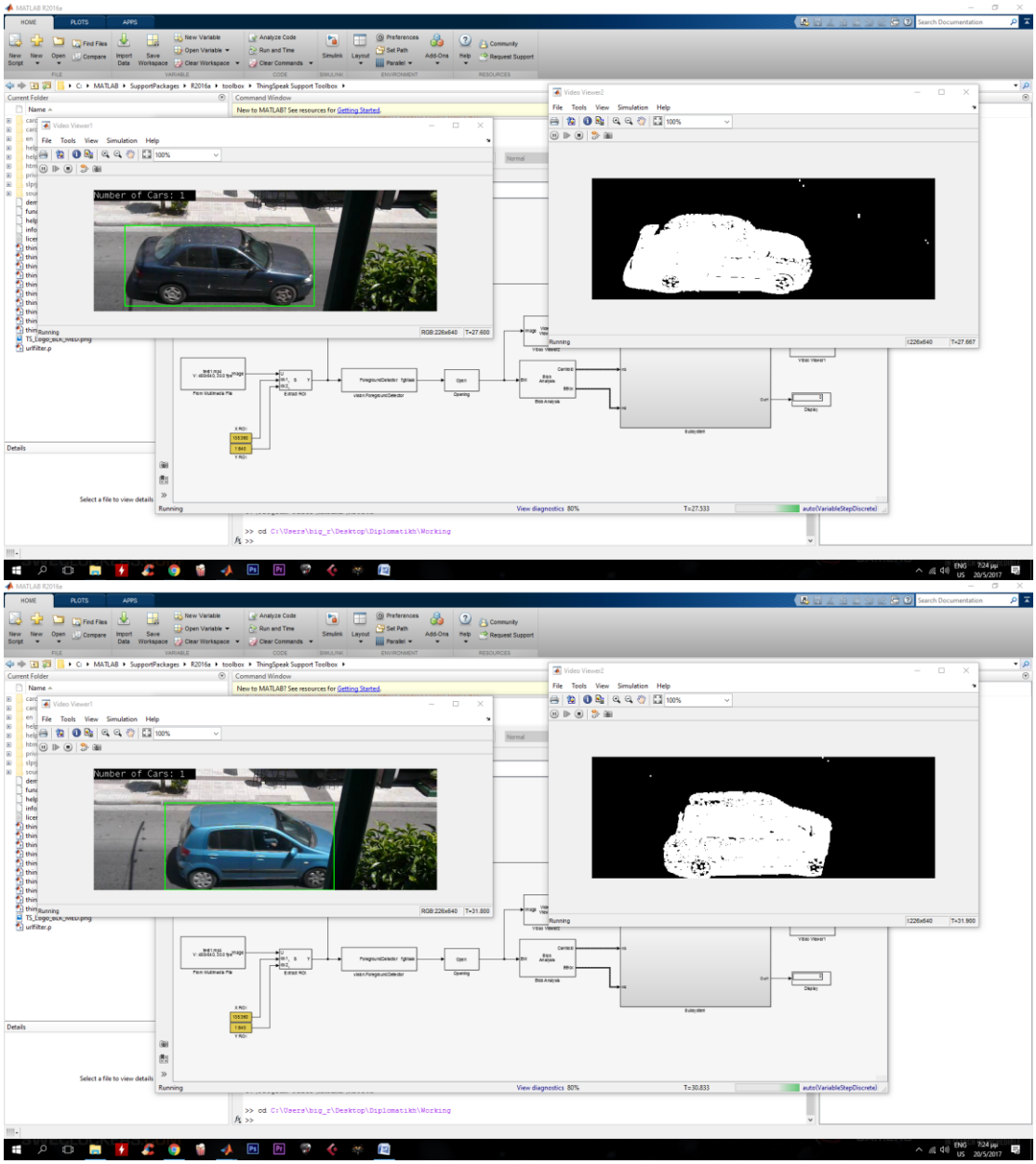

Number of Cars: 1
Number of Cars: 1


Number of Cars: 1


Number of Cars: 1

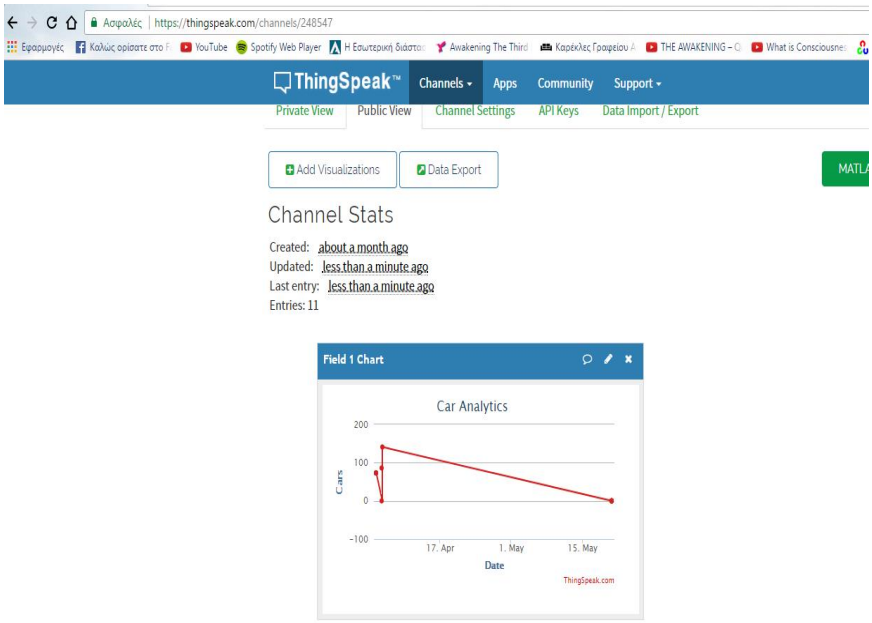

Number of Cars: 1


Number of Cars: 2



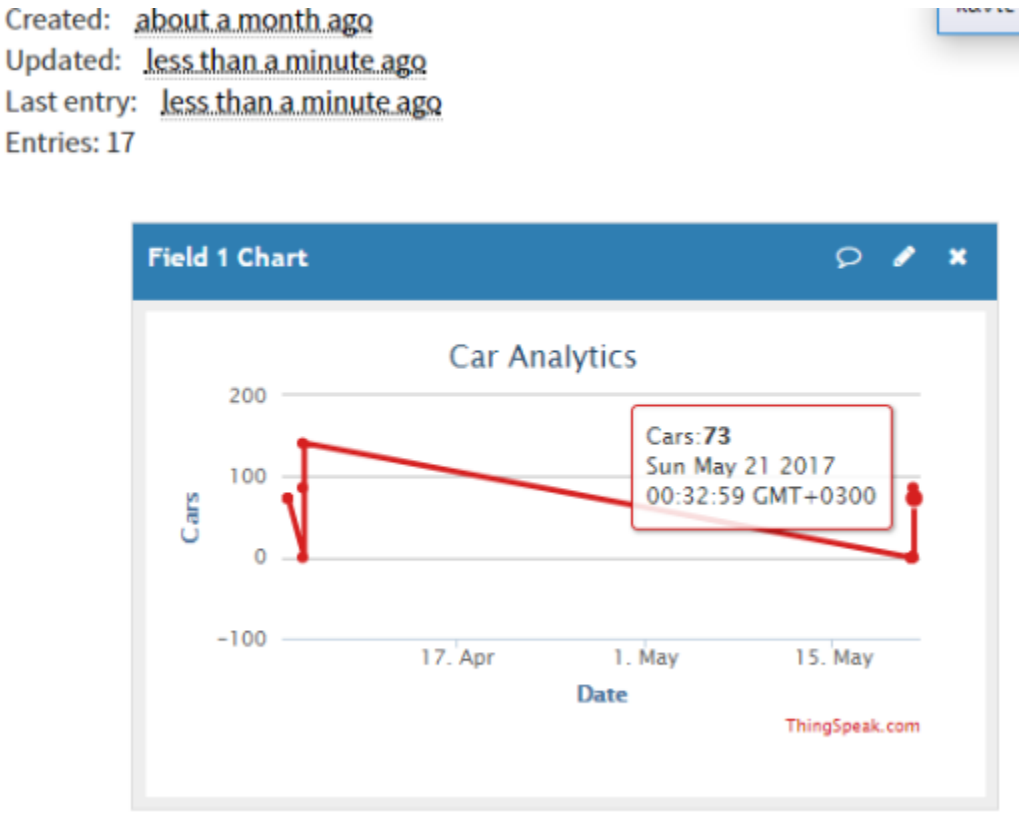
10.6-10.13 screenshot of the final results

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

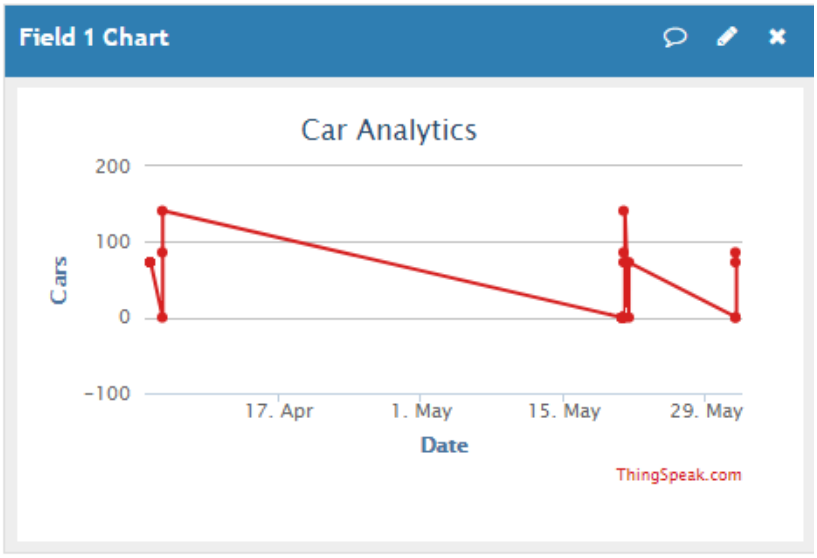


11.1 thingspeak channel analytics

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.

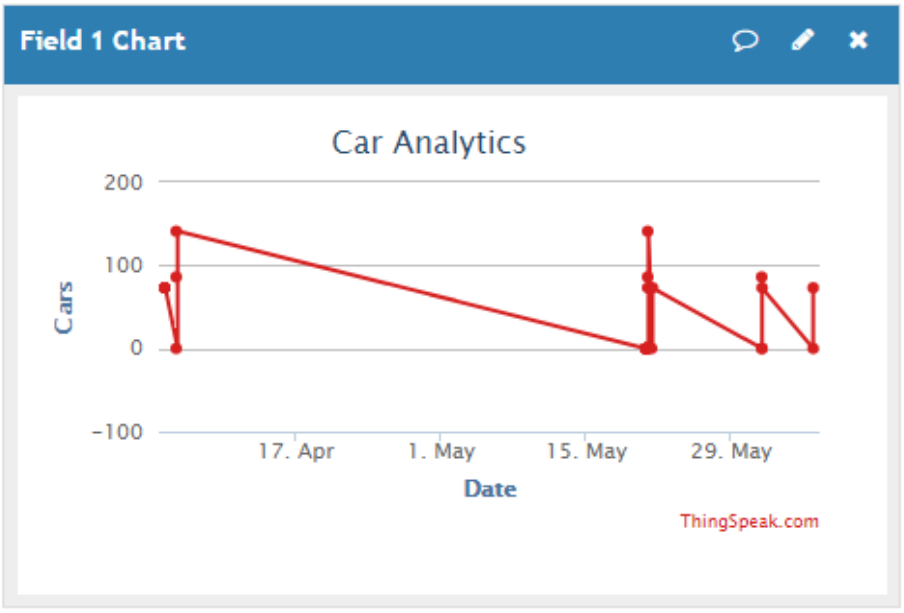


11.2 thingspeak channel analytics



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: Βελτίωση του αλγορίθμου με εφαρμογή κ χρήση του Kalman Filter σε Matlab

Το φίλτρο Κάλμαν ή Kalman Filter γνωστό κ ως γραμμική τετραγωνική εκτίμηση είναι γνωστό στην θεωρία στατιστικών και ελέγχου και αποτελεί έναν αλγόριθμο που χρησιμοποιεί μια σειρά μετρήσεων που παρατηρούνται σε μια συγκεκριμένη χρονική περίοδο και περιέχει στατιστικό θόρυβο και άλλες ανακρίβειες πάνω στις οποίες παράγει εκτιμήσεις αγνώστων μεταβλητών που τείνουν να είναι περισσότερο ακριβείς από εκείνες που βασίζονται αποκλειστικά σε μία μέτρηση μόνο, υπολογίζοντας μια κοινή κατανομή πιθανότητας στις μεταβλητές για κάθε χρονικό πλαίσιο φίλτρο πήρε την ονομασία του από τον Rudolf E. Kaman

Το φίλτρο Kalman είναι ένα σύνολο μαθηματικών εξισώσεων που παρέχει αποτελεσματικά υπολογιστικά (επαναληπτικά) μέσα για την εκτίμηση της κατάστασης μιας διαδικασίας, κατά τρόπο που να ελαχιστοποιείται ο μέσος όρος των τετραγώνων των σφαλμάτων. Το φίλτρο είναι πολύ ισχυρό σε διάφορες πτυχές: υποστηρίζει εκτιμήσεις του παρελθόντος, του παρόντος, και ακόμη και τις μελλοντικές καταστάσεις, και μπορεί να το πράξει ακόμη όταν η ακριβής φύση του διαμορφωμένου συστήματος είναι άγνωστη.

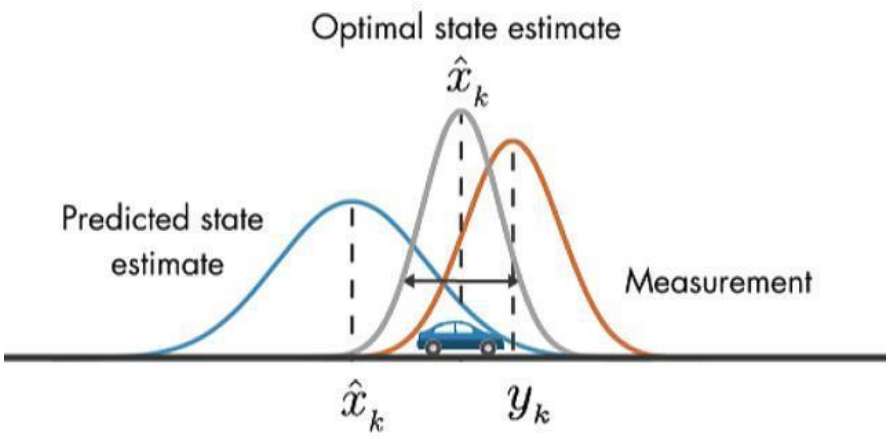
Το φίλτρο αυτό έχει πολλές εφαρμογές στην τεχνολογία η δημοφιλέστερη από αυτές είναι η καθοδήγηση, η πλοήγηση και ο έλεγχος των οχημάτων αεροσκαφών και διαστημικών οχημάτων καθώς και μαχητικών αεροπλάνων. Τα φίλτρα Kalman αποτελούν επίσης ένα από τα κύρια θέματα στον τομέα του σχεδιασμού και του ελέγχου ρομποτικής κίνησης και μερικές φορές συμπεριλαμβάνονται στη βελτιστοποίηση τροχιάς. Το φίλτρο Kalman λειτουργεί επίσης για τη μοντελοποίηση του ελέγχου της κίνησης του κεντρικού νευρικού συστήματος.

Ο αλγόριθμος αυτός λειτουργεί σε μια διαδικασία από δύο στάδια όπου στο πρώτο βήμα της πρόβλεψης όπου το φίλτρο Kalman παράγει εκτιμήσεις των μεταβλητών της τρέχουσας κατάστασης, μαζί με τις αβεβαιότητές τους. Μόλις παρατηρηθεί το αποτέλεσμα της επόμενης μέτρησης οι εκτιμήσεις αυτές επικαιροποιούνται χρησιμοποιώντας σταθμισμένο μέσο όρο, δίνοντας μεγαλύτερη βαρύτητα σε εκτιμήσεις με μεγαλύτερη βεβαιότητα.

Ο αλγόριθμος αυτός δρα αναδρομικά δηλαδή λειτουργεί και σε πραγματικό χρόνο χρησιμοποιώντας μόνο τις τρέχουσες μετρήσεις εισόδου και την κατάσταση που είχε υπολογιστεί πριν μαζί με τον πίνακα αβεβαιότητας.

Με την χρήση αυτού του φίλτρου δεν υποθέτουμε ότι τα σφάλματα είναι Γκαουσιανά. Ωστόσο το φίλτρο μας αποδίδει μια ακριβή εκτίμηση πιθανότητας υπό κάποιους συγκεκριμένους όρους όταν όλα τα σφάλματά μας τύχει να είναι Γκαουσιανά.

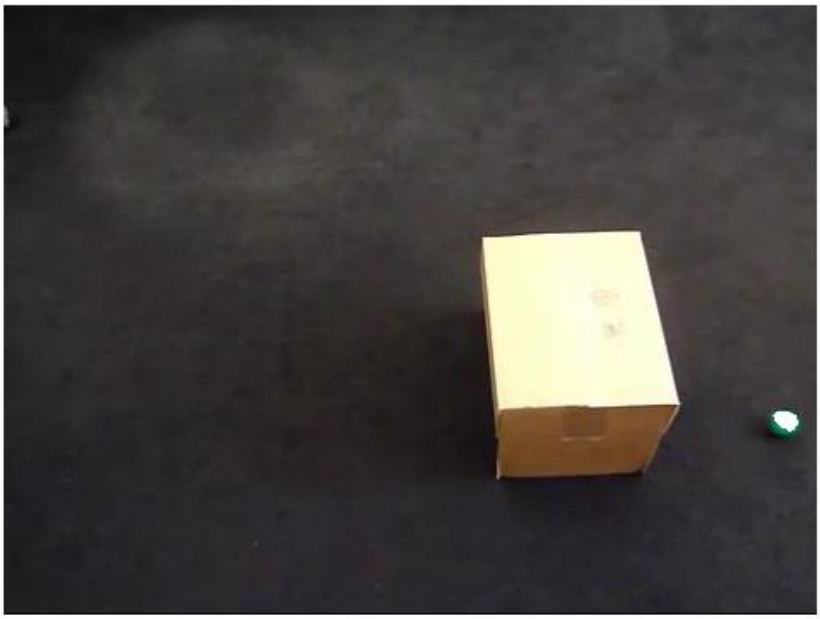
Επιπλέον, το φίλτρο Kalman είναι μια ευρέως εφαρμοσμένη έννοια στην ανάλυση χρονομετρών που χρησιμοποιείται σε πεδία όπως η επεξεργασία σήματος και η οικονομετρία.



Εφαρμογή του αλγορίθμου Kalman Filter στην ανίχνευση κινούμενου αντικειμένου σε MATLAB

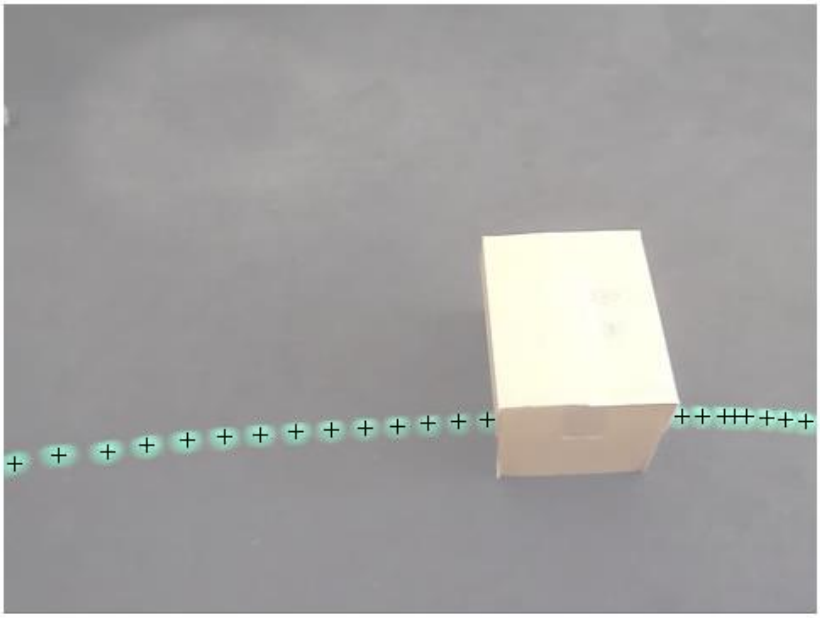
Πριν πάμε στην χρήση του φίλτρου Kalman πρώτα θα εξετάσουμε μια απλή ανίχνευση αντικειμένου σε ένα βίντεο.

Η εικόνα παρακάτω δείχνει μια πράσινη μπάλα που μετακινείται από αριστερά προς τα δεξιά στο πάτωμα κάνοντας χρήση της συνάρτησης showDetections () στο Matlab



Η λευκή περιοχή πάνω από τη σφαίρα υπογραμμίζει τα εικονοστοιχεία που ανιχνεύονται χρησιμοποιώντας vision ForegroundDetector, τα οποία διαχωρίζουν τα κινούμενα αντικείμενα από το φόντο. Η αφαίρεση υποβάθρου βρίσκει μόνο ένα τμήμα της μπάλας λόγω της χαμηλής αντίθεσης μεταξύ της μπάλας και του δαπέδου όμως το πρόβλημα που προκύπτει είναι ότι η διαδικασία αυτή μας παρέχει πολύ μεγάλο ποσοστό θορύβου.

Για μεγαλύτερη ευκολία απεικόνισης κατά μήκος όλης της τροχιάς του αντικειμένου, επικαλύπτουμε όλα τα video frames σε μία μόνο εικόνα. Τα "+" υποδεικνύουν τα κεντροειδή που υπολογίστηκαν χρησιμοποιώντας ανάλυση blob κάνοντας χρήση της συνάρτησης Matlab showTrajectory();



Από το αποτέλεσμα αυτό παρατηρούμε 2 πράγματα.

Αρχικά ότι το κέντρο της περιοχής είναι συνήθως διαφορετικό από το κέντρο της μπάλας δηλαδή υπάρχει ένα σφάλμα στη μέτρηση της θέσης της μπάλας.

Εν συνεχεία παρατηρώ ότι η θέση της σφαίρας δεν είναι διαθέσιμη όταν είναι κλεισμένη από το κουτί πράγμα που δηλώνει ότι έχουμε ελλιπή μέτρηση.

Παρακολούθηση σε ένα ενιαίο αντικείμενο χρησιμοποιώντας το φίλτρο Kalman

Κάνοντας χρήση του ίδιου βίντεο με πριν χρησιμοποιούμε τη συνάρτηση trackSingleObject που μας δείχνει πως μπορούμε να κάνουμε δημιουργία της vision.KalmanFilter για την εφαρμογή KalmanFilter στο παρών βίντεο.. Επίσης:

Μας κάνει μια πρόβλεψη και διόρθωση της μεθόδου σε μια αλληλουχία για να εξαλείψουμε τον παρών θόρυβο στο σύστημά ανίχνευσής μας.

Κάνοντας χρήση της predict μεθόδου μπορούμε αυτόματα να υπολογίσουμε την τρέχουσα τοποθεσία της σφαίρας όταν αυτή επικαλύπτεται απο το κουτί.

Η επιλογή των παραμέτρων φίλτρου Kalman μπορεί να είναι δύσκολη. Η configureKalmanFilter απλοποιεί αυτό το πρόβλημα όπως δείχνει το παρακάτω παράδειγμα.

Η trackSingleObject λειτουργία περιλαμβάνει ενσωματωμένες λειτουργίες βοήθειας. Οι παρακάτω μεταβλητές ανώτατου επιπέδου χρησιμοποιούνται για τη μεταφορά δεδομένων μεταξύ των ενσωματωμένων λειτουργιών.

```
frame             = [];  % A video frame
detectedLocation = [];  % The detected location
trackedLocation  = [];  % The tracked location
label             = '';  % Label for the ball
utilities          = [];  % Utilities used to process
the video
```

Η διαδικασία για ανίχνευση αντικειμένου είναι η παρακάτω:

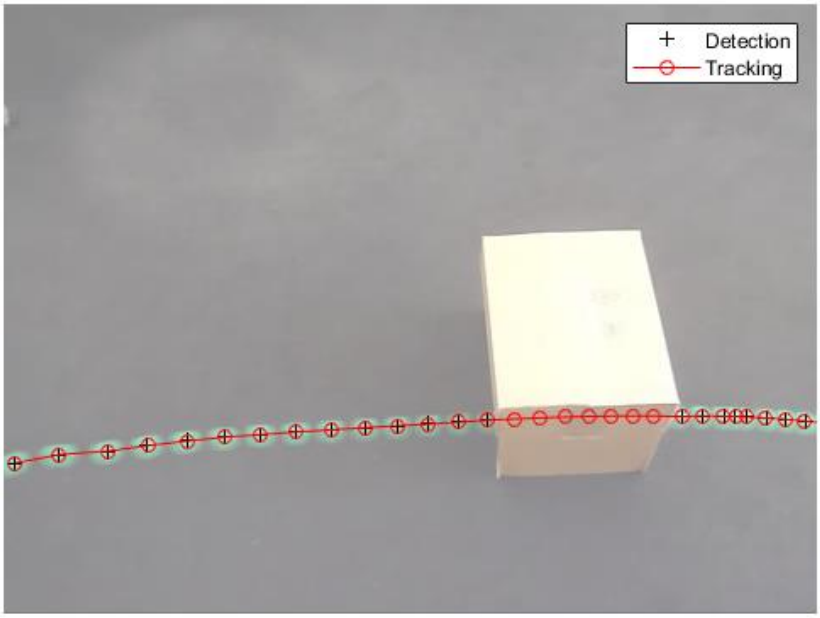
```
function trackSingleObject(param)
  % Create utilities used for reading video, detecting moving
objects,
  % and displaying the results.
  utilities = createUtilities(param);

  isTrackInitialized = false;
  while ~isDone(utilities.videoReader)
    frame = readFrame();

    % Detect the ball.
    [detectedLocation,           isObjectDetected]           =
detectObject(frame);

    if ~isTrackInitialized
      if isObjectDetected
        % Initialize a track by creating a Kalman filter when
the ball is
        % detected for the first time.
        initialLocation     =     computeInitialLocation(param,
detectedLocation);
        kalmanFilter                                          =
configureKalmanFilter(param.motionModel, ...
          initialLocation, param.initialEstimateError, ...
          param.motionNoise, param.measurementNoise);
        isTrackInitialized = true;
        trackedLocation         =         correct(kalmanFilter,
detectedLocation);
        label = 'Initial';
      else
        trackedLocation = [];
        label = '';
      end

    else
      % Use the Kalman filter to track the ball.
      if isObjectDetected % The ball was detected.
        %  Reduce  the  measurement  noise  by  calling  predict
followed by
        % correct.
        predict(kalmanFilter);
        trackedLocation         =         correct(kalmanFilter,
detectedLocation);
        label = 'Corrected';
      else % The ball was missing.
        % Predict the ball's location.
        trackedLocation = predict(kalmanFilter);
        label = 'Predicted';
      end
    end
    annotateTrackedObject();
  end % while
  showTrajectory();
end
```

Υπάρχουν δύο σενάρια για τα οποία το Kalmlan Filter μας είναι χρήσιμο.

- Όταν ανιχνευθεί η μπάλα, το φίλτρο Kalman προβλέπει αρχικά την κατάστασή του στο τρέχον πλαίσιο βίντεο και στη συνέχεια, χρησιμοποιεί τη θέση αντικειμένου που εντοπίστηκε πρόσφατα για να διορθώσει την κατάσταση του. Αυτό παράγει μια φιλτραρισμένη θέση.
- Όταν η μπάλα λείπει, το φίλτρο Kalman βασίζεται αποκλειστικά στην προηγούμενη του κατάσταση για να προβλέψει την τρέχουσα θέση της μπάλας.

Για να δούμε την τροχιά της μπάλας με την επικάλυψη όλων των πλαισίων βίντεο χρησιμοποιούμε τον κώδικα:

```
param = getDefaultParameters ();   % λαμβάνει τη διαμόρφωση Kalman που
λειτουργεί καλά
                                   % για αυτό το παράδειγμα

trackSingleObject (param);  % εμφάνιση των αποτελεσμάτων
```



Εξερεύνηση των επιλογών στη διαμόρφωση του φίλτρου Kalman.

Μπορεί να παρουσιαστεί σαν μεγάλη πρόκληση η διαμόρφωση του φίλτρου Kalman κ ο λόγος είναι ότι απαιτεί συχνά πειραματισμό έτσι ώστε να προκύψει ένα σύνολο κατάλληλων παραμέτρων διαμόρφωσης.

Η trackSingleObject λειτουργία που ορίζεται παραπάνω βοηθά να ερευνήσουμε τις διάφορες επιλογές διαμόρφωσης που προσφέρει η configureKalmanFilter μέθοδος.

Η configureKalmanFilter μέθοδος επιστρέφει ένα αντικείμενο φίλτρου Kalman η οποία απαιτεί 5 παραμέτρους σαν input

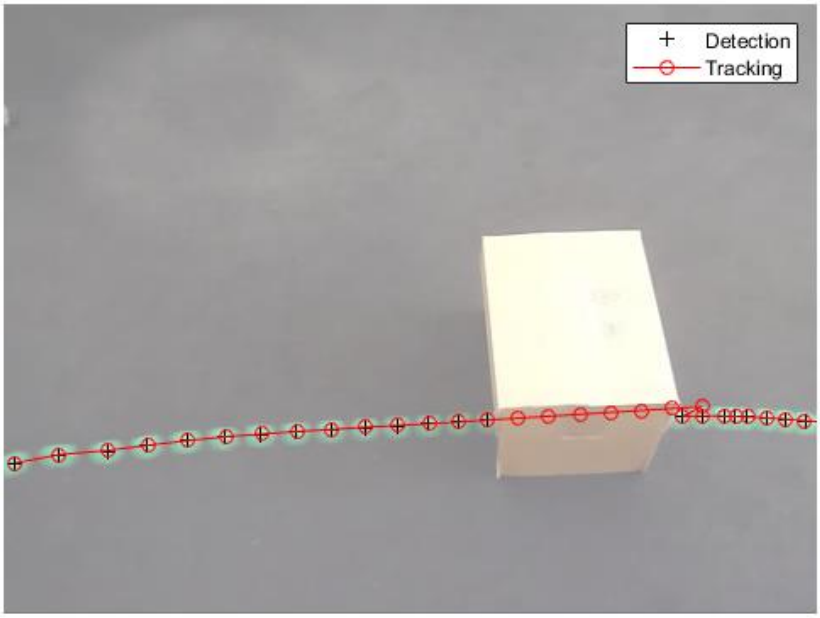
kalmanFilter = configureKalmanFilter (MotionModel, InitialLocation,

InitialEstimateError, MotionNoise, MeasurementNoise)

Η ρύθμιση **MotionModel** πρέπει να αντιστοιχεί στα φυσικά χαρακτηριστικά της κίνησης του αντικειμένου. Μπορούμε να το ρυθμίσουμε είτε σε μοντέλο σταθερής ταχύτητας είτε σταθερής επιτάχυνσης.

Το ακόλουθο παράδειγμα απεικονίζει τις συνέπειες της υποεπιλογικής επιλογής.

```
param = getDefaultParameters();            % get
parameters that work well
param.motionModel = 'ConstantVelocity'; % switch
from ConstantAcceleration
                                           %       to
ConstantVelocity
% After switching motion models, drop noise
specification entries
% corresponding to acceleration.
param.initialEstimateError                          =
param.initialEstimateError(1:2);
param.motionNoise                                   =
param.motionNoise(1:2);

trackSingleObject(param); % visualize the results
```



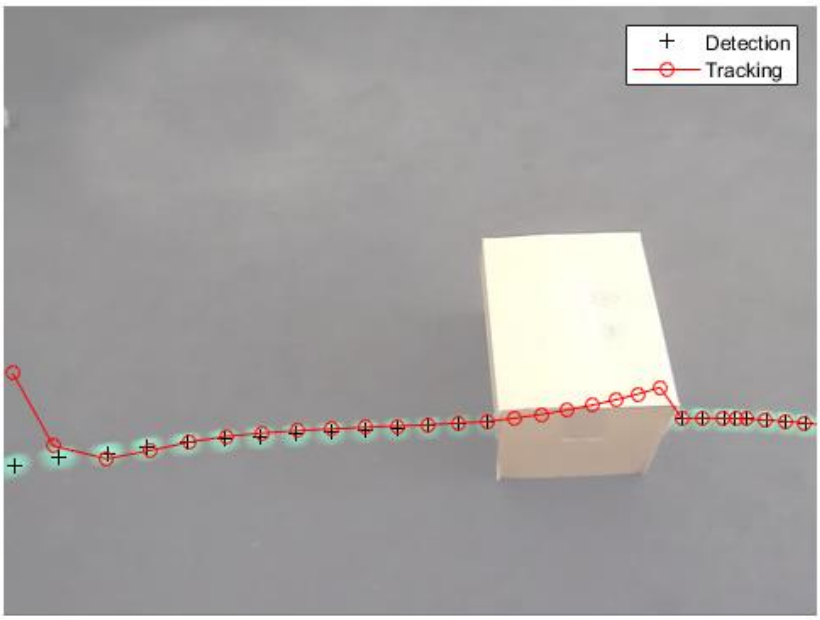
Αξιοσημείωτο είναι ότι η σφαίρα εμφανίστηκε σε ένα σημείο αρκετά διαφορετικό από το προβλεπόμενο σημείο. Από την στιγμή που η μπάλα ξεκίνησε την πορεία της υπέστη μια επιβράδυνση λόγο της τριβής του χαλιού. Επομένως καταλαβαίνουμε ότι το μοντέλο σταθερής επιτάχυνσης ήταν μια καλύτερη επιλογή.

Διατηρώντας το μοντέλο της σταθερής ταχύτητας δεν θα πάρουμε μη βέλτιστα αποτελέσματα ανεξάρτητα από το εύρος των τιμών που δίνουμε.Τυπικά θα θέταμε μια μεταβλητή InitialLocation σαν είσοδο στην τοποθεσία που πρωτοπαρατηρούμε το αντικείμενο.

Θα ρυθμίσουμε επίσης τον αρχικό δείκτη **EstimateError** σε μεγάλες τιμές, αφού η αρχική κατάσταση μπορεί να είναι πολύ θορυβώδης δεδομένου ότι προέρχεται από μία μόνο ανίχνευση.

Το παρακάτω σχήμα δείχνει την επίδραση της εσφαλμένης ρύθμισης αυτών των παραμέτρων:

```
param = getDefaultParameters();  % get parameters
that work well
param.initialLocation  =  [0,  0];    %  location
that's not based on an actual detection
param.initialEstimateError = 100*ones(1,3); % use
relatively small values

trackSingleObject(param); % visualize the results
```

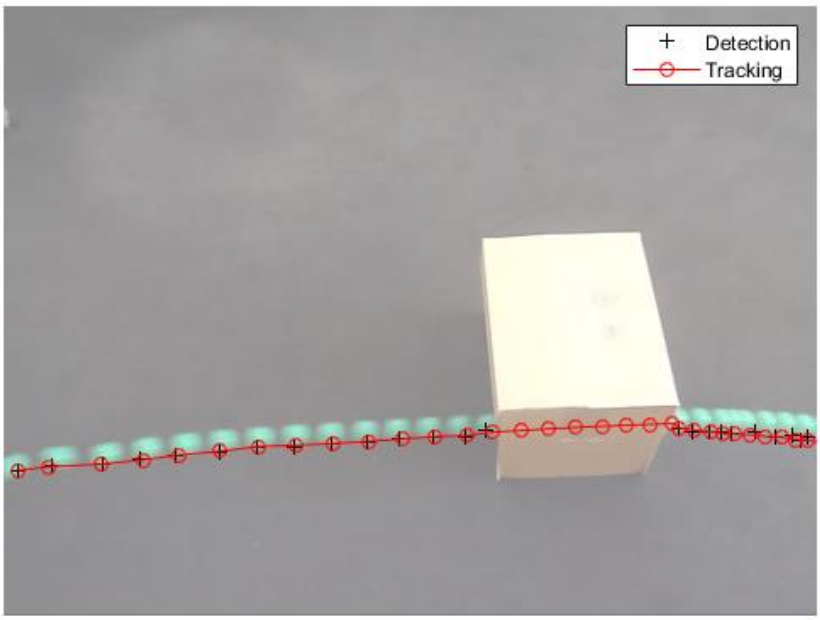


Με τις αλλαγμένες τιμές παραμέτρωην χρειάστηκαν μερικά βήματα πριν επιστρέψουν οι τοποθεσίες απο το Kalman φίλτρο το οποίο συμπέφτει με την πραγματική τροχιά του αντικειμένου.

Οι τιμές για το **MeasurementNoise** θα πρέπει να επιλέγονται με βάση την ακρίβεια του ανιχνευτή. Ρυθμίζοντας το θόρυβο μέτρησης σε μεγαλύτερες τιμές έχουμε σαν αποτέλεσμα μια λιγότερο ακριβή ανίχνευση.

Το ακόλουθο παράδειγμα απεικονίζει τις θορυβώδεις ανιχνεύσεις ενός κακώς καθορισμένου ορίου τμηματοποίησης. Η αύξηση του θορύβου μέτρησης αναγκάζει το φίλτρο Kalman να βασιστεί περισσότερο στην εσωτερική του κατάσταση παρά στις εισερχόμενες μετρήσεις και έτσι αντισταθμίζει το θόρυβο ανίχνευσης.

```
param = getDefaultParameters();
param.segmentationThreshold = 0.0005; % smaller
value resulting in noisy detections
param.measurementNoise      = 12500;  % increase
the value to compensate
                                      % for the
increase in measurement noise

trackSingleObject(param); % visualize the results
```



Συνήθως τα αντικείμενα στον πραγματικό κόσμο δεν κινούνται με σταθερή ταχύτητα ή επιτάχυνση. Με την χρήση της **MotionNoise** καθορίζουμε την ποσότητα απόκλισης απο το ιδανικό μοντέλο κίνησης.

Όταν αυξάνουμε τον θόρυβο κίνησης το φίλτρο Kalman βασίζεται περισσότερο στις εισερόχμενες μετρήσεις παρα στην εσωτερική του κατάσταση.

Παρακολούθηση πολλαπλών αντικειμένων με χρήση του Kalman φίλτρου.

Όπως είδαμε κ στο πρώτο μέρος της διπλωματικές η παρακολούθηση πολλαπλών αντικειμένων δημιουργεί αρκετές πρόσθετες προκλήσεις:

- Πολλαπλές ανιχνεύσεις πρέπει να συσχετίζονται με τα σωστά κομμάτια
- Πρέπει να χειριστούμε ανάλογα νέα αντικείμενα που θα εμφανίζονται σε μια σκηνή
- Η ταυτότητα του αντικειμένου πρέπει να διατηρείται όταν πολλαπλά αντικείμενα συγχωνευθούν σε μία μόνο ανίχνευση

Το vision.**KalmanFilterαντικείμενο** μαζί με τη **assignDetectionsToTracks** λειτουργία μπορεί να βοηθήσει στην επίλυση των προβλημάτων αυτών

- Αντιστοίχιση ανιχνεύσεων σε κομμάτια
- Προσδιορισμός αν μια ανίχνευση αντιστοιχεί σε ένα νέο αντικείμενο, με άλλα λόγια, τη δημιουργία κομματιών
- Ακριβώς όπως στην περίπτωση ενός αποκλεισμένου αντικειμένου, η πρόβλεψη μπορεί να χρησιμοποιηθεί για να βοηθήσει σε ξεχωριστά αντικείμενα που βρίσκονται κοντά μεταξύ τους

Για να μάθετε περισσότερα σχετικά με τη χρήση του φίλτρου Kalman για την παρακολούθηση πολλών αντικειμένων,

Χρήση λειτουργιών που εφαρμόζονται στο παράδειγμα

Οι λειτουργίες χρησιμότητας χρησιμοποιήθηκαν για την ανίχνευση των αντικειμένων και την εμφάνιση των αποτελεσμάτων. Αυτή η ενότητα επεξηγεί τον τρόπο με τον οποίο υλοποιήθηκαν αυτές οι λειτουργίες:

Απόκτηση των προκαθορισμένων παραμέτρων για την δημιουργία Kalman Filter και την τμηματοποίηση της σφαίρας.

```
function param = getDefaultParameters
  param.motionModel                              =
'ConstantAcceleration';
  param.initialLocation          = 'Same as first
detection';
  param.initialEstimateError  = 1E5 * ones(1, 3);
  param.motionNoise           = [25, 10, 1];
  param.measurementNoise      = 25;
  param.segmentationThreshold = 0.05;
end
```

Ανάγνωση του επόμενου video frame στο αρχείο

```
function frame = readFrame()
  frame = step(utilities.videoReader);
end
```

Ανίχνευση και καταγραφή της σφαίρας στο βίντεο

```
function showDetections()
  param = getDefaultParameters();
  utilities = createUtilities(param);
  trackedLocation = [];

  idx = 0;
  while ~isDone(utilities.videoReader)
    frame = readFrame();
    detectedLocation = detectObject(frame);
    % Show the detection result for the current
video frame.
    annotateTrackedObject();

    % To highlight the effects of the measurement
noise, show the detection
    % results for the 40th frame in a separate
figure.
    idx = idx + 1;
    if idx == 40
      combinedImage                                =
max(repmat(utilities.foregroundMask,    [1,1,3]),
frame);
      figure, imshow(combinedImage);
    end
  end % while

  % Close the window which was used to show
individual video frame.
  uiscopes.close('All');
end
```

Ανίχνευση της σφαίρας στο συγκεκριμένο βίντεο frame

```
function [detection, isObjectDetected] = detectObject(frame)
  grayImage = rgb2gray(frame);
  utilities.foregroundMask = step(utilities.foregroundDetector, grayImage);
  detection = step(utilities.blobAnalyzer, utilities.foregroundMask);
  if isempty(detection)
    isObjectDetected = false;
  else
    % To simplify the tracking process, only use the first detected object.
    detection = detection(1, :);
    isObjectDetected = true;
  end
end
```

Ένδειξη της κύριας ανίχνευσης και παρακολούθηση των αποτελεσμάτων

```
function annotateTrackedObject()
  accumulateResults();
  % Combine the foreground mask with the current video frame in order to
  % show the detection result.
  combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);

  if ~isempty(trackedLocation)
    shape = 'circle';
    region = trackedLocation;
    region(:, 3) = 5;
    combinedImage = insertObjectAnnotation(combinedImage, shape, ...
      region, {label}, 'Color', 'red');
  end
  step(utilities.videoPlayer, combinedImage);
end
```

Ένδειξη της τροχιάς της μπάλας, επικαλύπτοντας όλα τα πλαίσια βίντεο το ένα πάνω στο άλλο.

```
function showTrajectory
  % Close the window which was used to show
individual video frame.
  uiscopes.close('All');

  % Create a figure to show the processing
results for all video frames.
  figure;
imshow(utilities.accumulatedImage/2+0.5);     hold
on;
  plot(utilities.accumulatedDetections(:,1), ...
    utilities.accumulatedDetections(:,2), 'k+');

  if ~isempty(utilities.accumulatedTrackings)
    plot(utilities.accumulatedTrackings(:,1), ...
      utilities.accumulatedTrackings(:,2),     'r-
o');
    legend('Detection', 'Tracking');
  end
end
```

Συσώρευση των πλαισίων βίντεο, ανίχνευση και εντοπισμός τοποθεσιών για να δουμε την τροχιά της μπάλας.

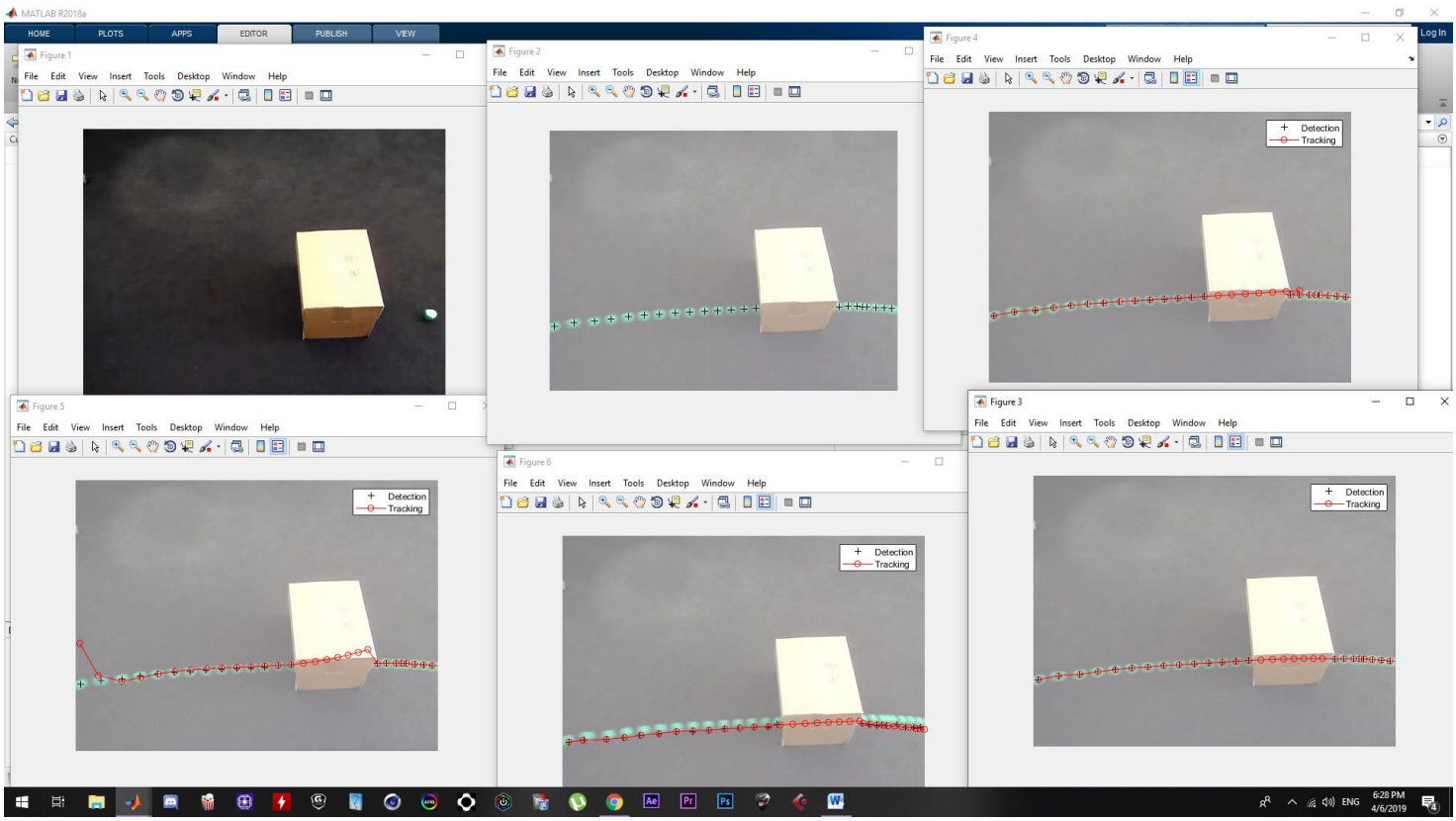
```
function accumulateResults()
  utilities.accumulatedImage                    =
max(utilities.accumulatedImage, frame);
  utilities.accumulatedDetections ...
    =            [utilities.accumulatedDetections;
detectedLocation];
  utilities.accumulatedTrackings  ...
    =             [utilities.accumulatedTrackings;
trackedLocation];
end
```

Επιλογή αρχικής θέσης που χρησιμοποιείται από το φίλτρο Kalman.

```
function accumulateResults()
  utilities.accumulatedImage                    =
max(utilities.accumulatedImage, frame);
  utilities.accumulatedDetections ...
    =           [utilities.accumulatedDetections;
detectedLocation];
  utilities.accumulatedTrackings  ...
    =            [utilities.accumulatedTrackings;
trackedLocation];
End
```

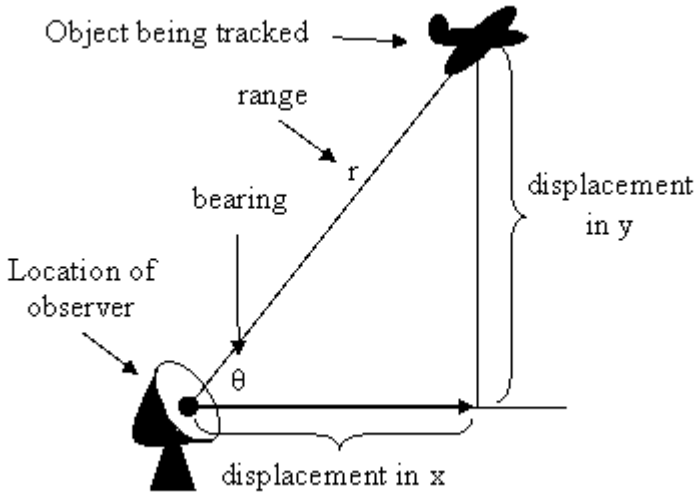
Δημιουργία βοηθητικών προγραμμάτων για την ανάγνωση βίντεο, την ανίχνευση κινούμενων αντικειμένων και την εμφάνιση των αποτελεσμάτων.

```
function utilities = createUtilities(param)
  % Create System objects for reading video,
displaying video, extracting
  %   foreground,   and   analyzing   connected
components.
  utilities.videoReader                         =
vision.VideoFileReader('singleball.mp4');
  utilities.videoPlayer                         =
vision.VideoPlayer('Position',
[100,100,500,400]);
  utilities.foregroundDetector                  =
vision.ForegroundDetector(...
    'NumTrainingFrames',  10,  'InitialVariance',
param.segmentationThreshold);
  utilities.blobAnalyzer                        =
vision.BlobAnalysis('AreaOutputPort', false, ...
    'MinimumBlobArea',  70,  'CentroidOutputPort',
true);

  utilities.accumulatedImage      = 0;
  utilities.accumulatedDetections = zeros(0, 2);
  utilities.accumulatedTrackings  = zeros(0, 2);
end

end
```


MATLAB R2018a
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Figure 1
Figure 2
Figure 4
Figure 5
Figure 6
Figure 3
File Edit View Insert Tools Desktop Window Help
Detection
Tracking
ENG
6:28 PM
4/6/2019

Βελτίωση του αλγορίθμου με εφαρμογή κ χρήση του Kalman Filter σε Simulink

Μία από τις πρώτες εφαρμογές του Extended Kalman Filter ήταν να επιλύσει το πρόβλημα της ανίχνευσης ιπτάμενων αντικειμένων. Το βασικό πρόβλημα παρουσιάζεται παρακάτω:



Σε κάθε χρονική στιγμή το αεροπλάνο έχει ένα δεδομένο εύρος και καθορίζεται από τον παρατηρητή. Συχνά ο παρατηρητής θεωρείται η θέση ενός ραντάρ που εντοπίζει το αντικείμενο. Η εμβέλεια και το έδρανο δημιουργούνται από μετατοπίσεις (δηλ. Αποστάσεις από τον παρατηρητή) και στις δύο κατευθύνσεις x και y.

Το πρόβλημα της ανίχνευσης περιλαμβάνει την εκτίμηση όχι μόνο των μετατοπίσεων x και y του αντικειμένου αλλά και των ταχυτήτων του x και y. Αυτές οι τέσσερις καταστάσεις πρέπει να εκτιμηθούν δίνοντας μόνο θορυβώδεις μετρήσεις της εμβέλειας και του ρουλεμάν. Δεδομένου ότι οι μετατοπίσεις και οι ταχύτητες δεν σχετίζονται γραμμικά με το εύρος αυτό είναι ένα ιδανικό πρόβλημα για λύση χρησιμοποιώντας ένα Extended Kalman Filter.

Ο αλγόριθμος του Extended Kalman Filter απαιτεί τον υπολογισμό Jacobian πινάκων για τις εξισώσεις κατάστασης και μέτρησης. Αυτά έχουν τις ακόλουθες μορφές. Σε μια μικρή χρονική περίοδο η μετατόπιση μπορεί να θεωρηθεί ότι αλλάζει σύμφωνα με την προσέγγιση της πρώτης τάξης.

$$\hat{x}(k+1) = \begin{bmatrix} \hat{x}_{pos}(k+1) \\ \hat{x}_{vel}(k+1) \\ \hat{y}_{pos}(k+1) \\ \hat{y}_{vel}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{pos}(k) \\ \hat{x}_{vel}(k) \\ \hat{y}_{pos}(k) \\ \hat{y}_{vel}(k) \end{bmatrix} = F_k \hat{x}(k)$$

Η παραπάνω εξίσωση δηλώνει ότι σε μια μικρή μεταβολή του χρόνου μια θέση που μετατοπίζεται κατά Δt επί την ταχύτητα (στον χ κ y άξονα) και ότι αυτή η ταχύτητα παραμένει σταθερή κ στις δύο κατευθύνσεις x , y. Το F_k είναι ο απαιτούμενος Jacobian πίνακας.

Η εξίσωση ενημέρωσης της απόστασης είναι ελαφρώς πιο περίπλοκη - στηριζόμενη στη διαφοροποίηση μιας τριγωνομετρικής ταυτότητας. Το εύρος και το έδρανο σχετίζονται με τις μετακινήσεις x και y από τις εξισώσεις,

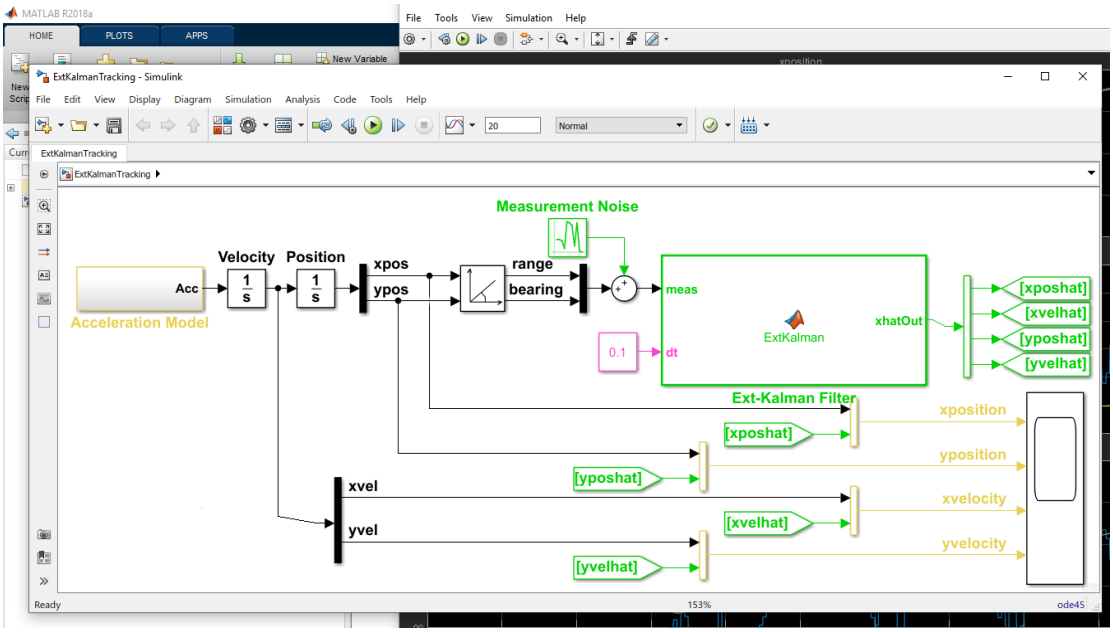
$$\hat{m}_k = \begin{bmatrix} \hat{r}_k \\ \hat{\theta}_k \end{bmatrix} = \begin{bmatrix} \sqrt{(\hat{x}_{pos})^2 + (\hat{y}_{pos})^2} \\ \arctan 2(\hat{y}_{pos} / \hat{x}_{pos}) \end{bmatrix}$$

Ο δείκτης δείκτης χρόνου k φθίνει από τον φορέα κατάστασης για να αποφευχθεί η συσσώρευση των εξισώσεων. Επομένως, η Jacobian για τις εξισώσεις μέτρησης δίνεται από

$$H_k = \left.\frac{\partial(\hat{m}_k)}{\partial \hat{x}}\right|_{\hat{x}} = \begin{bmatrix} \cos\left(\hat{\theta}_k\right) & 0 & \sin\left(\hat{\theta}_k\right) & 0 \\ -\sin\left(\hat{\theta}_k\right)/\hat{r}_k & 0 & \cos\left(\hat{\theta}_k\right)/\hat{r}_k & 0 \end{bmatrix}$$

Εφαρμογή του Kalman Filter σε Simulink

Ένα μοντέλο Simulink που υλοποιεί το βασικό πρόβλημα παρακολούθησης που αναφέρθηκε παραπάνω και το οποίο χρησιμοποιεί ένα εκτεταμένο φίλτρο Kalman για την εκτίμηση της τροχιάς του αντικειμένου παρουσιάζεται παρακάτω

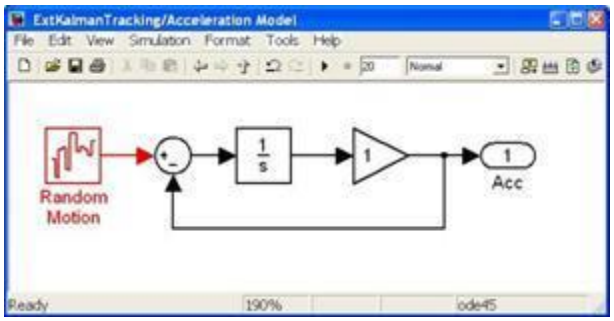


Υπάρχουν δύο βασικά μέρη σε αυτό το μοντέλο: πρώτον τα μπλοκ που μοντελοποιούν την πραγματική τροχιά του αντικειμένου που παρακολουθείται και δεύτερον το Εκτεταμένο φίλτρο Kalman που χρησιμοποιήθηκε για την εκτίμηση της τροχιάς του αντικειμένου από τα δεδομένα που μετρήθηκαν. Καθένα από αυτά περιγράφεται λεπτομερέστερα στις παρακάτω ενότητες.

Η πραγματική πορεία

Τα μαύρα μπλόκς χρησιμοποιούνται για να μοντελοποιήσουν την πραγματική τροχιά ενός αντικειμένου που πετάει σε δισδιάστατο χώρο. Οι ιδιότητες και το έδρανο υπολογίζονται από τις μετατοπίσεις x και y, οι οποίες παράγονται με την ενσωμάτωση των ταχυτήτων, οι οποίες με τη σειρά τους παράγονται με την ενσωμάτωση επιταχύνσεων.

Οι επιταχύνσεις δημιουργούνται από το μοντέλο επιτάχυνσης που παρουσιάζεται στο σχήμα παρακάτω .



Σε αυτό το παράδειγμα ότι το αντικείμενο αρχίζει στα βορειοδυτικά και ταξιδεύει ανατολικά στα 100 m / s. Αυτό αντιστοιχεί στις ρυθμίσεις που δίνονται στον παρακάτω πίνακα

Ποσότητα	αξία
Αρχική x θέση	-1000 μ
Αρχική θέση y	-1000 μ
Αρχική x ταχύτητα	100 m / s
Αρχική ταχύτητα y	0 m / s
x θορύβου μετατόπισης	10
y θορύβου μετατόπισης	0.1

Το εκτεταμένο φίλτρο Kalman

Το εκτεταμένο φίλτρο Kalman χρησιμοποιεί έναν αλγόριθμο διόρθωσης πρόβλεψης για την εκτίμηση των μη μετρημένων καταστάσεων μιας διακριτής διαδικασίας.

Για το πρόβλημα παρακολούθησης που εξετάζεται, τα μετρηθέντα δεδομένα είναι η πραγματική περιοχή του αντικειμένου και το έδρανο είναι με μηδενικό μέσον θόρυβο Gauss και δειγματοληψία σε διαστήματα των 0,1 s. Ο θόρυβος της περιοχής έχει διακύμανση 50, ενώ ο θόρυβος του ρουλεμάν έχει διακύμανση 0,005.

Το εκτεταμένο φίλτρο Kalman έχει υλοποιηθεί χρησιμοποιώντας ένα μπλοκ ενσωματωμένης λειτουργίας MATLAB . Το μπλοκ είναι διακεκριμένο με χρόνο δειγματοληψίας 0,1 δευτερόλεπτα. Ο κωδικός για το μπλοκ φαίνεται παρακάτω.

Σημείωση: το φίλτρο έχει εσκεμμένα αρχικοποιηθεί με λανθασμένα δεδομένα για να φανεί ότι η πραγματική τροχιά η οποία δεν είναι γνωστή εκ των προτέρων.

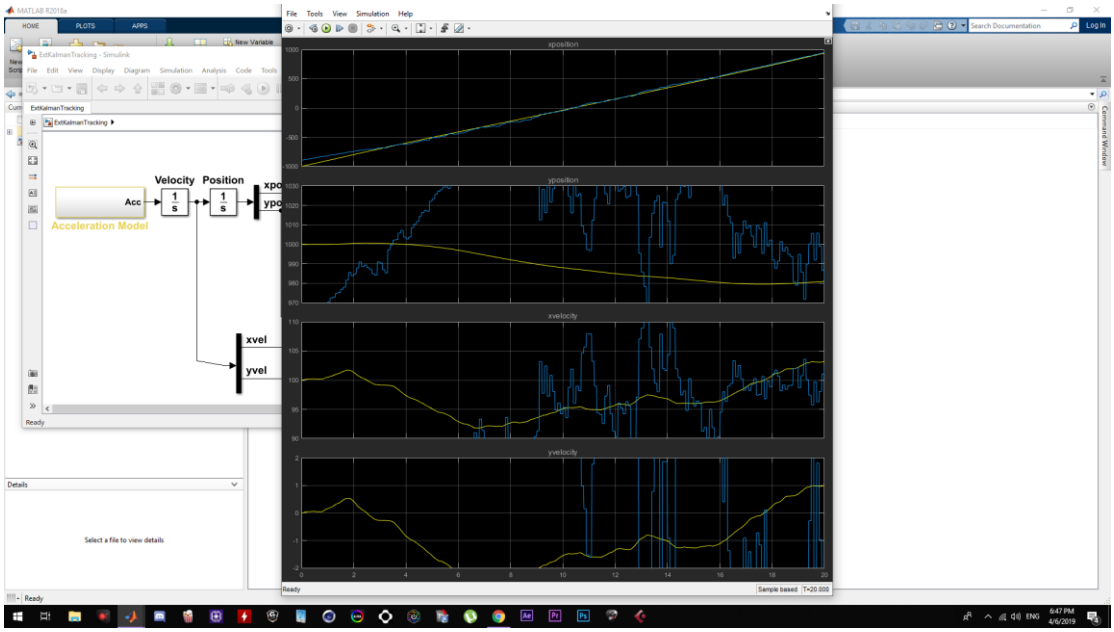
```
function xhatOut = ExtKalman(meas,dt)
% This Embedded MATLAB Function implements an
extended Kalman filter used
% for object tracking.
%
% The states of the process are given by
%   x   =   [x_position;   x_velocity;   y_position;
y_velocity];
%
% and the measurements are given by
% y = [range; bearing]
%
% where
% range = sqrt(x_position^2 + y_position^2)
% bearing = atan2(y_position/x_position)

%          Author:          Phil          Goddard
(phil@goddardconsulting.ca)
% Date: Q2, 2011.
```

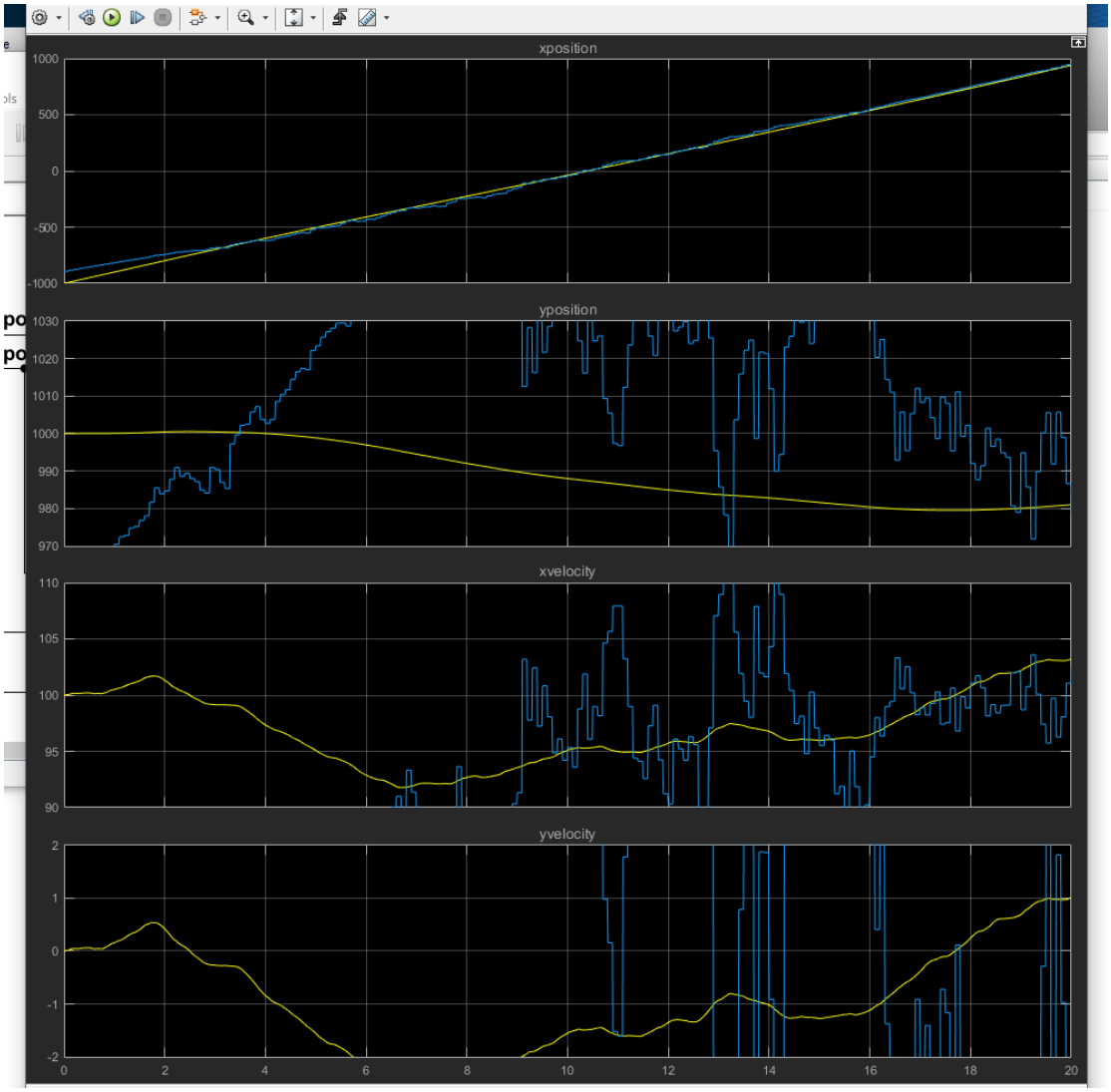
```
% Define storage for the variables that need to
persist
% between time periods.
persistent P xhat Q R
if isempty(P)
   % First time through the code so do some
initialization
   xhat = [-900;80;950;20];
   P = zeros(4,4);
   Q = diag([0 .1 0 .1]);
   R = diag([50^2 0.005^2]);
end
% Calculate the Jacobians for the state and
measurement equations
F = [1 dt 0 0;0 1 0 0;0 0 1 dt;0 0 0 1];
rangeHat = sqrt(xhat(1)^2+xhat(3)^2);
bearingHat = atan2(xhat(3),xhat(1));
yhat = [rangeHat; bearingHat];
H = [cos(bearingHat) 0 sin(bearingHat) 0;
     -sin(bearingHat)/rangeHat                    0
cos(bearingHat)/rangeHat 0];
% Propogate the state and covariance matrices
xhat = F*xhat;
P = F*P*F' + Q;
% Calculate the Kalman gain
K = P*H'/(H*P*H' + R);
% Calculate the measurement residual
resid = meas - yhat;
% Update the state and covariance estimates
xhat = xhat + K*resid;
P = (eye(size(K,1))-K*H)*P;
% Post the results
xhatOut = xhat;
```

Αποτελέσματα προσομοίωσης

Κάθε μία από τις πραγματικές και τις εκτιμώμενες καταστάσεις εμφανίζεται σε ξεχωριστό άξονα, με την πραγματική τροχιά να εμφανίζεται με μπλε χρώμα και την εκτιμώμενη τροχιά με κόκκινο χρώμα.

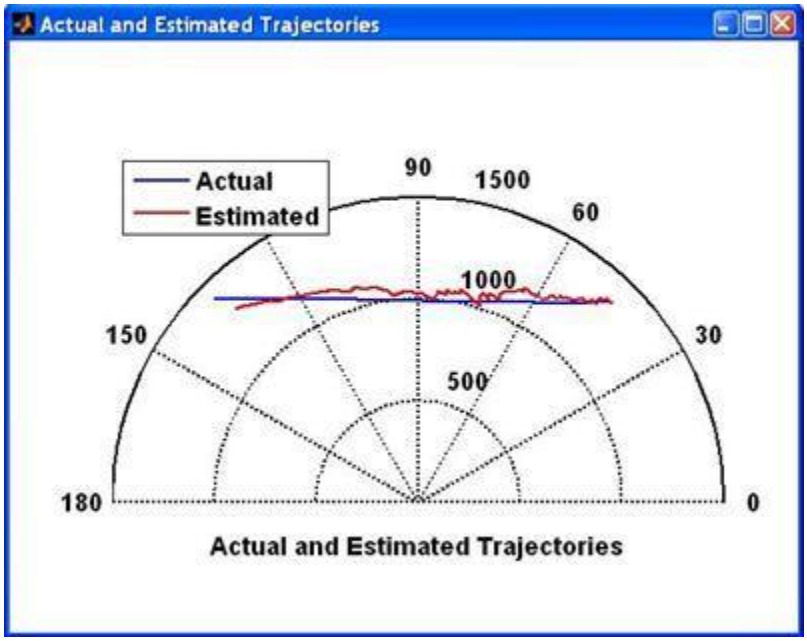
Οι εκτιμήσεις έχουν εσκεμμένα αρχικοποιηθεί με μη ακριβείς τιμές. Αυτό σημαίνει ότι το φίλτρο χρειάζεται κάποιο χρόνο για να συγκλίνει σε αποδεκτή εκτίμηση. Ωστόσο, μετά από 10-15 δευτερόλεπτα, οι εκτιμήσεις παρακολουθούν τις πραγματικές τροχιές σε λογική ακρίβεια. Οι εκτιμήσεις θα μπορούσαν να βελτιωθούν περαιτέρω με τον συντονισμό των εκτιμήσεων συνεισφοράς θορύβου πριν (και κατά τη διάρκεια) που εφαρμόζει το φιλτράρισμα.




xposition
1000
500
0
-500
-1000
yposition
1030
1020
1010
1000
990
980
970
xvelocity
110
105
100
95
90
yvelocity
2
1
0
-1
-2
0
2
4
6
8
10
12
14
16
18
20

Πολλές φορές είναι χρήσιμο να πραγματοποιούμε την παρακολούθηση της τροχιάς σε πολικό επίπεδο. Όπως βλέπουμε παρακάτω το αντικείμενο κινείται από τα βορειοδυτικά προς τα ανατολικά με κατάληξη στα βορειοανατολικά.

Αρχίζοντας από τα 1400 μέτρα το αεροπλάνο έρχεται σε απόσταση 1 χιλιομέτρου από τον παρατηρητή κ μετά κινείται ξανά σε απόσταση 1400 μέτρων. Απόδειξη ότι το φίλτρο παίρνει ένα σύντομο χρονικό διάστημα ώστε να εκτιμήσει σωστά την θέση του αντικειμένου κ στην συνέχεια ακολουθεί με ακρίβεια την τροχιά του



Βιβλιογραφία

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

13. https://la.mathworks.com/videos/introduction-to-kalman-filters-for-object-tracking-79674.html?requestedDomain= - introduction to kalman filter

14. http://www.morethantechnical.com/2011/06/17/simple-kalman-filter-for-tracking-using-opencv-2-2-w-code/ - Simple calman Filter example

15. https://la.mathworks.com/help/vision/examples/using-kalman-filter-for-object-tracking.html?fbclid=IwAR0o87BOoK158gliiBzv-EHILWs4QryoFw4qK5oQdlho307GSMrpdlglafA#d120e4924- Kalman Filter Example

16. https://en.wikipedia.org/wiki/Kalman_filter -Kalman Filter Wiki

17. http://www.samos.aegean.gr/actuar/dlekkas/forecasting%20and%20sim/forecasting%20and%20simulation%204.pdf –Forecasting Simulation