Πανεπιστήμιο Δυτικής Αττικής

Μεταπτυχιακό Πρόγραμμα Σπουδών

Εφαρμοσμένα Πληροφοριακά Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Κίνησης Αντικειμένων με Raspberry- Pi και Python

Αθανάσιος Λιακόπουλος AIS-0126

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΜΑΙΟΣ 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανίχνευση Κίνησης αντικειμένων με Raspberry- Pi και Python

Λιακόπουλος Αθανάσιος
AIS-0126

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η ……Λιακόπουλος Αθανάσιος……., του ……Δημητρίου……, με αριθμό μητρώου …ais-0126…….. 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

Θα ήθελα να ευχαριστήσω θερμά τον καθηγητή κ. Ιωάννη Έλληνα για την καθοδήγηση του και την εμπιστοσύνη που έδειξε στο πρόσωπο μου στην εκπόνηση της παρούσας διπλωματικής εργασίας. Στη συνέχεια επιθυμώ να ευχαριστήσω όλους τους καθηγητές μου στο μεταπτυχιακό πρόγραμμα σπουδών με τίτλο ”Εφαρμοσμένα Πληροφοριακά Συστήματα” για τις πολύτιμες γνώσεις - εφόδια που θα με συνοδεύσουν στην επαγγελματική σταδιοδρομία μου. Κλείνοντας επιθυμώ να αφιερώσω τη παρούσα διπλωματική εργασία στους γονείς μου Δημήτρη και Σοφία ,των οποίων η υπομονή και αρωγή συνέβαλε τα μέγιστα στην μέχρι τώρα πορεία μου.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη αλγορίθμων - μοντέλων σε κώδικα Python και την εφαρμογή αυτών στην συσκευή Raspberry Pi 3 για την πολλαπλή ανάλυση και ανίχνευση αντικειμένων συγκεκριμένου ενδιαφέροντος.

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 άλλα.

ABSTRACT

The present thesis concerns the development of algorithms-models on Python Code and the application on a Raspberry Pi 3 for multi-object detection.

The following methods that took place for developing this project are based on Artificial Intelligence (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.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Εκμάθηση Έξυπνων Συστημάτων (Α.Ι)

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Τεχνητή Νοημοσύνη, Raspberry P, IoT Analytics , MultiObject Detection.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

Raspi: Raspberry Pi

GMM: Gaussian Mixture Model

ΕΙΣΑΓΩΓΗ

Raspberry 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 όσο 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.

Το 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, 1.2GHToz*
-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:

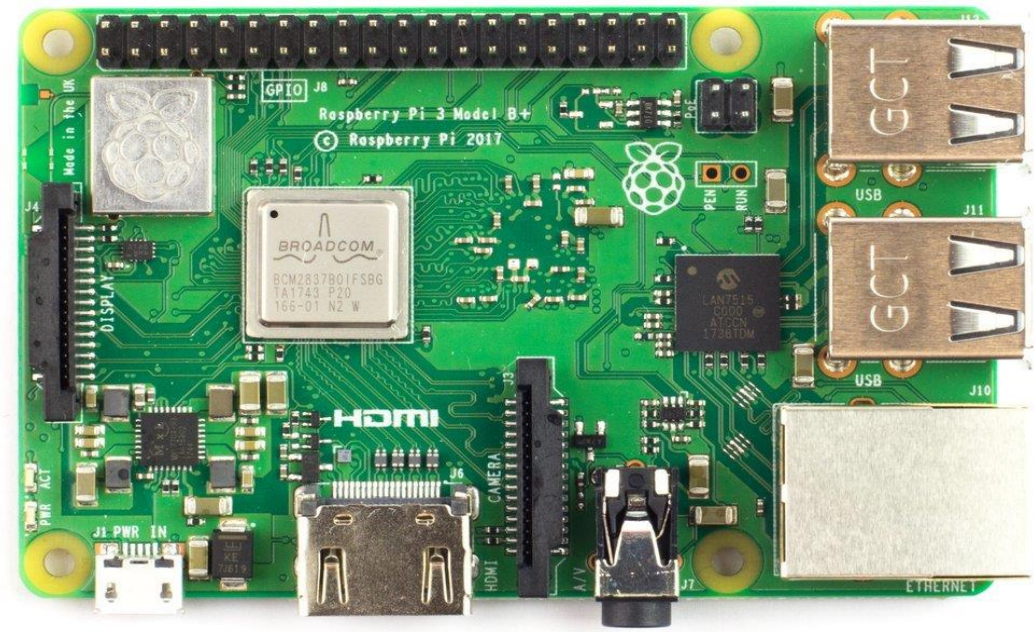
http://stratux.me/

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



1.1 Raspberry Pi 3 Model B

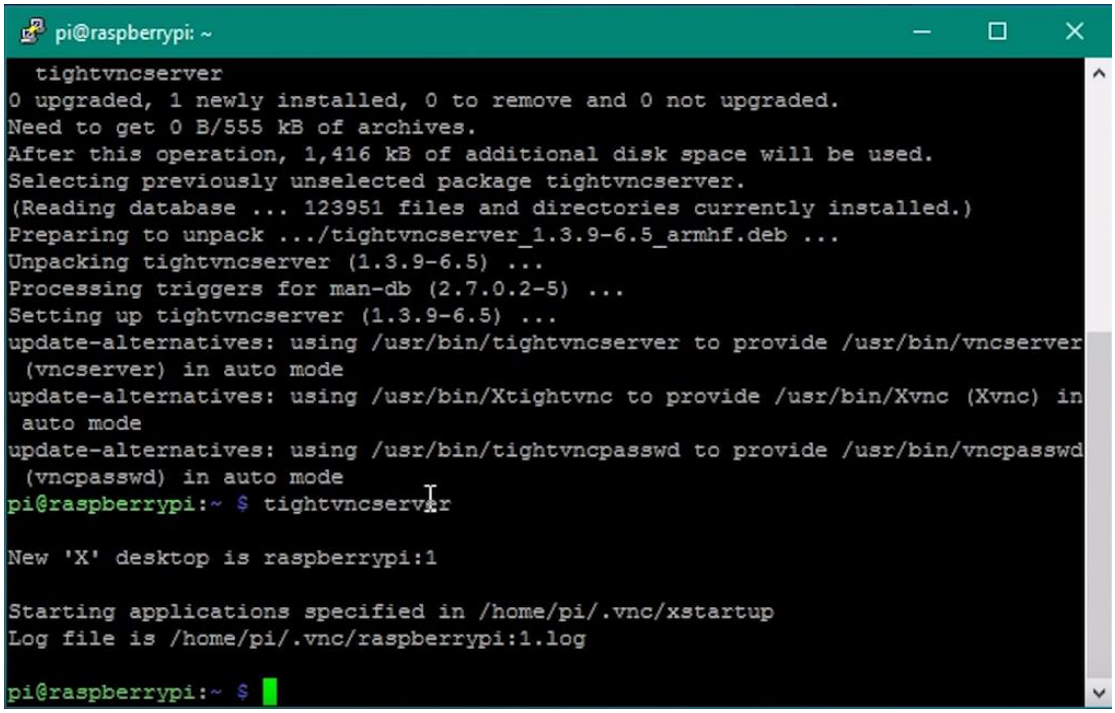
Multi-object Motion Detection με python και Debian

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

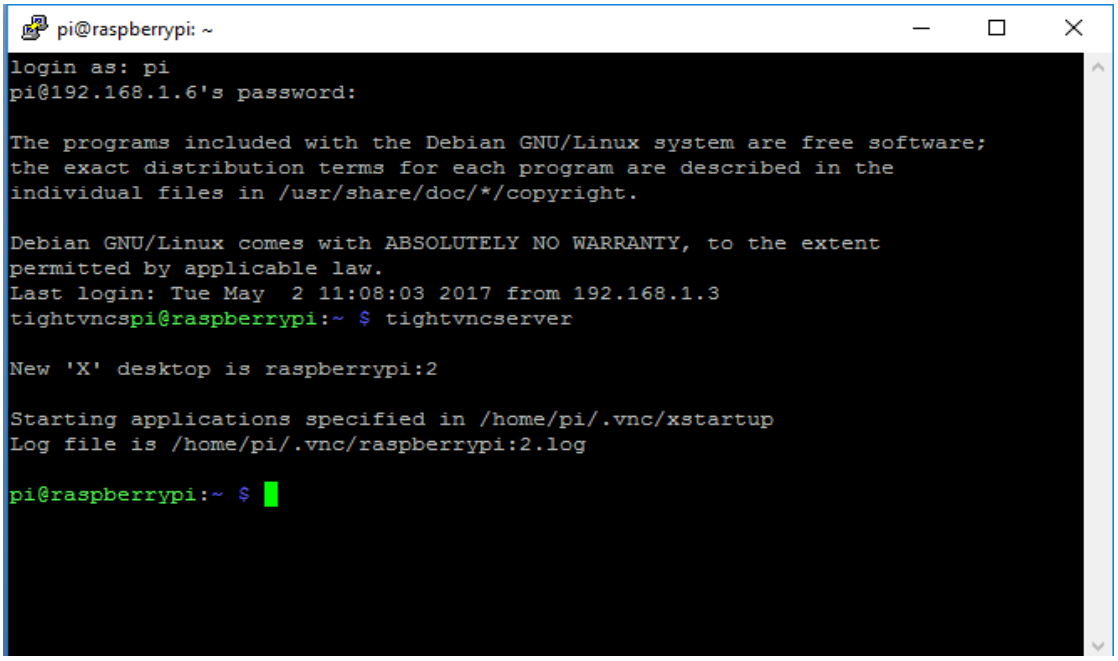


2.1 using putty to establish pi connection

Όταν ολοκληρωθεί η διαδικασία πατάμε την εντολή:

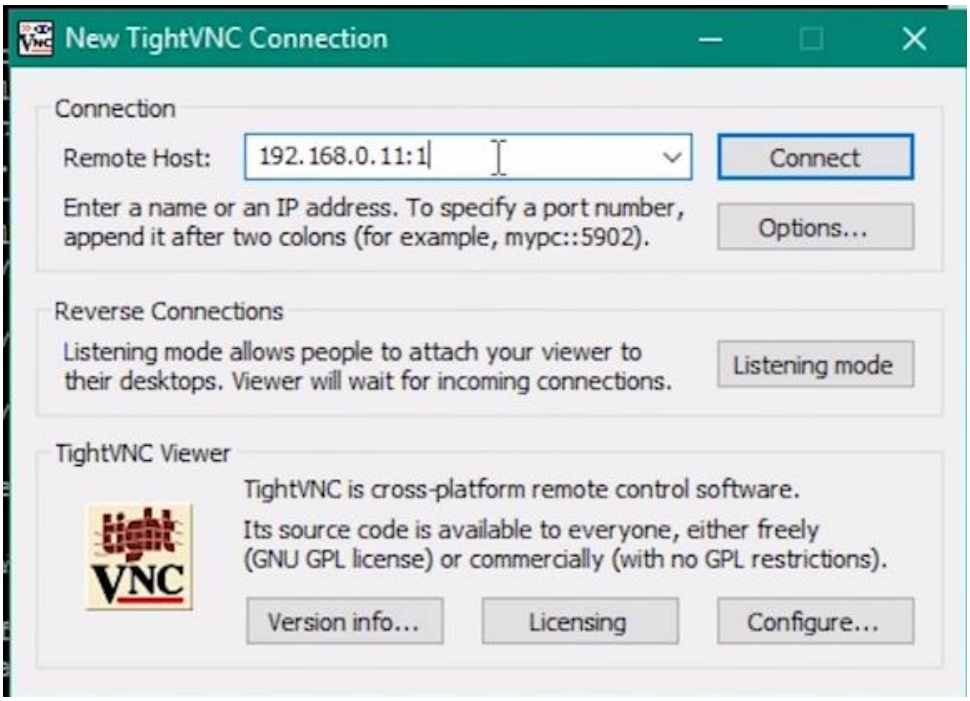
>>tightvncserver

Και παίρνουμε την ακόλουθη οθόνη**:**



2.2 using putty to establish pi connection (next)

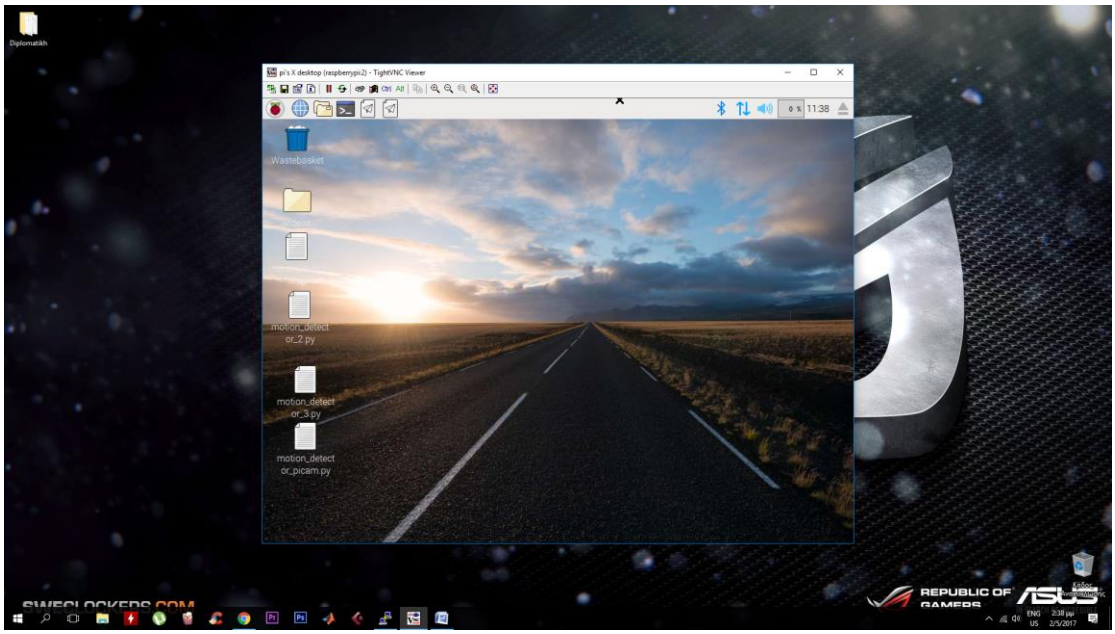
Είμαστε πλέον έτοιμοι να ανοίξουμε το tightvnc viewer και να κάνουμε σύνδεση remotely με την συσκευή μας.



2.3 using putty to establish pi connection

Πληκτρολογούμε την ip του raspi ακολουθούμενη 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.



2.4 Final Results

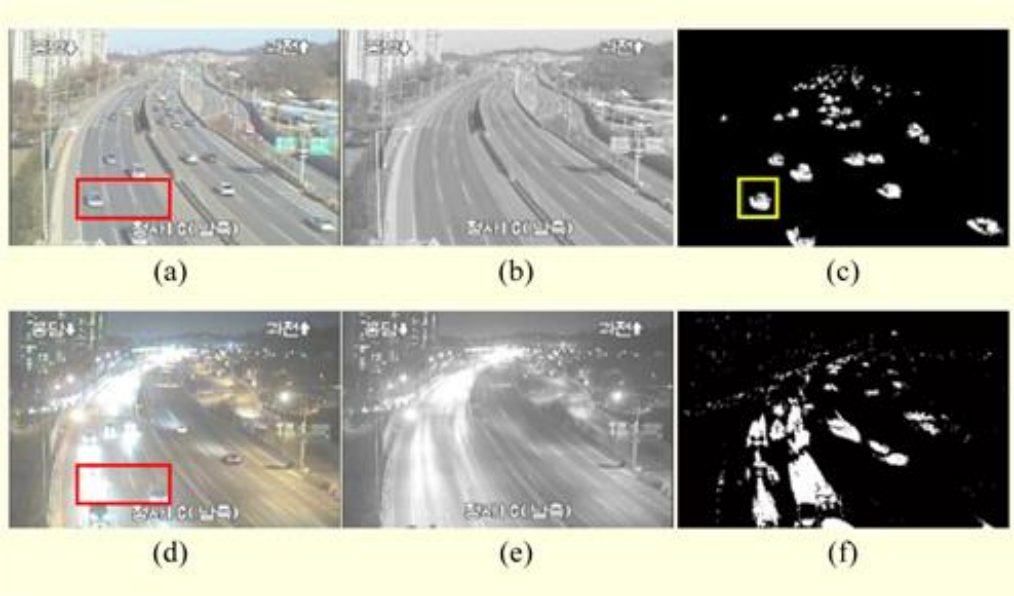
Foreground and Background

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 effects. Η κάμερα πρέπει να είναι στατική ώστε να αποφευχθούν 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 effects 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.

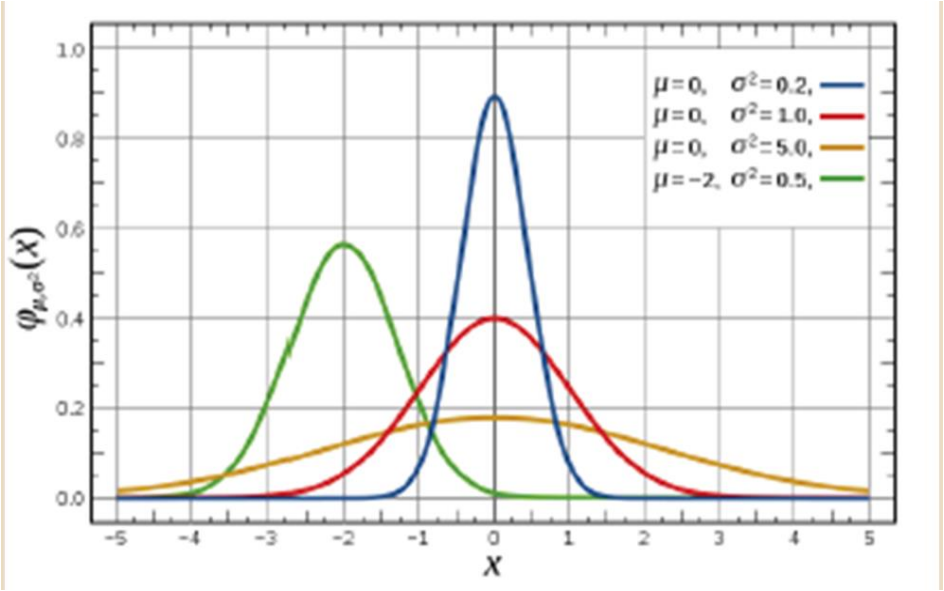


3.1 Foreground-background detection

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

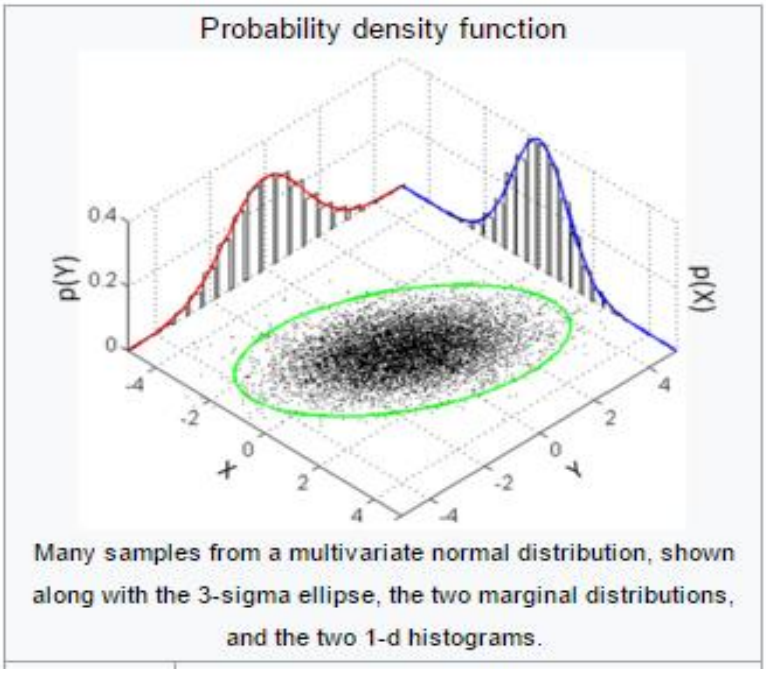
Normal or Gaussian Distribution – Univariate – One dimensional

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



4.1 Normal or Gaussian Distribution – Multivariate – Multidimensional

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



4.2 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 K 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 N είναι ο αριθμός της πολυπαραγοντικής τιμής διανομής, w η βαρύτητα κάθε τιμής.

Σε κάθε επανάληψη τα μείγματα αξιολογούνται και αν ένα pixel ταιριάζει σε κάποιο 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 προαιρετικές 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

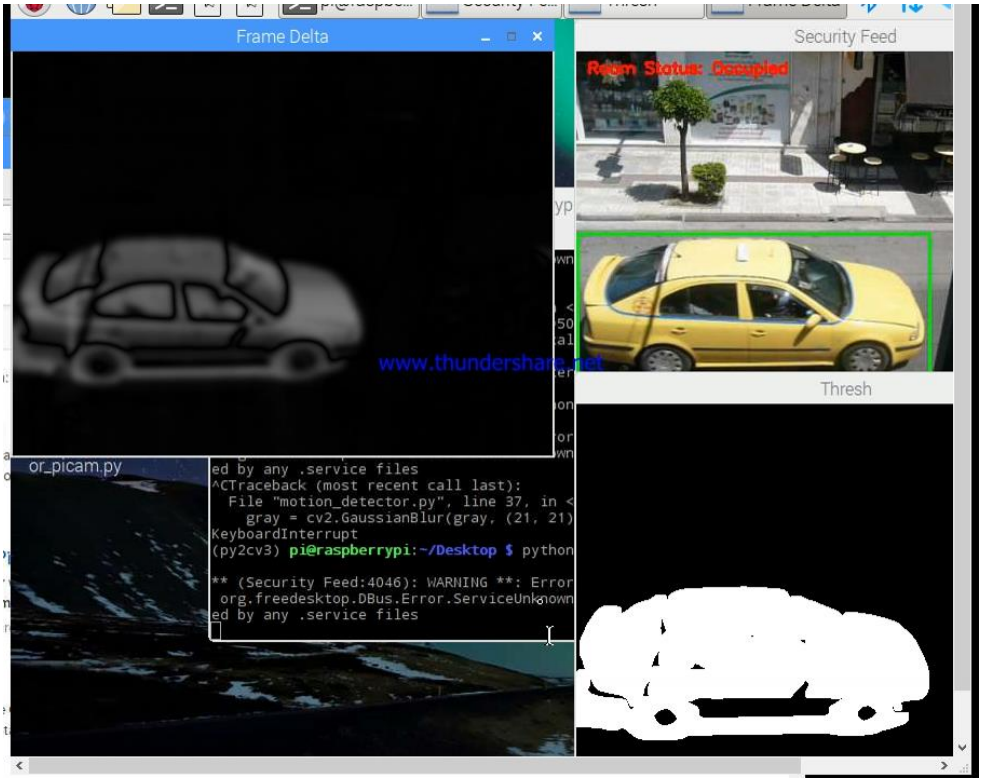
Στο παρακάτω πείραμα θα εκτελέσουμε τα ίδια βήματα με πρίν για την ανίχνευση οχημάτων. Ο κώδικας μας θα γίνει σε Python και compiled από τον τερματικό του Debian όπου κατά την εκτέλεση εν τέλει θα χρησιμοποιεί foreground detection για την ανίχνευση οχημάτων σε έναν δρόμο και θα υποδεικνύει τον αριθμό 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())
```

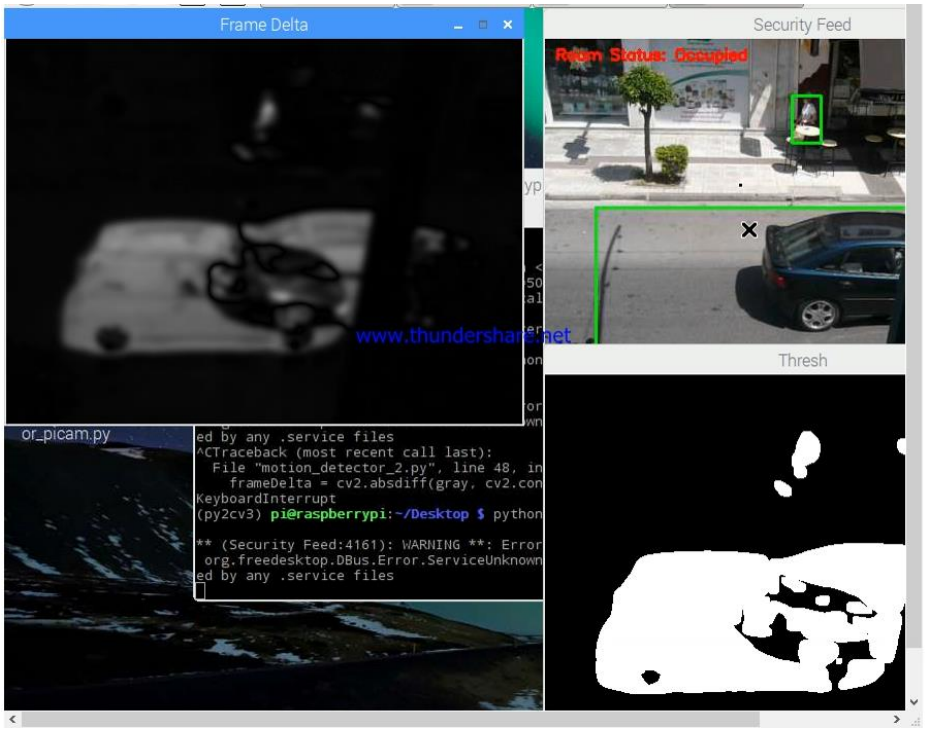
```
# video file
camera = cv2.VideoCapture(args["video"])


# αρχικοποίηση πρώτου frame του video
firstFrame = None
# επανάληψη στα frames του video
while True:
# εύρεση και αρχικοποίηση του πρώτου frame σαν occupied/unocupied
(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
```

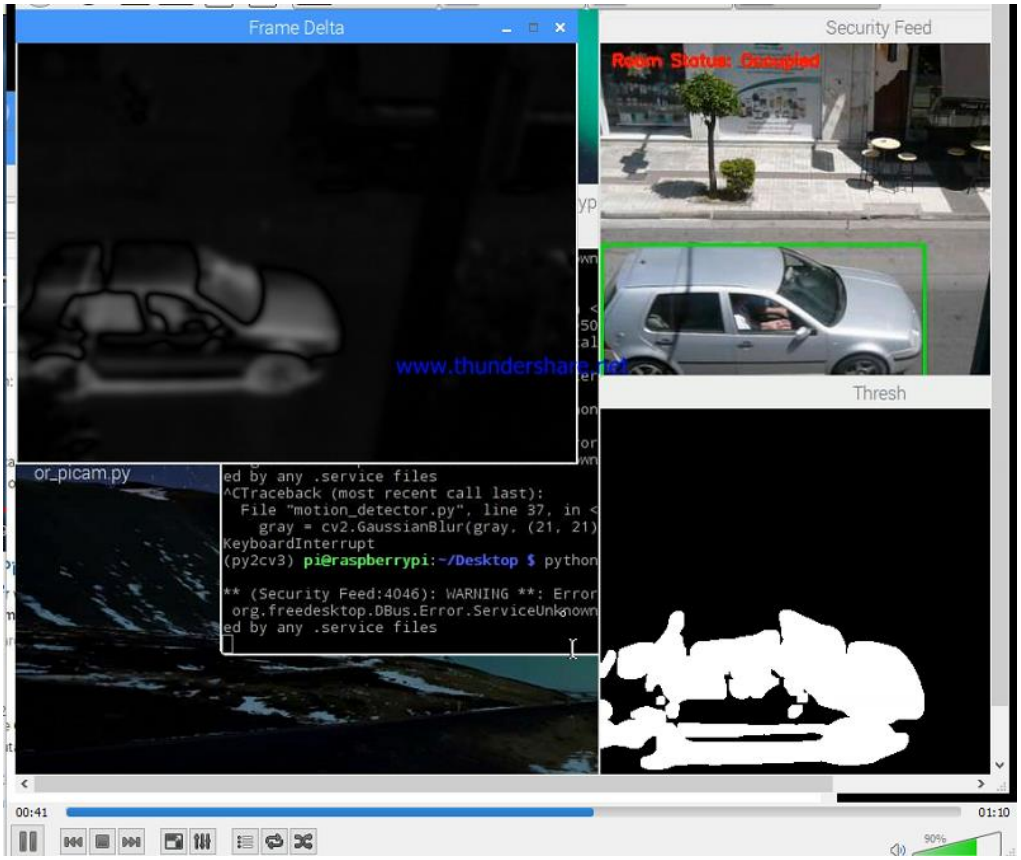
```
# υπολογισμός της απόλυτης διαφοράς μεταξύ του τρέχοντος και του αρχικού frame

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 δοθεί 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()
```



6.1 results of foreground detection on python

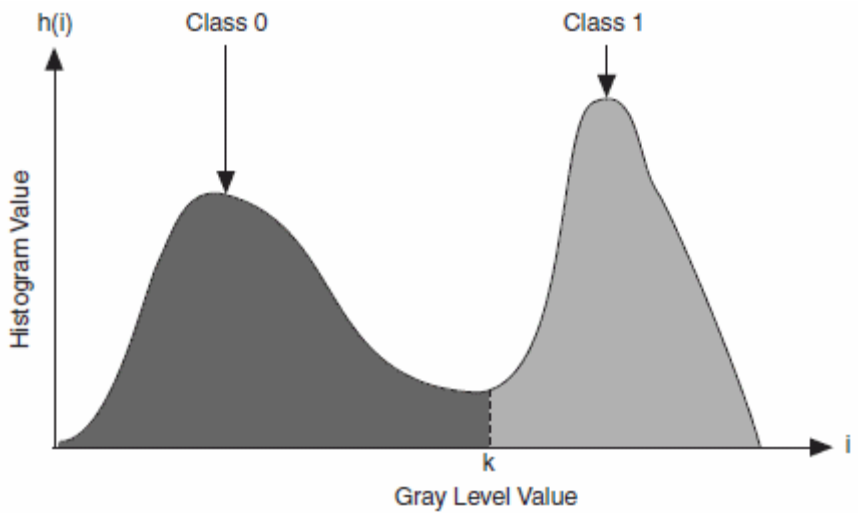


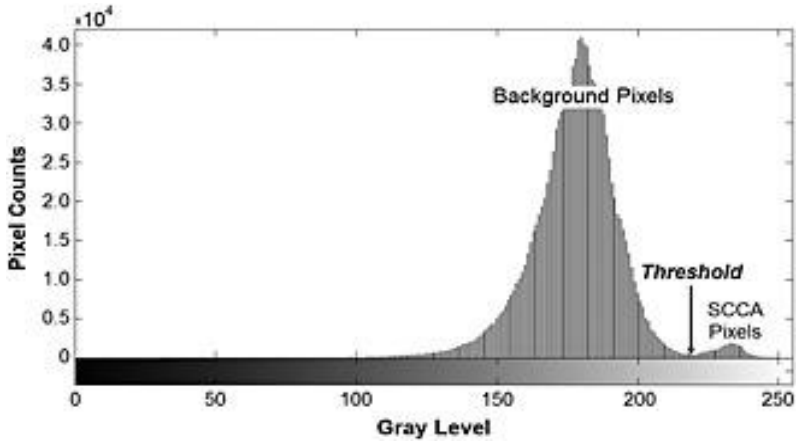


6.2-6.3 results of foreground detection on python

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.





6.4-6.5 Gray Level Value Diagram in Moving Average Background

Καταμέτρηση αυτοκινήτων με ανίχνευση Moving Average Background

Όπως είδαμε πριν καταφέραμε να κάνουμε ανίχνευση των αυτοκινήτων με τον τρόπο Background Detection αλλά 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 όπου 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 κατανείμουμε τα αντικείμενα μας χρησιμοποιώντας την τεχνική αφαίρεσης παρασκηνίου δηλαδή 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.

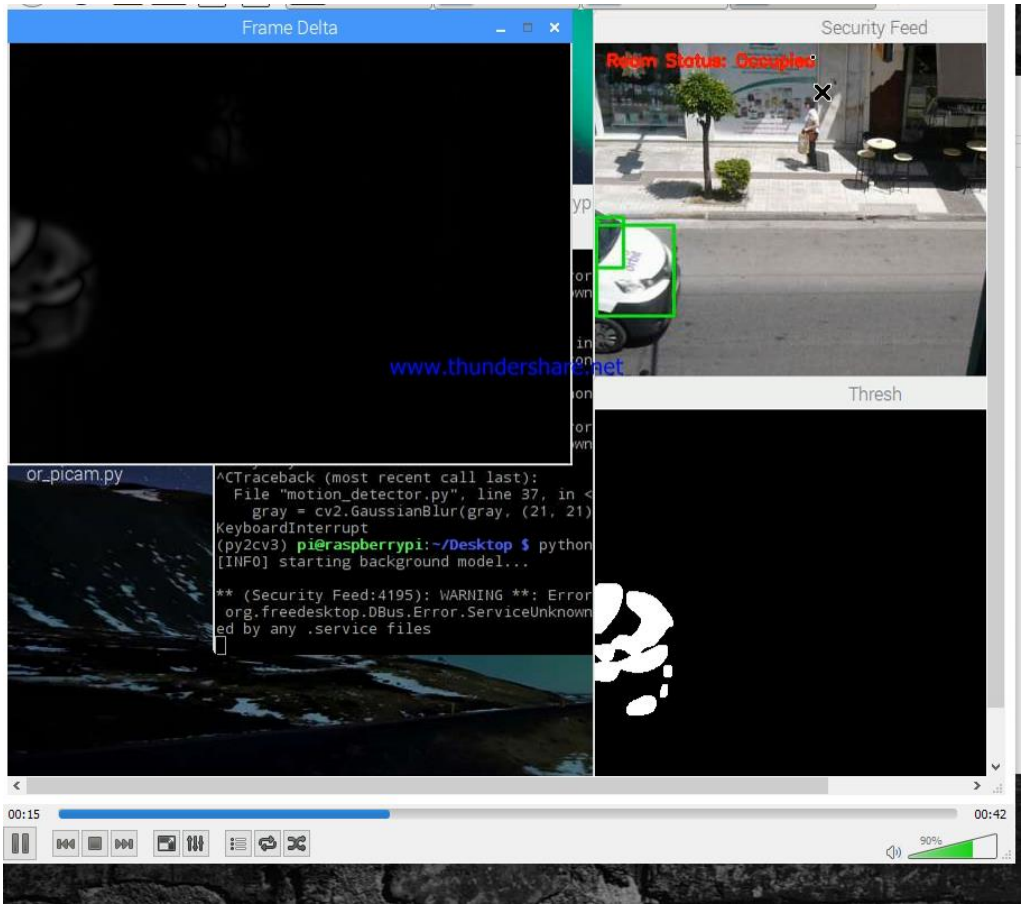
Ο κώδικας σε Python είναι ο ακόλουθος:

```
# 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)
```

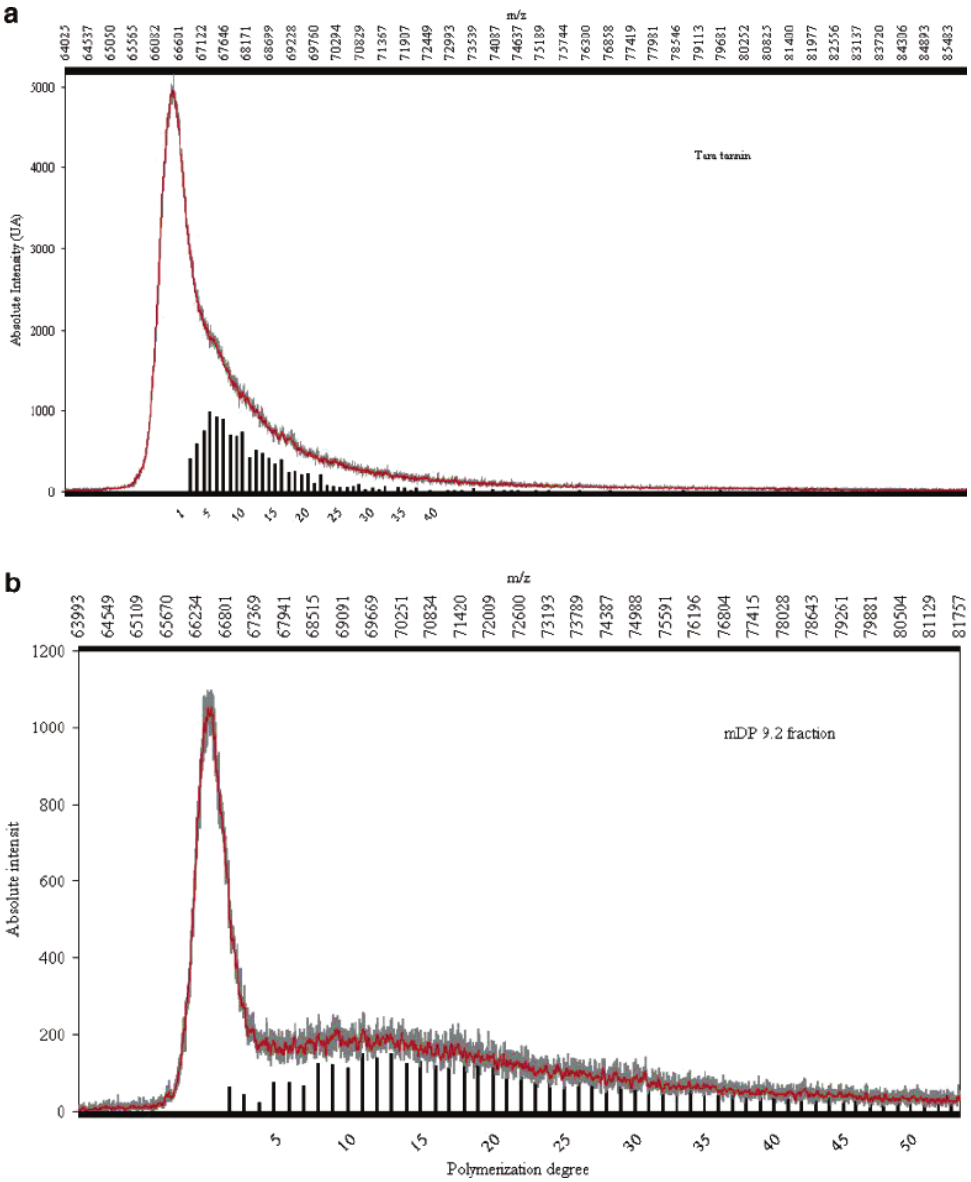
```
# show the frame and record if the user presses a key
cv2.imshow("Security Feed", frame)
cv2.imshow("Thresh", thresh)
cv2.imshow("Frame Delta", frameDelta)
key = cv2.waitKey(1) & 0xFF

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



7.1 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



7.2-7.3 Moving Average background graph

Καταμέτρηση αυτοκινήτων με Gaussian Mixture Model

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

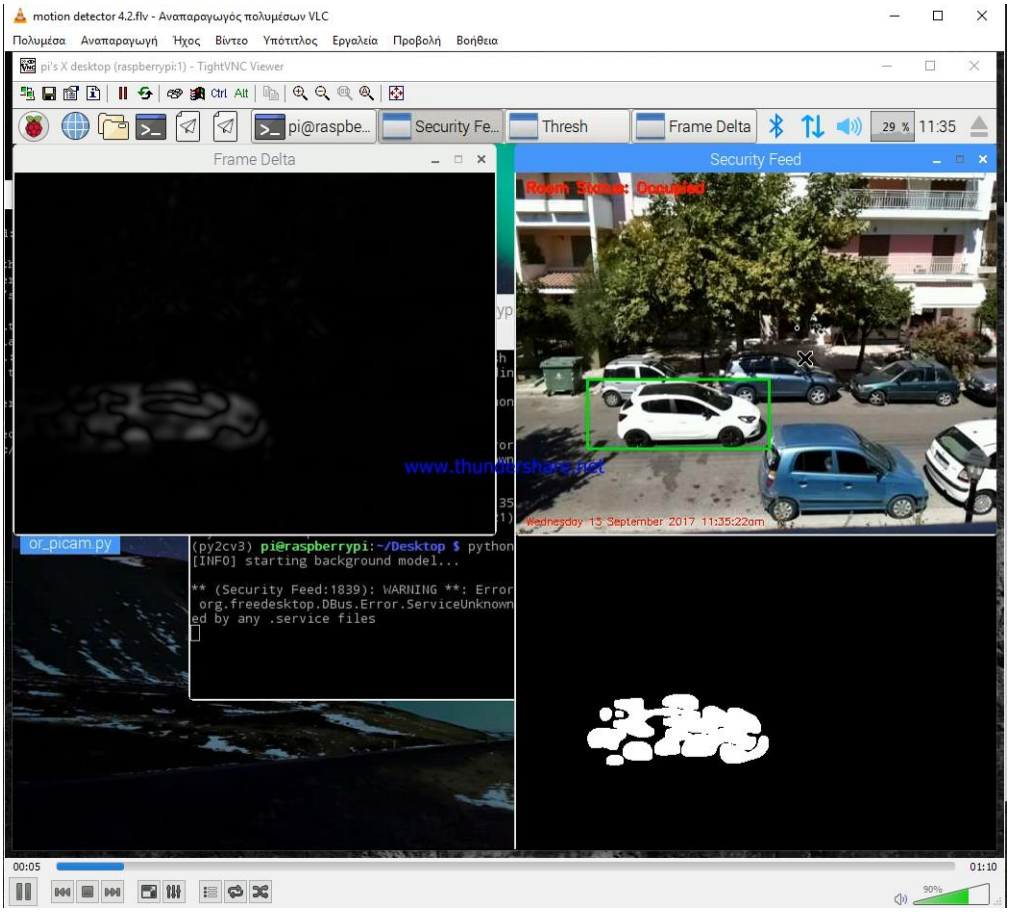
Ο κώδικας σε Python που αντιστοιχεί σε αυτή την μέθοδο είναι η εξής:

```
# 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()
```

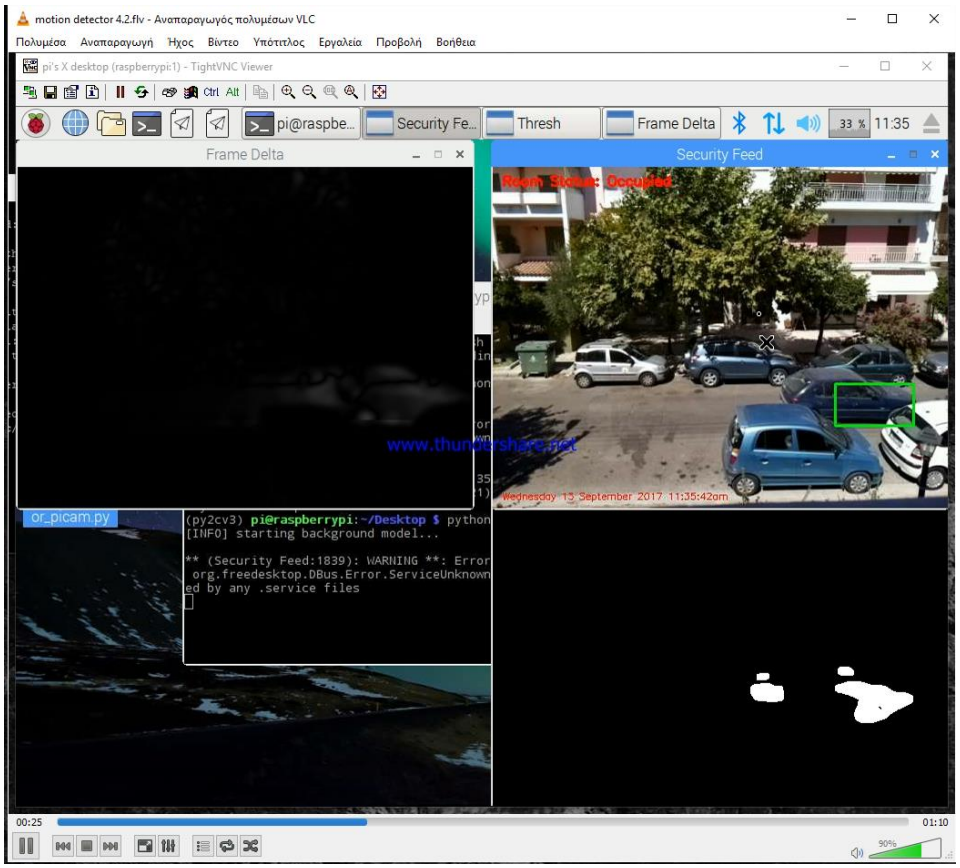
```
# loop over the frames of the video
while True:
# grab the current frame and initialize the occupied/unoccupied text
(grabbed, frame) = camera.read()
text = "Unoccupied"

# 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.

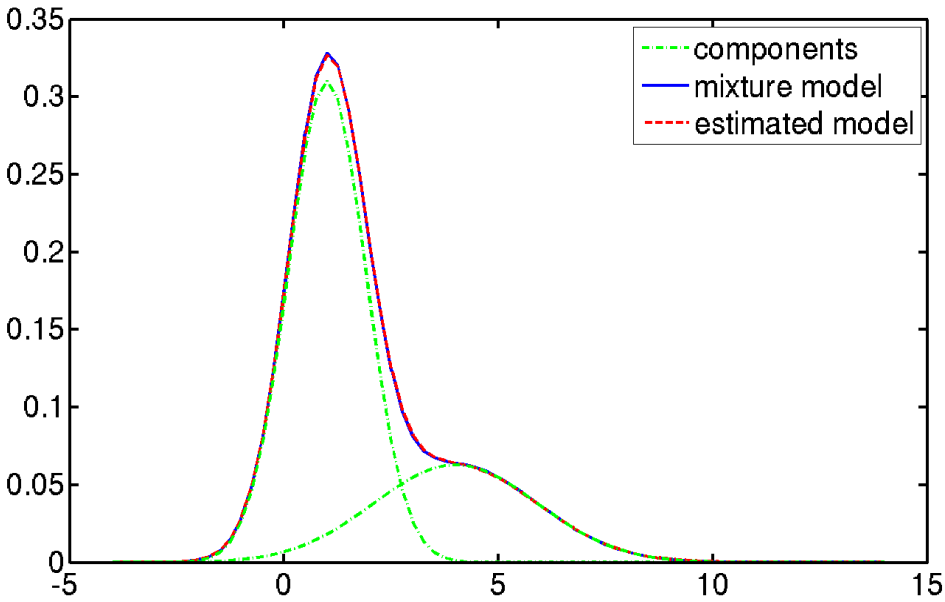


8.1 results of Gaussian mixture model in Python



8.2 results of Gaussian mixture model in Python

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



8.3 Gaussian Mixture Model Pixel Analysis

Βελτίωση του αλγορίθμου με εφαρμογή και χρήση του φίλτρου Kalman

Το 1960, ο R.E.Kalman δημοσίευσε το διάσημο έγγραφό του περιγράφοντας μια επαναλαμβανόμενη λύση στο γραμμικό πρόβλημα φιλτραρίσματος διακριτών δεδομένων. Από εκείνη τη στιγμή, οφείλεται σε αυτόν σε μεγάλο βαθμό η πρόοδος στον ψηφιακό υπολογισμό, το φίλτρο Kalman έχει αποτελέσει αντικείμενο εκτεταμένης έρευνας και εφαρμογής, ιδιαίτερα στον τομέα της αυτόνομης ή υποβοηθούμενης πλοήγησης. Το Φίλτρο Kalman αποτελεί μια από τις σημαντικότερες τεχνολογικές εφευρέσεις του περασμένου αιώνα. Είναι ένας αλγόριθμος που εφαρμόζεται σε συστήματα που δέχονται εξωτερικές φυσικές διαταραχές (θορύβους), με σκοπό τον «καθαρισμό» των μετρήσεων που γίνονται και τη δημιουργία μιας νέας εκτίμησης της κατάστασης του συστήματος, απογυμνωμένη από διαταραχές.

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.

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 χρονοσειρών που χρησιμοποιείται σε πεδία όπως η επεξεργασία σήματος και η οικονομετρία.

Μαθηματικός φορμαλισμός του φίλτρου Kalman

Το φίλτρο Kalman αντιμετωπίζει το γενικό πρόβλημα της προσπάθειας να εκτιμηθεί η κατάσταση nxℜ∈ μιας διακριτά ελεγχόμενης διαδικασίας που διέπεται από την γραμμική στοχαστική διαφορική εξίσωση.

$xk = Axk-1+Buk+wk-1$

με τη διάσταση $y \in Rm$:

$yk = Hxk+vk$

Οι τυχαίες μεταβλητές wk και vk αντιπροσωπεύουν αντίστοιχα τον θόρυβο επεξεργασίας και μέτρησης. Θεωρούνται ότι είναι ανεξάρτητα το ένα από το άλλο και με κανονικές κατανομές πιθανότητας:

$p(w) \approx N(0,Q)$
$p(v) \approx N(0,R)$

Ο πίνακας n × n συνδέει την κατάσταση στο προηγούμενο βήμα του χρόνου με την κατάσταση στο τρέχον βήμα, ελλείψει είτε εισόδου οδηγού είτε θορύβου διεργασίας. Ο πίνακας B n × l αντιστοιχεί στην είσοδο ελέγχου u∈Rl στην κατάσταση x. Ο πίνακας H m × n στην εξίσωση της μέτρησης συνδέει την κατάσταση με τη μέτρηση yk.

Η διαδικασία Kalman Filtering έχει δύο βήματα:

* **Πρόβλεψη**: η κατάσταση επόμενου βήματος του συστήματος προβλέπεται με δεδομένες τις προηγούμενες μετρήσεις

* **Ενημέρωση**: εκτιμάται η τρέχουσα κατάσταση του συστήματος δεδομένης της μέτρησης εκείνης της χρονικής στιγμής

Αυτά τα βήματα εκφράζονται σε εξίσωση-μορφή ως εξής:

Prediction

$$Xk-=Ak-1Xk-1+BkUk$$
$$Pk-=Ak-1Pk-1ATk-1+Qk-1$$

Update

$$Vk=Yk-Hk-X-k$$
$$Sk=HkPk-HTk+Rk$$
$$Kk=Pk-HTkSk-1$$
$$Xk=Xk-+KkVk$$
$$Pk=Pk--KkSkKTk$$

όπου:

* Xk- και Pk- είναι ο προβλεπόμενος μέσος όρος και η συνδιακύμανση της κατάστασης, αντίστοιχα, στο βήμα k πριν από τη μέτρηση.
* Xk και Pk είναι ο εκτιμώμενος μέσος όρος και η συνδιακύμανση της κατάστασης, αντίστοιχα, στο βήμα k μετά τη μέτρηση.
* Yk είναι ο μέσος όρος της μέτρησης στο χρονικό βήμα k.
* Vk είναι η καινοτομία ή η υπολειπόμενη μέτρηση στο χρονικό βήμα k.
* Sk είναι η συνδιακύμανση πρόβλεψης μέτρησης στο χρονικό βήμα k.

* Kk είναι το κέρδος του φίλτρου, το οποίο δείχνει πόσο οι προβλέψεις θα πρέπει να διορθωθούν στο βήμα χρόνου k.

Ανίχνευση κίνησης αντικειμένων με φίλτρο Kalman

Προαπαιτούμενα: OpenCv 3.0 για γλώσσα Python 2.7

Ανίχνευση αντικειμένων:

```
# Import python libraries
import cv2
import copy
from detectors import Detectors
from tracker import Tracker


def main():

    # δημιουργία OpenCv αντικειμένου καταγραφής βίντεο
    cap = cv2.VideoCapture('data/TrackingBugs.mp4')

    detector = Detectors()

    # δημιουργία ανιχνευτή
    tracker = Tracker(160, 30, 5, 100)

    # αρχικοποίηση μεταβλητών
    skip_frame_count = 0
    track_colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0),
                    (0, 255, 255), (255, 0, 255), (255, 127, 255),
```

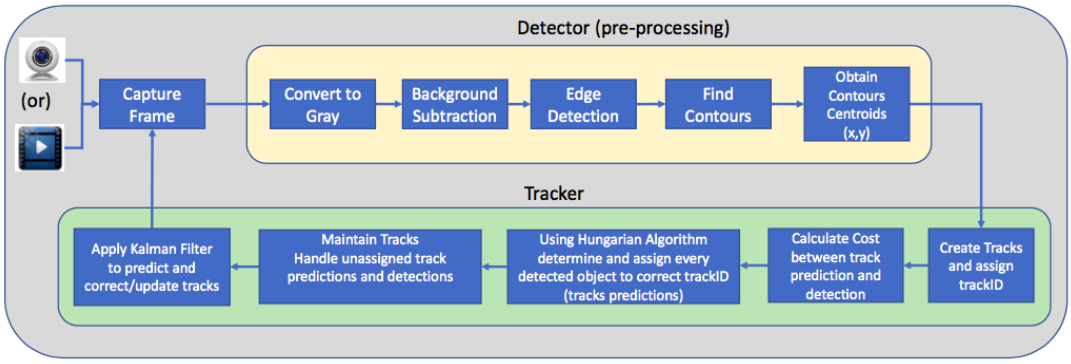
```
            (127, 0, 255), (127, 0, 127)]
pause = False

# άπειρος βρόχος για την επεξεργασία πλαισίων βίντεο
while(True):

    ret, frame = cap.read()

    # αντίγραφο αρχικού frame
    orig_frame = copy.copy(frame)

    # παράβλεψη αρχικών frames
    if (skip_frame_count < 15):
        skip_frame_count += 1
        continue

    # Εντοπισμός και επιστροφή κεντροειδών αντικειμένων στο πλαίσιο
    centers = detector.Detect(frame)

    # αν ανιχνευθούν κεντροειδή
    if (len(centers) > 0):

        # Track object using Kalman Filter
        tracker.Update(centers)


        # χρήση χρωμμάτων για διαφορετικές γραμμές
        for i in range(len(tracker.tracks)):
            if (len(tracker.tracks[i].trace) > 1):
                for j in range(len(tracker.tracks[i].trace)-1):
                    # Draw trace line
                    x1 = tracker.tracks[i].trace[j][0][0]
                    y1 = tracker.tracks[i].trace[j][1][0]
                    x2 = tracker.tracks[i].trace[j+1][0][0]
                    y2 = tracker.tracks[i].trace[j+1][1][0]
                    clr = tracker.tracks[i].track_id % 9
                    cv2.line(frame, (int(x1), int(y1)), (int(x2), int(y2)),
                             track_colors[clr], 2)
```

```
            # τελικό frame ανιχνευσης
            cv2.imshow('Tracking', frame)

        # αρχικό frame
        cv2.imshow('Original', orig_frame)


    # έλεγχος πλήκτρων
        k = cv2.waitKey(50) & 0xff
        if k == 27:  # 'esc' key has been pressed, exit program.
            break
        if k == 112:  # 'p' has been pressed. this will pause/resume the code.
            pause = not pause
            if (pause is True):
                print("Code is paused. Press 'p' to resume..")
                while (pause is True):
                    # stay in this loop until
                    key = cv2.waitKey(30) & 0xff
                    if key == 112:
                        pause = False
                        print("Resume code..!!")
                        break

    # οταν ολοκληρωθεί ελευθέρωσε
    cap.release()
    cv2.destroyAllWindows()


if __name__ == "__main__":
    # execute main
    main()
```

Φίλτρο Kalman:

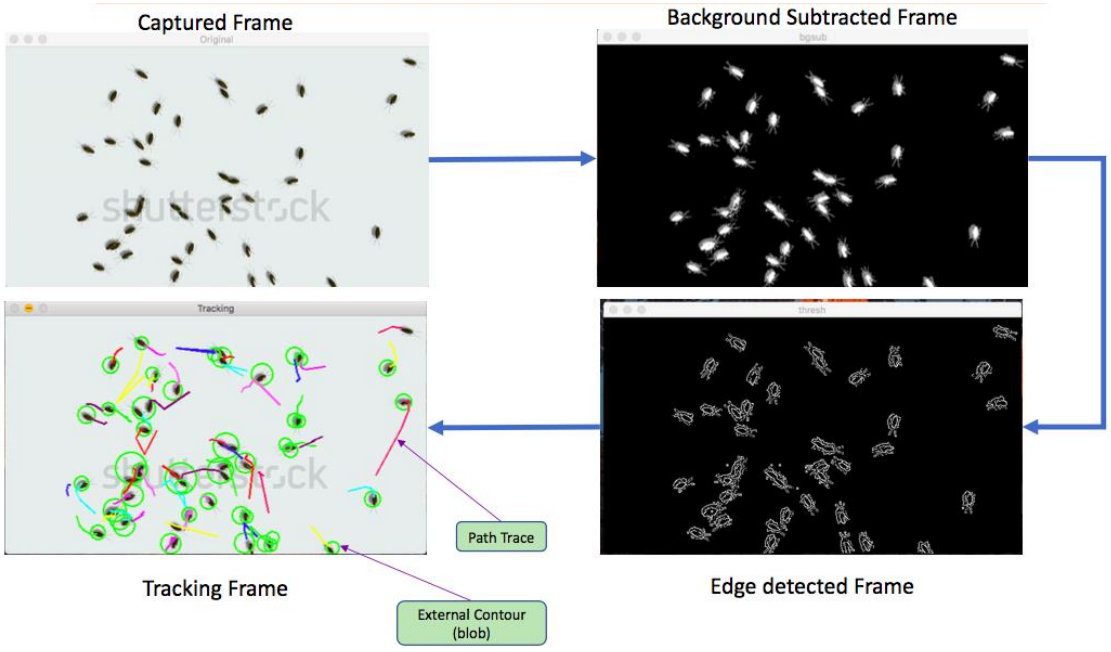
```
# Import python libraries
import numpy as np


class KalmanFilter(object):

    def __init__(self):

        self.dt = 0.005  # delta time

        self.A = np.array([[1, 0], [0, 1]])  # matrix in observation equations
        self.u = np.zeros((2, 1))  # previous state vector

        # (x,y) tracking object center
        self.b = np.array([[0], [255]])  # vector of observations

        self.P = np.diag((3.0, 3.0))  # covariance matrix
        self.F = np.array([[1.0, self.dt], [0.0, 1.0]])  # state transition mat

        self.Q = np.eye(self.u.shape[0])  # process noise matrix
        self.R = np.eye(self.b.shape[0])  # observation noise matrix
        self.lastResult = np.array([[0], [255]])

    def predict(self):

        # εκτίμηση προβλεπόμενης κατάστασης
        self.u = np.round(np.dot(self.F, self.u))
        # προβλεπόμενη συνιστώσα εκτίμησης
        self.P = np.dot(self.F, np.dot(self.P, self.F.T)) + self.Q
        self.lastResult = self.u  # same last predicted result
```

```
    return self.u


def correct(self, b, flag):


    if not flag:
        # ενημέρωσε με χρήση πρόβλεψης
        self.b = self.lastResult
    else:
        # ενημέρωσε με χρήση ανίχνευσης
        self.b = b
    C = np.dot(self.A, np.dot(self.P, self.A.T)) + self.R
    K = np.dot(self.P, np.dot(self.A.T, np.linalg.inv(C)))

    self.u = np.round(self.u + np.dot(K, (self.b - np.dot(self.A,
                                                  self.u))))
    self.P = self.P - np.dot(K, np.dot(C, K.T))
    self.lastResult = self.u
    return self.u
```



11.1 Object Tracking Pipeline

Τα αποτελέσματα της εφαρμογής του φίλτρου Kalman φαίνονται στο σχήμα 11.2 με τις χρωματιστές γραμμές να δείχνουν τις προβλεπόμενες πορείες των αντικειμένων.



11.2 Kalman filter results

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

13. https://github.com/senko/srianant/kalman_filter_multi_object_tracking .py - Kalman filter using Python