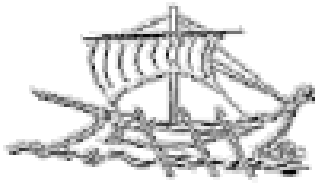


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία εικόνας και αναγνώριση προτύπων με το Raspberry Pi

Πέτρος Παπαδάκος

Εισηγητής: Ιωάννης Ν. Έλληνας, Καθηγητής

ΑΘΗΝΑ
- 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επεξεργασία εικόνας και αναγνώριση προτύπων με το Raspberry Pi 3

Πέτρος Παπαδάκος
Α.Μ. 42719

Εισηγητής:

Ιωάννης Ν. Έλληνας, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης -/-/2017

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Πέτρος Παπαδάκος, του Γεωργίου, με αριθμό μητρώου 42719, 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 ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της επεξεργασίας εικόνας και της αναγνώρισης προτύπων. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, Ιωάννης Ν. Έλληνας, τον οποίο θα ήθελα να ευχαριστήσω. Ακόμα θα ήθελα να ευχαριστήσω τους φίλους μου που με στήριξαν σε αυτό το διάστημα εκπόνησης της πτυχιακής μου εργασίας. Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για τη διαρκή της υποστήριξη όλα αυτά τα χρόνια.

ΠΕΡΙΛΗΨΗ

Η επεξεργασία εικόνας και η αναγνώριση προτύπων είναι δυο επιστημονικοί κλάδοι, οι οποίοι έχουν γνωρίσει ραγδαία ανάπτυξη. Αλγόριθμοι επεξεργασίας εικόνας ξεκίνησαν να χρησιμοποιούνται στο κομμάτι της μετάδοσης εικόνων στη βιομηχανία των εφημερίδων και σήμερα, με την τεχνολογία να βρίσκεται στα ύψη, η ανάγκη ύπαρξης της επεξεργασίας εικόνας είναι δεδομένη. Η αναγνώριση προτύπων χρησιμοποιήθηκε για πρώτη φορά για την επίλυση του προβλήματος ταξινόμησης αντικειμένων σε κατηγορίες. Σήμερα, όλα τα προηγμένα συστήματα ασφαλείας, οι αυτόνομες συσκευές, ακόμα και οι σαρωτές δαχτυλικών αποτυπωμάτων των κινητών τηλεφώνων, χρησιμοποιούν αλγορίθμους αναγνώρισης προτύπων

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη αλγορίθμων επεξεργασίας εικόνας και συστημάτων αναγνώρισης προτύπων με χρήση του Raspberry Pi. Στις πρώτες ενότητες γίνεται η παρουσίαση του Raspberry Pi, της γλώσσας προγραμματισμού Python και των βιβλιοθηκών που θα χρησιμοποιηθούν. Στη συνέχεια, γίνεται παρουσίαση του θεωρητικού υποβάθρου της επεξεργασίας εικόνας, ορισμένων συναρτήσεων και αναπτύσσονται αλγόριθμοι χειρισμού και τεχνικές βελτιστοποίησης εικόνων. Έπειτα, γίνεται παρουσίαση του θεωρητικού υποβάθρου της αναγνώρισης προτύπων, ορισμένων αλγορίθμων και αναπτύσσονται ορισμένα συστήματα αναγνώρισης προτύπων όπως ανιχνευτές προσώπων και συστήματα αναγνώρισης χαρακτήρων. Τέλος παρουσιάζονται τα συμπεράσματα της πτυχιακής εργασίας και οι προοπτικές μελλοντικής εργασίας.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία εικόνας, Αναγνώριση προτύπων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Raspberry Pi, Python, τεχνικές βελτιστοποίησης εικόνων, ανιχνευτές προσώπων, συστήματα αναγνώρισης χαρακτήρων

ABSTRACT

Image processing and pattern recognition are two disciplines that have experienced rapid growth. Image processing algorithms have begun to be used in the transmission of images in the newspaper industry, and today, with the technology being so high, the need for image processing is a given. Pattern recognition was first used to resolve the problem of classifying objects into categories. Today, all advanced security systems, stand-alone devices, and even fingerprint scanners on mobile phones, use pattern recognition algorithms

This dissertation deals with the development of image processing algorithms and pattern recognition systems using Raspberry Pi. In the first sections, we present the Raspberry Pi, the Python programming language and the libraries that will be used. Subsequently, the theoretical background of image processing, and certain functions are presented and manipulation algorithms and image optimization techniques are developed. Then, a presentation of the theoretical background of pattern recognition and certain algorithms is made and some pattern recognition systems such as face detectors and character recognition systems are being developed. Finally, the conclusions of the dissertation and the prospects for future work are presented.

SCIENTIFIC AREA: Image processing, Pattern recognition

KEYWORDS: Raspberry Pi, Python, image optimization techniques, face detectors, character recognition systems

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Σκοπός της πτυχιακής εργασίας

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη, η εφαρμογή και η αξιολόγηση αλγορίθμων που αφορούν την επεξεργασία εικόνας και την αναγνώριση προτύπων.

Συγκεκριμένα, στο κομμάτι της επεξεργασίας εικόνας θα παρουσιάσουμε συναρτήσεις: βασικού χειρισμού εικόνων (ανάγνωση, εμφάνιση, αποθήκευση εικόνας, εμφάνιση χαρακτηριστικών εικόνας, υπολογισμό ιστογραμμάτων), εφαρμογής τεχνικών βελτιστοποίησης εικόνας (επέκταση αντίθεσης, κατωφλίωση, εξισορρόπηση ιστογράμματος) και εφαρμογής φίλτρων με σκοπό τον εμπλουτισμό εικόνας, ανάδειξη ακμών και απαλοιφής θορύβου και θα αναπτύξουμε κώδικα για την υλοποίηση των προαναφερθέντων. Οι αλγόριθμοι αυτοί θα αναπτυχθούν με χρήση της γλώσσας Python και των βιβλιοθηκών OpenCV, NumPy και Matplotlib.

Στο κομμάτι της αναγνώρισης προτύπων θα αναπτύξουμε συστήματα εντοπισμού και αναγνώρισης αντικειμένων και χαρακτήρων χρησιμοποιώντας γνωστούς αλγορίθμους αναγνώρισης προτύπων. Συγκεκριμένα θα ασχοληθούμε με τους αλγορίθμους Κ-Κοντινότερων γειτόνων (K-Nearest Neighbors – KNN), Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), LeNet-5 και τη μέθοδο αναγνώρισης αντικειμένων κατά Viola & Jones. Ορισμένα συστήματα θα μπορούν να εφαρμοστούν σε πραγματικό χρόνο με χρήση κάμερας ενώ τα υπόλοιπα θα λειτουργούν με δεδομένα από το δίσκο. Τα συστήματα αυτά θα αναπτυχθούν με χρήση της γλώσσας Python και του λογισμικού που φαίνεται στην ενότητα 1.2.

1.2 Υλικό και λογισμικό που χρησιμοποιήθηκε

Παρακάτω παραθέτουμε το υλικό (hardware) και το λογισμικό (software) που χρησιμοποιήθηκε για την εκπόνηση της παρούσας πτυχιακής εργασίας:

- Υλικό (Hardware)
  - ένα RPi 3 Model B

- μια RPi Camera V2
- μια usb webcam Creative Live! Cam Sync HD

➢ Λογισμικό (Software)

Χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

- OpenCV
- NumPy
- SciPy
- Matplotlib
- scikit-learn
- scikit-image
- h5py
- Dlib
- Keras
- imutils
- argparse

1.3 Διάρθρωση της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία διαρθρώνεται ως εξής:

Στο **Κεφάλαιο 2** γίνεται η παρουσίαση του Raspberry Pi. Θα μάθουμε τι είναι το Raspberry Pi, τη πορεία του από τις αρχικές ιδέες μέχρι και σήμερα και τους τρόπους χρήσης του. Επίσης γίνεται μια σύντομη παρουσίαση της γλώσσας προγραμματισμού Python

Στο **Κεφάλαιο 3** γίνεται αναλυτική παρουσίαση της βιβλιοθήκης OpenCV. Θα μάθουμε τι είναι η OpenCV, ορισμένα ιστορικά γεγονότα, τα πλεονεκτήματα και τα πεδία εφαρμογής της. Επίσης, γίνεται μια συνοπτική παρουσίαση της βιβλιοθήκης Dlib και ορισμένων βιβλιοθηκών της Python που θα χρησιμοποιηθούν στη παρούσα εργασία

Στο **Κεφάλαιο 4** γίνεται παρουσίαση του θεωρητικού υποβάθρου του αντικειμένου της επεξεργασίας εικόνας και των αλγορίθμων επεξεργασίας εικόνας με χρήση των βιβλιοθηκών OpenCV, NumPy και Matplotlib.

Στο **Κεφάλαιο 5** γίνεται παρουσίαση του θεωρητικού υποβάθρου του αντικειμένου της αναγνώρισης προτύπων και των συστημάτων αναγνώρισης προτύπων που αναπτύχθηκαν με χρήση διαφόρων βιβλιοθηκών επεξεργασίας εικόνας, αναγνώρισης προτύπων και μηχανικής μάθησης.

Στο **Κεφάλαιο 6** συγκεντρώνονται τα συμπεράσματα της πτυχιακής εργασίας.
Στο **Παράρτημα Α** παρουσιάζονται αρχικά τα χαρακτηριστικά του RPi 3 Model B και παρατίθενται βήμα-βήμα η εγκατάσταση του λειτουργικού Raspbian Jessie και η σύνδεση κάμερας στο RPi (Raspberry Pi Camera V2 και κλασική webcam) καθώς και την εγκατάσταση των απαραίτητων πακέτων (picamera και fswebcam).
Στο **Παράρτημα Β** παρουσιάζεται βήμα-βήμα η εγκατάσταση των βιβλιοθηκών που χρησιμοποιήθηκαν στη παρούσα εργασία.

ΚΕΦΑΛΑΙΟ 2

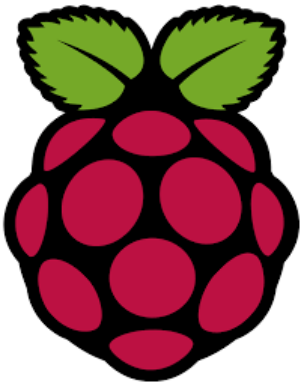
ΜΑΘΑΙΝΟΝΤΑΣ ΤΟ RASPBERRY PI ΚΑΙ ΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PYTHON

2.1 Raspberry Pi

2.1.1 Τι είναι το Raspberry Pi

Το Raspberry Pi είναι μια σειρά υπολογιστών “τσέπης” η οποία αναπτύχθηκε στο Ηνωμένο Βασίλειο από τη Raspberry Pi Foundation με σκοπό τη προώθηση της διδασκαλίας της βασικής επιστήμης των υπολογιστών στα σχολεία. Το πρώτο μοντέλο, το RPi 1 Model B, κυκλοφόρησε το 2012 και μάλιστα είχε πολύ μεγαλύτερη απήχηση από αυτή που όλοι θα περίμεναν. Εκ τότε κυκλοφόρησαν και άλλα μοντέλα (7 μαζί με το αρχικό) φτάνοντας έτσι στο τελευταίο, και πιο δυνατό, RPi 3 Model B. **[18]**

Σύμφωνα με το Raspberry Pi Foundation, πριν από το Φεβρουάριο του 2015 είχαν πωληθεί πάνω από 5 εκατομμύρια RPis, καθιστώντας το, το καλύτερο σε πωλήσεις βρετανικό υπολογιστή ενώ μέχρι τις 9 Σεπτεμβρίου του 2016 είχαν πωληθεί 10 εκατομμύρια. **[18]**



Σχήμα 2.1: Το λογότυπο του Raspberry Pi **[18]**

2.1.2 Λίγα λόγια για την ιστορία του Raspberry Pi

Οι πρώτες ιδέες για το RPi υπήρξαν το 2006 και βασίστηκαν στο μικροελεγκτή Atmel ATmega644. Ο διαχειριστής του Raspberry Pi Foundation Eben Upton συγκέντρωσε μια ομάδα από καθηγητές, πανεπιστημιακούς και λάτρεις των

υπολογιστών προκειμένου να επινοήσουν έναν υπολογιστή ο οποίος θα εμπνεύσει τα παιδιά. Ο υπολογιστής αυτός ήταν εμπνευσμένος από το BBC Micro ο οποίος αναπτύχθηκε το 1981 από την Acorn Computers. Η πρώτη πρωτότυπη έκδοση του υπολογιστή είχε το μέγεθος ενός USB memory stick και στο ένα άκρο του είχε μια θύρα USB ενώ στο άλλο του μια θύρα HDMI. **[18]**

Το πρώτο RPi, το Raspberry Pi Model B κυκλοφόρησε στις 29 Φεβρουαρίου του 2012 σημαίνοντας έτσι την αρχή των υπολογιστών Raspberry Pi. Σχεδόν ένα χρόνο αργότερα και συγκεκριμένα στις 4 Φεβρουαρίου του 2013 κυκλοφόρησε μια φθηνότερη έκδοση του αρχικού μοντέλου, το Raspberry Pi Model A (κόστιζε 25$ σε αντίθεση με τα 35$ που κόστιζε το Model B). Στις 14 Ιουλίου του 2014 κυκλοφόρησε μια βελτιωμένη έκδοση του αρχικού RPi, το Raspberry Pi Model B+ ενώ λίγο αργότερα, στις 10 Νοεμβρίου του 2014 κυκλοφόρησε η αντίστοιχη βελτιωμένη έκδοση το Model A, το Raspberry Pi Model A+. Η 2η Φεβρουαρίου του 2015 σήμανε την έναρξη της δεύτερης γενιάς υπολογιστών RPi με τη κυκλοφορία του Raspberry Pi 2 Model B. Στις 26 Νοεμβρίου του 2015 κυκλοφόρησε το μικρότερο και φθηνότερο RPi, το Raspberry Pi Zero με τιμή μόλις 5$. Τέλος στις 29 Φεβρουαρίου του 2016 κυκλοφόρησε το Raspberry Pi 3 Model B το οποίο αποτελεί τη τρίτη γενιά υπολογιστών RPi. **[19]**

2.1.3 Τρόποι χρήσης του Raspberry Pi

Οι τρόποι με τους οποίους μπορεί να χρησιμοποιηθεί ένα RPi είναι πραγματικά αμέτρητοι. Ο βασικός σκοπός ήταν η εκπαίδευση στα σχολεία, τόσο στο χειρισμό ενός υπολογιστή όσο και στην εκμάθηση προγραμματισμού. Μπορεί να χρησιμοποιηθεί για απλές λειτουργίες που θα έκανε κάποιος σε έναν οποιοδήποτε υπολογιστή (internet, χρήση εφαρμογών office, αναπαραγωγή μουσικής και βίντεο κλπ). Μπορεί να χρησιμοποιηθεί σε εφαρμογές για το σπίτι, όπως: μετατροπή απλής τηλεόρασης σε smart tv, μετατροπή απλού εκτυπωτή σε ασύρματο, σύστημα παρακολούθησης χώρου, συσκευή αύξησης της εμβέλειας του wifi, δημιουργία media streaming box, δημιουργία retro παιχνιδομηχανής κλπ. Χρησιμοποιείται επίσης σε ποικίλες ηλεκτρονικές εφαρμογές αφού μπορούμε να συνδέσουμε πάνω του διάφορους αισθητήρες (θερμοκρασίας, υγρασίας, κίνησης κλπ), άλλες πλακέτες (breadboards, arduino), lcd οθόνες, καθώς και να χρησιμοποιηθεί για οδήγηση ρομποτικών βραχιόνων, κινητήρων κλπ. **[18] [20] [21]**

Σχήμα 2.2: Εφαρμογές με το RPi **1)** Χρήση του RPi ως Desktop PC **2)** Μετατροπή απλής τηλεόρασης σε Smart TV **3)** Χρήση του RPi σε ηλεκτρονικές εφαρμογές **4)** Μετατροπή απλού εκτυπωτή σε ασύρματο **5)** Χρήση του RPi για αύξηση της εμβέλειας του WIFI **6)** Χρήση του RPi ως retro παιχνιδοκονσόλα.

2.2 Python

2.2.1 Τι είναι η Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, γενικής χρήσης και ανοιχτού πηγαίου κώδικα (open source), η οποία δημιουργήθηκε από τον Guido van Rossum, και κυκλοφόρησε για πρώτη φορά το 1991. Σαν γλώσσα, δίνει μεγάλη έμφαση στην εύκολη ανάγνωση του κώδικα και στη σύνταξη, όπου επιτρέπει στους προγραμματιστές να εκφράζουν έννοιες σε λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες προγραμματισμού, όπως στη C ++ και στη Java. Χρησιμοποιείται, διεθνώς, ως εκπαιδευτική γλώσσα αλλά και στην ανάπτυξη σημαντικών εφαρμογών με μεγάλη επιτυχία ενώ διαθέτει εκδόσεις για μια ευρεία γκάμα λειτουργικών συστημάτων, συμπεριλαμβανομένων των Windows, Linux, Mac OSX κλπ. **[7][30]**

Σχήμα 2.3: Το λογότυπο της Python **[30]**

2.2.2 Λίγα λόγια για την ιστορία της Python

Η γλώσσα Python σχεδιάστηκε στα τέλη της δεκαετίας του 1980 και η εφαρμογή της άρχισε το 1989 από τον Guido van Rossum με σκοπό τον χειρισμό εξαιρέσεων και τη διασύνδεση διεπαφής – χρήστη με το λειτουργικό σύστημα Amoeba. Η γλώσσα αυτή αποτέλεσε τον διάδοχο της γλώσσας ABC και το όνομα της προέρχεται από την ομάδα κωμικών Monty Python. **[7][30]**

Η Python 2.0 κυκλοφόρησε στις 16 Οκτωβρίου του 2000, και είχε πολλά νέα χαρακτηριστικά όπως η υποστήριξη του Unicode. Με τη κυκλοφορία της έκδοσης αυτής, η διαδικασία ανάπτυξης προγραμμάτων άλλαξε και υποστηρίχτηκε περισσότερο από τη κοινωνία. **[30]**

Η Python 3.0 κυκλοφόρησε στις 3 Δεκεμβρίου του 2008, μετά από μια μακρά περίοδο δοκιμών. Πολλά από τα κύρια χαρακτηριστικά γνωρίσματα της υποστηρίζονται από τις εκδόσεις 2.6.x και 2.7.x. **[30]**

Το τέλος ζωής (End Of Life – EOL) για τη Python 2.7 αρχικά καθορίστηκε για το 2015 και στη συνέχεια αναβλήθηκε για το 2020, λόγω του γεγονότος ότι ένα μέγαλο μέρος του υπάρχοντος κώδικα δε μπορεί να μεταφερθεί εύκολα στη Python 3.0. **[30]**

ΚΕΦΑΛΑΙΟ 3

ΒΙΒΛΙΟΘΗΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΤΥΠΩΝ

3.1 OpenCV

3.1.1 Τι είναι η OpenCV

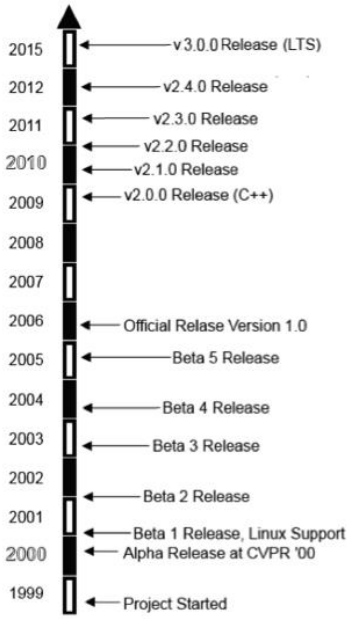
Η OpenCV (Open source Computer Vision) είναι μια βιβλιοθήκη ανοιχτού κώδικα η οποία περιέχει συναρτήσεις που αφορούν την μηχανική όραση (computer vision) καθώς και αλγόριθμους μηχανικής μάθησης (machine learning). Είναι δωρεάν διαθέσιμη στο διαδίκτυο και λόγο της άδειας χρήσης BSD (Berkeley Software Distribution) μπορεί να χρησιμοποιηθεί τόσο για ερευνητική όσο και για εμπορική χρήση. Είναι γραμμένη σε γλώσσα C και C++ και υποστηρίζει ένα μεγάλο εύρος γλωσσών προγραμματισμού, όπως Python, Java, MATLAB κλπ. Τρέχει σε λειτουργικό σύστημα Windows, Linux, MacOS, Android και IOS. **[13] [22] [24]**

Σχήμα 3.1: Το λογότυπο της OpenCV **[22]**

3.1.2 Ιστορική αναδρομή

Η OpenCV αναπτύχθηκε στο κέντρο ερευνών της Intel στο Νίζνι Νοβγκόροντ της Ρωσίας το 1999. Η πρώτη alpha έκδοση της OpenCV έγινε διαθέσιμη στο κοινό στην IEEE Conference on Computer Vision and Pattern Recognition το 2000 ενώ από το 2001 μέχρι και το 2005 κυκλοφόρησαν 5 beta εκδόσεις. Η

πρώτη επίσημη έκδοση της βιβλιοθήκης, η έκδοση 1.0., κυκλοφόρησε το 2006. Τον Οκτώβριο του 2008 κυκλοφόρησε η έκδοση 1.1 της βιβλιοθήκης και στα μέσα του ίδιου χρόνου έλαβε υποστήριξη από το Willow Garage προκειμένου να είναι ξανά υπό ανάπτυξη. Τον Οκτώβριο του 2009 κυκλοφόρησε η έκδοση 2.0.0 της OpenCV η οποία περιελάμβανε σημαντικές αλλαγές στο περιβάλλον της C++ με σκοπό την ευκολότερη και ασφαλέστερη πληκτρολόγηση μοτίβων (patterns), τη προσθήκη νέων συναρτήσεων και τη καλύτερη εφαρμογή της, όσον αφορά την απόδοση, σε πολυπύρηνα συστήματα (multi-core systems) ενώ η ανάπτυξη της γινόταν από ανεξάρτητη ρωσική ομάδα που υποστηριζόταν από τις εμπορικές επιχειρήσεις. Το 2010 κυκλοφόρησαν οι εκδόσεις 2.1.0 και 2.2.0. Το 2011 κυκλοφόρησε η έκδοση 2.3.0 και ένα χρόνο αργότερα η έκδοση 2.4.0, ενώ η υποστήριξη της εξαγοράστηκε από ένα μη κερδοσκοπικό ίδρυμα, το OpenCV.org. Τέλος, το 2015 κυκλοφόρησε η έκδοση 3.0.0 της βιβλιοθήκης. **[22]**



Σχήμα 3.2: Χρονοδιάγραμμα με τη κυκλοφορία των εκδόσεων της OpenCV **[26]**

3.1.3 Πλεονεκτήματα της OpenCV

Το βασικότερο πλεονέκτημα της OpenCV είναι η ταχύτητα. Αυτός είναι και ο βασικός λόγος που δημιουργήθηκε διότι η Intel ήθελε να δείξει πως διέθετε ικανούς επεξεργαστές που μπορούσαν να επεξεργάζονται εικόνες και βίντεο σε πραγματικό χρόνο. Το δεύτερο είναι ότι ενώ αποτελεί ελεύθερο λογισμικό χρησιμοποιείται από μεγάλες εταιρίες όπως τη Google, την Intel, την IBM και

πανεπιστήμια όπως το MIT. Τέλος, δεν απαιτεί περίπλοκα μηχανήματα και εξοπλισμούς. **[13]**

3.1.4 Πεδία εφαρμογής της OpenCV

Η OpenCV διαθέτει περισσότερους από 2.500 βελτιστοποιημένους αλγορίθμους υπολογιστικής όρασης και εκμάθησης μηχανής. Για το λόγο αυτό η βιβλιοθήκη μπορεί να χρησιμοποιηθεί σε εφαρμογές: ανίχνευσης και αναγνώρισης προσώπων, εντοπισμού αντικειμένων, εξαγωγής τρισδιάστατων μοντέλων (3D) από αντικείμενα, αφαίρεσης του κόκκινου από τα μάτια σε φωτογραφίες που έχουν τραβηχτεί με φλας, αναγνώρισης τοπίων και τοποθέτησης ψηφιακών αντικειμένων σε συστήματα Επαυξημένης Πραγματικότητας (Augmented Reality – AR) και της. **[24]**

Έκτος από καθιερωμένες εταιρείες της η Google, η Yahoo, η Intel, η IBM, η Honda και η Toyota οι οποίες χρησιμοποιούν την OpenCV, υπάρχουν και της startup εταιρείες, της η Applied Minds, η VideoSurf και η Zeitera, οι οποίες κάνουν εκτεταμένη χρήση της βιβλιοθήκης. Χρησιμοποιείται παγκοσμίως σε της εφαρμογές και συστήματα, της στην ανίχνευση εισβολών σε βίντεο επιτήρησης στο Ισραήλ, στη παρακολούθηση του εξοπλισμού στα ορυχεία της Κίνας, στην οδήγηση ρομπότ και στη μεταφορά αντικειμένων με χρήση αυτών στο Willow Garage, στην ανίχνευση πνιγμών σε πισίνες της Ευρώπης, στην επιθεώρηση ετικετών των προϊόντων στα εργοστάσια σε όλο τον κόσμο και στη ταχεία ανίχνευση προσώπων στην Ιαπωνία. **[24]**

3.2 Dlib

3.2.1 Τι είναι η Dlib

Η Dlib είναι μια βιβλιοθήκη ανοιχτού κώδικα η οποία περιέχει αλγόριθμους μηχανικής μάθησης και εργαλεία για τη δημιουργία σύνθετων λογισμικών Είναι δωρεάν διαθέσιμη στο διαδίκτυο και ελεύθερη για οποιαδήποτε χρήση λόγο της άδειας χρήσης BSL (Boost Software License). Είναι γραμμένη σε γλώσσα C++ και εκτός της C++, υποστηρίζει και τη γλώσσα Python. Τρέχει σε λειτουργικό σύστημα Windows, Mac OSX , Linux, Solaris και HP-UX. **[32]**

Ο Davis King υπήρξε ο κύριος συγγραφέας της dlib από τότε που ξεκίνησε η ανάπτυξη της το 2002. Εκείνη την εποχή η dlib έχει αναπτυχθεί ώστε να

περιλαμβάνει μια μεγάλη ποικιλία εργαλείων. Σήμερα, περιέχει συστατικά λογισμικού για δίκτυα, χειρισμό διεργασιών και νημάτων, γραφικές διεπαφές, σύνθετες δομές δεδομένων , γραμμική άλγεβρα, μηχανική μάθηση, επεξεργασία εικόνας, εξόρυξη δεδομένων, ανάλυση αρχείων *.txt και *.xml, βελτιστοποίηση αριθμητικών πράξεων, δίκτυα Bayes και άλλα. Τα τελευταία χρόνια, μεγάλο μέρος της ανάπτυξης έχει επικεντρωθεί στη δημιουργία μιας ευρείας σειράς εργαλείων μηχανικής μάθησης. **[32]**

Σχήμα 3.3: Το λογότυπο της Dlib **[31]**

3.3 Python Libraries

Στην ενότητα αυτή, παρουσιάζονται συνοπτικά ορισμένες βιβλιοθήκες της Python που χρησιμοποιήθηκαν στη παρούσα εργασία.

3.3.1 NumPy

Η NumPy είναι μια βιβλιοθήκη για τη γλώσσα προγραμματισμού Python που μεταξύ άλλων παρέχει υποστήριξη για μεγάλους πολυδιάστατους πίνακες τιμών. Χρησιμοποιώντας το NumPy, μπορούμε να εκφράσουμε τις εικόνες ως πολυδιάστατους πίνακες τιμών. Αυτό είναι σημαντικό, όχι μόνο από πλευράς απόδοσης λόγω της χρήσης μικρού αριθμού υπολογιστικών πόρων αλλά και γιατί πολλές βιβλιοθήκες επεξεργασίας εικόνας και μηχανικής μάθησης, όπως η OpenCV, χρησιμοποίει το NumPy για την αναπαράσταση πινάκων. Επιπλέον, χρησιμοποιώντας τις ενσωματωμένες συναρτήσεις μαθηματικών πράξεων υψηλού επιπέδου του NumPy, είναι δυνατή η πραγματοποίηση αριθμητικής ανάλυσης σε μια εικόνα. **[11]**

3.3.2 SciPy

Η SciPy είναι μια βιβλιοθήκη για τη γλώσσα προγραμματισμού Python, η οποία προσθέτει περαιτέρω υποστήριξη στο κομμάτι της επιστήμης των υπολογιστών (scientific and technical computing). Περιέχει συναρτήσεις για βελτιστοποίηση, γραμμική άλγεβρα, ολοκλήρωση, παρεμβολή, ειδικές μαθηματικές λειτουργίες (special functions), μετασχηματισμό Fourier (Fast Fourier Transform – FFT), επεξεργασία σήματος και εικόνας και άλλες. **[33]**

3.3.3 Matplotlib

Η Matplotlib είναι μια βιβλιοθήκη απεικόνισης (plotting library) για τη γλώσσα προγραμματισμού Python, η οποία διαθέτει συναρτήσεις εμφάνισης εικόνων και σχεδίασης γραφημάτων. Το περιβάλλον της είναι αρκετά κοντά στο περιβάλλον του MATLAB. **[34]**

3.3.4 scikit-learn

Η scikit-learn είναι μια βιβλιοθήκη μηχανικής μάθησης για τη γλώσσα προγραμματισμού Python. Διαθέτει αλγορίθμους ταξινόμησης, παλινδρόμησης και ομαδοποίησης όπως υποστήριξη διανύσματος μηχανής (Support Vector Machine – SVM), τυχαία δάση (random forests), ενίσχυση κλίσης (gradient boosting), κ-μέσα (k-means) και άλλους, ενώ έχει σχεδιαστεί για να συνεργάζεται με τις βιβλιοθήκες NumPy και SciPy. **[35]**

3.3.5 scikit-image

Η scikit-image είναι μια βιβλιοθήκη επεξεργασίας εικόνας για τη γλώσσα προγραμματισμού Python. Διαθέτει αλγορίθμους για κατακερματισμό εικόνας, εφαρμογή γεωμετρικών μετασχηματισμών, χειρισμό χρωματικών συστημάτων, φιλτράρισμα, ανίχνευση χαρακτηριστικών και άλλους ενώ έχει σχεδιαστεί για να συνεργάζεται με τις βιβλιοθήκες NumPy και SciPy. **[36]**

3.3.6 h5py

Η h5py είναι μια βιβλιοθήκη για τη γλώσσα προγραμματισμού Python και χρησιμοποιείται για την αποθήκευση μεγάλων αριθμητικών συνόλων δεδομένων, ενώ παρέχει υποστήριξη και για πίνακες τιμών του NumPy. **[11]**

3.3.7 Keras

Η Keras είναι μια βιβλιοθήκη νευρωνικών δικτύων, ανοικτού κώδικα γραμμένη σε γλώσσα Python. Είναι ικανή να τρέχει πάνω από τη Deeplearning4j, τη Tensorflow και τη Theano. Σχεδιασμένη για γρήγορο πειραματισμό με βαθιά νευρωνικά δίκτυα (deep neural networks), εστιάζει στο να είναι απλή, αρθρωτή και επεκτάσιμη. **[37]**

3.3.8 imutils

Η imutils είναι μια βιβλιοθήκη για τη γλώσσα προγραμματισμού Python, σχεδιασμένη από τον Adrian Rosebrock. Όταν σχεδιάστηκε και κυκλοφόρησε για πρώτη φόρα είχε διαθέσιμες ορισμένες συναρτήσεις γεωμετρικών μετατροπών. Σήμερα διαθέτει μια πληθώρα συναρτήσεων επεξεργασίας εικόνας και αναγνώρισης προτύπων. **[42]**

3.3.9 argparse

Η argparse είναι μια βιβλιοθήκη για τη γλώσσα Python και χρησιμοποιείται για την εισαγωγή ορισμάτων από τη γραμμή εντολών.

ΚΕΦΑΛΑΙΟ 4

ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

4.1 Εισαγωγή

4.1.1 Τι είναι η επεξεργασία εικόνας

Η Επεξεργασία Εικόνας (Image Processing) είναι ο κλάδος της πληροφορικής ο οποίος ασχολείται με την εφαρμογή αλγορίθμων σε εικόνες με χρήση ηλεκτρονικού υπολογιστή. Μέσω των αλγορίθμων αυτών είναι δυνατή η ανάλυση, η αποθήκευση, η μετάδοση, η συμπίεση, η ανάκτηση, η βελτίωση, και η αποκατάστασης μιας εικόνας. **[1] [16]**

Η ΕΕ βρίσκει εφαρμογή σε πολλούς κλάδους και τομείς, όπως στην φωτογραφία, στη γραφιστική, στην ιατρική (υπέρηχους, τομογραφίες), στη ρομποτική (ρομποτική όραση), στη μετάδοση (τηλεδιάσκεψη, τηλεομοιοτυπία), στα συστήματα ασφαλείας, στην αναγνώριση προτύπων (αναγνώριση προσώπων, αναγνώριση σχημάτων), στη τεχνητή νοημοσύνη (μηχανική όραση) κλπ. **[1]**

4.1.2 Ιστορική αναδρομή

Μια από τις πρώτες εφαρμογές των ψηφιακών εικόνων ήταν στη βιομηχανία εφημερίδων, όταν οι εικόνες αρχικά μεταφέρονταν από το Λονδίνο στη Νέα Υόρκη μέσω υποθαλάσσιου καλωδίου. **[2]**

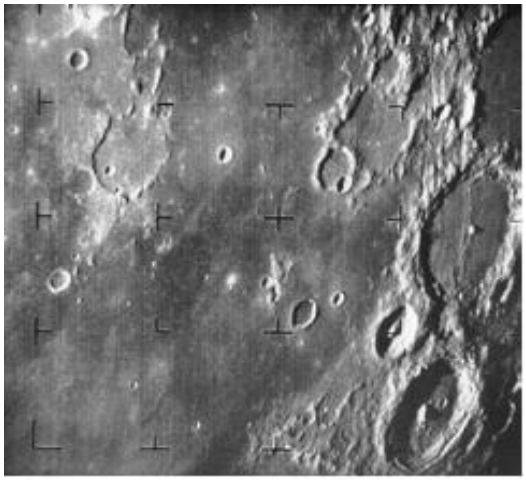
Το 1920 εφευρέθηκε το σύστημα μετάδοσης εικόνας Bartlane (Bartlane cable picture transmission system) από τους Harry G. Bartholomew και Maynard D. McFarlane. Με το σύστημα αυτό οι εικόνες μπορούσαν να μεταδοθούν μέσω του Ατλαντικού ωκεανού σε λιγότερο από 3 ώρες και με 5 αποχρώσεις του γκρι. Να σημειωθεί ότι πριν το σύστημα αυτό μια εικόνα χρειαζόταν περισσότερο από 1 εβδομάδα για να μεταδοθεί μέσω του Ατλαντικού. Το σύστημα Bartlane χρησιμοποιήθηκε για πρώτη φορά για μετάδοση εικόνας μεταξύ Λονδίνου και Νέας Υόρκης το 1921. Η εικόνα εκτυπώθηκε, χωρίς να αποθηκευτεί, στη λήψη της με τεχνική Halftoning. **[2]**

Εικόνα 4.1: Η εικόνα που μεταδόθηκε το 1921 με το σύστημα Bartlane **[2]**

Το 1929 έχουμε μετάδοση εικόνας από το Λονδίνο στη Νέα Υόρκη και κατευθείαν εκτύπωση της, όχι με τη τεχνική Bartlane, αλλά με χρήση τεχνικών φωτογραφικής αναπαραγωγής. Η χρήση των τεχνικών αυτών είχε ως αποτέλεσμα την αύξηση των αποχρώσεων του γκρι από 5 σε 15. **[2]**

Εικόνα 4.2: Η εικόνα που μεταδόθηκε το 1929 με τεχνικές φωτογραφικής αναπαραγωγής και 15 αποχρώσεις του γκρι **[2]**

Το 1964 χρησιμοποιούνται για πρώτη φορά ψηφιακοί υπολογιστές για την επεξεργασία εικόνων του φεγγαριού που λήφθηκαν από το διαστημόπλοιο Ranger 7 με χρήση φωτογραφικής μηχανής, προκειμένου να απαλειφθούν οι παραμορφώσεις που είχαν προκύψει από τη κίνηση του διαστημόπλοιου μαζί με τη φωτογραφική μηχανή. **[2]**



Εικόνα 4.3: Η πρώτη εικόνα από το φεγγάρι από το αμερικάνικο διαστημόπλοιο Ranger 7 **[2]**

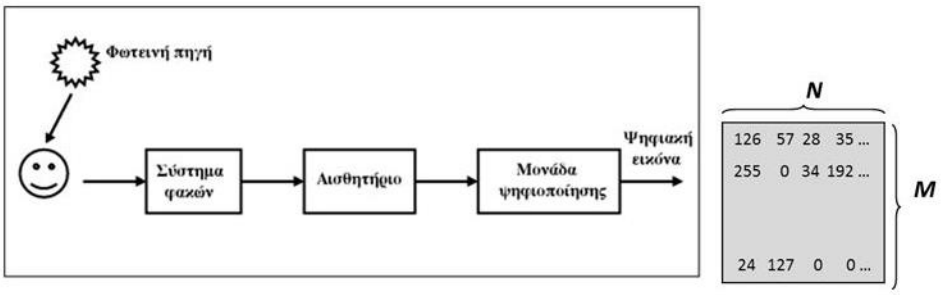
Παράλληλα με τις διαστημικές εφαρμογές, τεχνικές ΕΕ άρχισαν να χρησιμοποιούνται στις ιατρικές εικόνες, στη παρατήρηση απομακρυσμένων πόρων της Γης και στην αστρονομία στα τέλη της δεκαετίας του 1960 και στις αρχές της δεκαετίας του 1970. Στις αρχές της δεκαετίας του 1970 έχουμε την εφεύρεση της αξονικής τομογραφίας (computerized axial tomography) και το 1979 οι εφευρέτες της αξονικής τομογραφίας Sir Godfrey N. Hounsfield και Allan M. Cormack μοιράζονται το βραβείο Nobel ιατρικής για την εφεύρεση αυτή. **[2]**

Από τη δεκαετία του 1980 μέχρι και σήμερα, η ΕΕ θα γνωρίσει ραγδαία εξέλιξη και θα αρχίσει να χρησιμοποιείται σε όλο και περισσότερους τομείς. Ο λόγος είναι ότι τόσο οι υπολογιστές όσο και το απαραίτητο υλικό για ΕΕ αρχίζουν να γίνονται φθηνότεροι και πιο εύκολα διαθέσιμοι εκείνη την εποχή. Το 1994 εγκαθίσταται τεχνολογία ΕΕ για ιατρικές εφαρμογές στο Space Foundation στο Colorado. Τέλος, το 2002 ο Raanan Fattel παρουσίασε το Gradient-Domain Image Processing, ένα νέο τύπο ΕΕ ο οποίος λειτουργεί με βάση τις διαφορές μεταξύ δυο γειτονικών εικονοστοιχείων αντί της τιμής κάθε εικονοστοιχείου. **[16]**

4.2 Ψηφιακή εικόνα

4.2.1 Τι είναι η ψηφιακή εικόνα

Μια εικόνα μπορεί να οριστεί ως μια συνάρτηση δυο διαστάσεων $f(x, y)$, όπου x και y οι συντεταγμένες ενός εικονοστοιχείου (pixel) της εικόνας και η τιμή f για αυτές τις συντεταγμένες x, y καλείται ως φωτεινότητα της εικόνας στο σημείο αυτό ή αλλιώς ως φωτεινότητα του συγκεκριμένου εικονοστοιχείου. Για την ψηφιοποίηση μίας αναλογικής εικόνας, με σκοπό την επεξεργασία της με χρήση ηλεκτρονικού υπολογιστή, απαιτείται η μετατροπή των συντεταγμένων σε διακριτές τιμές μέσω δειγματοληψίας (sampling) και η μετατροπή των συνεχών τιμών φωτεινότητας σε διακριτές μέσω κβάντισης (quantization). Με τον τρόπο αυτό προκύπτει μία ψηφιακή εικόνα. **[9]**



Σχήμα 4.1: Τυπικό σύστημα παραγωγής και μορφή ψηφιακής εικόνας **[17]**

4.2.2 Τύποι εικόνων

Οι εικόνες χωρίζονται σε τρεις κατηγορίες: τις εικόνες φωτεινότητας (grayscale images), τις δυαδικές (binary images) και τις έγχρωμες (color images).

Η εικόνα φωτεινότητας είναι ένας δυσδιάστατος πίνακας $M * N$ εικονοστοιχείων του οποίου οι τιμές αντιπροσωπεύουν τη φωτεινότητα (intensity) του κάθε εικονοστοιχείου. Αναπαριστάται ως :

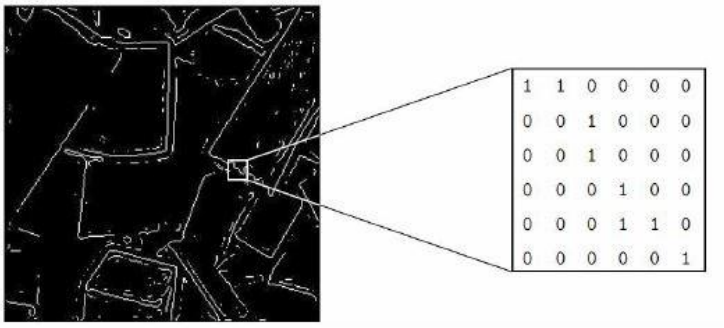
$$I(i,j) \text{ με } i = 0, \dots, M-1 \text{ και } j = 0, \dots, N-1$$

$$\text{όπου } 0 \leq I(i,j) \leq L-1.$$

Το L είναι συνήθως ίσο με 2^k με συνηθέστερη τιμή το $k = 8$ που αντιστοιχεί σε 256 αποχρώσεις του γκρι. Συνήθως οι διαστάσεις μιας εικόνας φωτεινότητας είναι δυνάμεις του δυο (π.χ. 512 x 512). **[1]**

Σχήμα 4.2: Εικόνα φωτεινότητας **[9]**

Η δυαδική εικόνα αποτελεί την απλούστερη μορφή μιας εικόνας. Έχει μόνο δυο τιμές φωτεινότητας, που συνήθως είναι το μαύρο και το άσπρο. Το μαύρο αντιστοιχεί στη τιμή “0” και το άσπρο στη τιμή “1”. Μια δυαδική εικόνα καταλαμβάνει μικρότερη μνήμη και για την επεξεργασία της απαιτείται μικρότερη υπολογιστική ισχύς. **[1]**

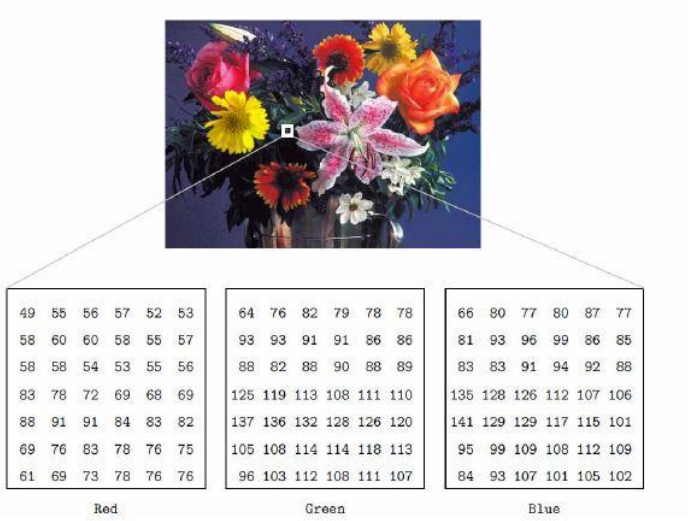


Σχήμα 4.3: Δυαδική εικόνα **[9]**

Η έγχρωμη εικόνα αποτελείται από τρείς εικόνες φωτεινότητας. Αυτό σημαίνει ότι το χρώμα του κάθε εικονοστοιχείου έχει τρεις συνιστώσες που αντιστοιχούν στις φωτεινότητες των τριών εικόνων φωτεινότητας. Μια έγχρωμη εικόνα διαστάσεων $M * N$ αναπαριστάται ως:

$$I_C(i,j) \text{ με } i = 0, \dots, M-1 \text{ και } j = 0, \dots, N-1$$

όπου $0 \leq I_c(i,j) \leq L-1$ για κάθε $c = 1,2,3$. **[1]**



Σχήμα 4.4: Έγχρωμη εικόνα **[9]**

4.3 Οι εικόνες στην OpenCV

Στην ενότητα αυτή θα παρουσιάσουμε βασικές έννοιες, όπως τους υποστηριζόμενους τύπους εικόνων και δεδομένων, βασικές συναρτήσεις χειρισμού εικόνων, αριθμητικές και γεωμετρικές μετατροπές κλπ.

4.3.1 Αριθμητικοί τύποι

Στην OpenCV οι εικόνες είναι πίνακες αριθμητικών στοιχείων του NumPy. Οι αριθμητικοί τύποι που υποστηρίζονται από την OpenCV είναι οι εξής: uint8 (CV_8U), uint16 (CV_16U), int8 (CV_8S), int16 (CV_16S), int32 (CV_32S), float32 (CV_32F) και float64 (CV_64F)

4.3.2 Τύποι εικόνων

Οι τύποι εικόνων που υποστηρίζονται από την OpenCV είναι οι εξής:

- Εικόνες φωτεινότητας
- Δυαδικές εικόνες, συνήθως με τιμές 0 και 255 τύπου uint8
- Έγχρωμες εικόνες, οι οποίες αποτελούνται από τρείς πίνακες. Ο πρώτος αντιστοιχεί στο μπλε χρώμα, ο δεύτερος στο πράσινο χρώμα και ο τρίτος στο κόκκινο χρώμα. Αυτό σημαίνει ότι η OpenCV ακολουθεί το πρότυπο BGR και όχι το RGB.

4.3.3 Μετατροπές αριθμητικών τύπων

Η μετατροπή ενός αριθμητικού τύπου σε έναν άλλον είναι μια συνηθισμένη διαδικασία κατά τον χειρισμό των εικόνων. Για τη μετατροπή ενός αριθμητικού τύπου σε έναν άλλον μπορούμε να χρησιμοποιήσουμε τη συνάρτηση του Numpy **array.astype()**, όπου array η εικόνα ως πίνακας τιμών, δίνοντας ως όρισμα το τελικό τύπο δεδομένων.

Έστω η εικόνα img με τύπο δεδομένων uint8. Για τη μετατροπή από uint8 σε float64 γράφουμε την εξής εντολή: `img.astype(float64).` Ισοδύναμα μπορούμε να γράψουμε: `numpy.float64(img).`

4.3.4 Μετατροπές τύπων εικόνων

Η OpenCV διαθέτει δύο συναρτήσεις για τη μετατροπή μιας έγχρωμης εικόνας σε εικόνα φωτεινότητας και μιας εικόνας φωτεινότητας σε δυαδική αντίστοιχα.

- **cv2.cvtColor()** – μετατροπή έγχρωμης εικόνας σε εικόνα φωτεινότητας ως εξής: `gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`
  Θα δούμε αναλυτικά τη συνάρτηση αυτή στην ενότητα 5.1.8.
- **cv2.threshold()** – μετατροπή εικόνας φωτεινότητας σε δυαδική ως εξής:
  `ret, bin = cv2.threshold(img,127,255,cv2.THRESH_BINARY)`
  Θα δούμε αναλυτικά τη συνάρτηση αυτή στην ενότητα 5.2.1.4

4.3.5 Ανάγνωση εικόνας

Η ανάγνωση μιας εικόνας γίνεται με τη συνάρτηση **cv2.imread()** και συντάσσεται ως εξής:

cv2.imread(filename, flags)

Παράμετροι:

- **filename:** Όνομα του αρχείου εικόνας προς ανάγνωση. Οι τύπο εικόνων που υποστηρίζει η OpenCV είναι οι εξής: *.bmp, *.dib, *.jpeg, *.jpg, *.jpe, *.jp2, *.png, *.tiff, *.tif, *.webp, *.pbm, *.pgm, *.ppm, *.sr και *.ras
- **flags:** Τρόπος ανάγνωσης της εικόνας. Οι συνηθέστερες τιμές είναι οι εξής:
  - cv2.IMREAD_COLOR: Η εικόνα διαβάζεται ως έγχρωμη. Αποτελεί τη εξ' ορισμού τιμή. (1)
  - cv2.IMREAD_GRAYSCALE: Η εικόνα διαβάζεται ως εικόνα φωτεινότητας. (0)

- cv2.IMREAD_UNCHANGED: Η εικόνα διαβάζεται ως έχει. (-1)

Η ανάγνωση μιας εικόνας γίνεται με τις εξής εντολές:

```
import numpy as np
import cv2

# Load a color image in grayscale mode
img = cv2.imread('football.jpg', 0)
```

4.3.6 Εμφάνιση εικόνας

Στην ενότητα αυτή θα δούμε πως μπορούμε να εμφανίσουμε μια εικόνα με χρήση της βιβλιοθήκης OpenCV και με χρήση της βιβλιοθήκης Matplotlib.

Εμφάνιση εικόνας με την OpenCV

Η OpenCV διαθέτει τη συνάρτηση **cv2.imshow()** για την εμφάνιση μιας εικόνας και συντάσσεται ως εξής:

cv2.imshow(windowname, img)

Παράμετροι:

- **windowname:** Όνομα του παραθύρου στο οποίο θα εμφανιστεί η εικόνα.
- **img:** Εικόνα προς εμφάνιση

Η εμφάνιση μιας εικόνας φωτεινότητας γίνεται με τις εξής εντολές:

```
import numpy as np
import cv2

# Load a grayscale image
img = cv2.imread('./Desktop/images/street.tif, 0)
# Show the image
cv2.imshow('Grayscale Image', img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

Εικόνα 4.4: Εμφάνιση εικόνας φωτεινότητας με την OpenCV

Εμφάνιση εικόνας με τη Matplotlib

Η Matplotlib διαθέτει τη συνάρτηση **matplotlib.pyplot.imshow()** για την εμφάνιση μιας εικόνας και συντάσσεται ως εξής:

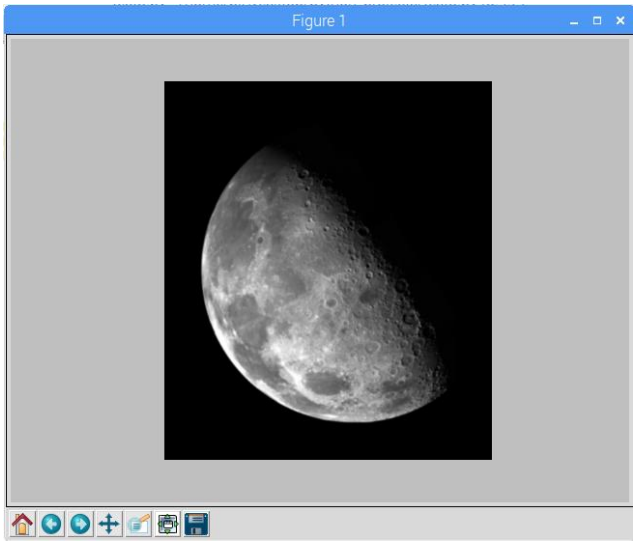
matplotlib.pyplot.imshow(img, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin, extent, shape, filternorm, filterrad, imlim, resample, url, hold, data)

Παράμετροι (θα ασχοληθούμε με τις δύο πρώτες):

- **img:** Εικόνα προς εμφάνιση.
- **cmap:** Πίνακας χρωμάτων (colormap).

Η εμφάνιση μιας εικόνας φωτεινότητας με χρήση του matplotlib γίνεται με τις εξής εντολές:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('./Desktop/images/moon.tif',0)
plt.axis("off") # remove numbered axis
plt.imshow(img, cmap = 'gray'); plt.show()
```

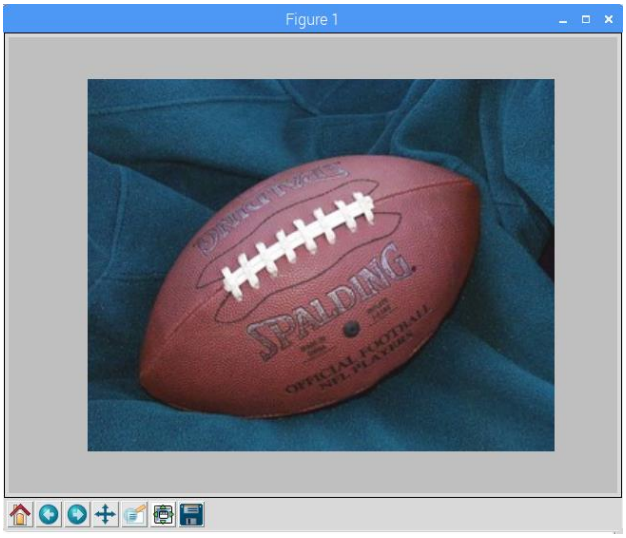


Εικόνα 4.5: Εμφάνιση εικόνας φωτεινότητας με τη Matplotlib

Για την εμφάνιση μιας έγχρωμης εικόνας, θα πρέπει πρώτα να τη μετατρέψουμε από BGR σε RGB διότι χειρίζεται τις έγχρωμες εικόνες με το πρότυπο RGB. Η μετατροπή αυτή μπορεί να γίνει με τη συνάρτηση **cv2.cvtColor()** ως εξής:

```
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('./Desktop/images/football.jpg')
img2 = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.axis("off") # remove numbered axis
plt.imshow(img); plt.show()
```



Εικόνα 4.6: Εμφάνιση έγχρωμης εικόνας με τη Matplotlib

4.3.7 Αποθήκευση εικόνας

Η αποθήκευση μιας εικόνας γίνεται με τη συνάρτηση **cv2.imwrite()** και συντάσσεται ως εξής:

cv2.imwrite(filename, img ,params)

Παράμετροι:

- **filename:** Όνομα του αρχείου εικόνας που θα αποθηκευτεί.
- **img:** Εικόνα προς αποθήκευση.
- **params:** Παράμετροι αποθήκευσης.

Η αποθήκευση μιας εικόνας μπορεί να γίνει με τις εξής εντολές:

```
import numpy as np
import cv2

# Load a color image in grayscale mode
img = cv2.imread('./Desktop/images/football.jpg', 0)

#Save image in grayscale mode
cv2.imwrite('./Desktop/images/footballgray.jpg', img)
```

4.3.8 Εμφάνιση χαρακτηριστικών εικόνας

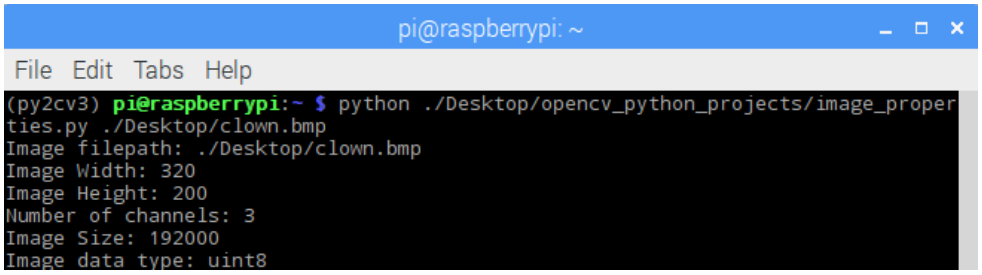
Για την εύρεση των διαστάσεων μιας εικόνας χρησιμοποιούμε την συνάρτηση **array.shape**. Η εντολή αυτή επιστρέφει τον αριθμό των γραμμών (rows), τον αριθμό των στηλών (columns) και τον αριθμό των καναλιών (channels) εάν η εικόνα είναι έγχρωμη. Αν πρόκειται για εικόνα φωτεινότητας τότε η εντολή αυτή επιστρέφει μόνο τον αριθμό των γραμμών και τον αριθμό των στηλών.

Για την εύρεση του αριθμού των εικονοστοιχείων μιας εικόνας χρησιμοποιούμε την συνάρτηση **array.size**.

Για την εύρεση του αριθμητικού τύπου των δεδομένων μιας εικόνας χρησιμοποιούμε την συνάρτηση **array.dtype**.

Παρακάτω παρατίθεται ένα πρόγραμμα εμφάνισης των χαρακτηριστικών μιας έγχρωμης εικόνας.

```
import numpy as np
import cv2
import sys

img = cv2.imread(sys.argv[1])
height, width, channels = img.shape
# Path of the image
print('Image filepath:', sys.argv[1])
# Shape of the image
print ('Image Width:', width)
print ('Image Height:', height)
print ('Number of channels:', channels)
# Number of pixels of the image
print ('Image Size:', img.size)
# Pixel's value data type
print ('Image data type:', img.dtype)
```



Εικόνα 4.7: Εμφάνιση χαρακτηριστικών εικόνας

4.3.9 Το ιστόγραμμα

Το ιστόγραμμα είναι ένα γράφημα το οποίο μας δίνει δείχνει τη συχνότητα εμφάνισης μια φωτεινότητας σε μια εικόνα. Εκτός από το ιστόγραμμα, υπάρχει και το κανονικοποιημένο ιστόγραμμα (normalized histogram). Το τελευταίο είναι ένα γράφημα το οποίο μας δείχνει τη πιθανότητα εμφάνισης μιας φωτεινότητας σε μια εικόνα. Μπορούμε να υπολογίσουμε το κανονικοποιημένο ιστόγραμμα με βάση τη παρακάτω σχέση:

$$Κανονικοποιημένο\ Ιστόγραμμα = \frac{Ιστόγραμμα}{Αριθμό\ εικονοστοιχείων}$$

Εύρεση ιστογράμματος

Η εύρεση του ιστογράμματος μιας εικόνας γίνεται με τη συνάρτηση **cv2.calcHist()** και συντάσσεται ως εξής:

cv2.calcHist(img, channels, mask, histSize, range)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **channels:** Αριθμός καναλιού στο οποίο θα υπολογιστεί το ιστόγραμμα ([0] για εικόνα φωτεινότητας, [0], [1], [2] για τη μπλε, τη πράσινη και τη κόκκινη συνιστώσα μιας έγχρωμης εικόνας).
- **mask:** Προαιρετική μάσκα λειτουργίας.
- **histSize:** Αριθμός τιμών για την αναπαράσταση του ιστογράμματος. ([256] για ολόκληρο το ιστόγραμμα)
- **range:** Εύρος σχεδίασης του ιστογράμματος ([0,255] για ολόκληρο το ιστόγραμμα)

Ο υπολογισμός των ιστογραμμάτων μιας εικόνας γίνεται με τις εξής εντολές:

```
import numpy as np
import cv2

img = cv2.imread('./Desktop/images/pentagon.png',0)
# Calculate the histogram
hist = cv2.calcHist([img],[0],None,[256],[0,255])
# Calculate the normalized histogram
norm_hist = np.divide(hist, img.size)
```

Εμφάνιση ιστογράμματος

Η OpenCV δε διαθέτει συνάρτηση plot για την εμφάνιση γραφημάτων. Για το λόγο αυτό, για την εμφάνιση του ιστογράμματος θα χρησιμοποιήσουμε τη Matplotlib. Διαθέτει μια συνάρτηση η οποία υπολογίζει και εμφανίζει απευθείας το ιστόγραμμα (αυτό σημαίνει ότι δε χρειάζεται να υπολογιστεί αρχικά με την **cv2.calcHist()**) και είναι η **matplotlib.pyplot.hist()** η οποία συντάσσεται ως εξής:

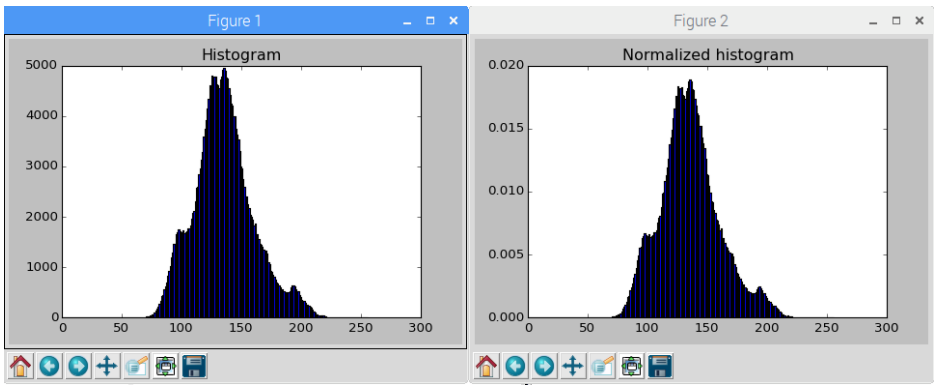
matplotlib.pyplot.hist(img, bins, range, normed, weights, cumulative, bottom, histtype, align, orientation, rwidth, log, color=None, label, stacked, hold, data)

Παράμετροι (θα ασχοληθούμε με τις πρώτες τέσσερις):

- **img:** Εικόνα εισόδου.
- **bins:** Βλέπε παράμετρο histSize της **cv2.calcHist()**.
- **range:** Βλέπε παράμετρο range της **cv2.calcHist()**.
- **normed:** Υπολογισμός ή όχι κανονικοποιημένου ιστογράμματος

Ο υπολογισμός και η εμφάνιση των ιστογραμμάτων με το matplotlib γίνεται με τις εξής εντολές:

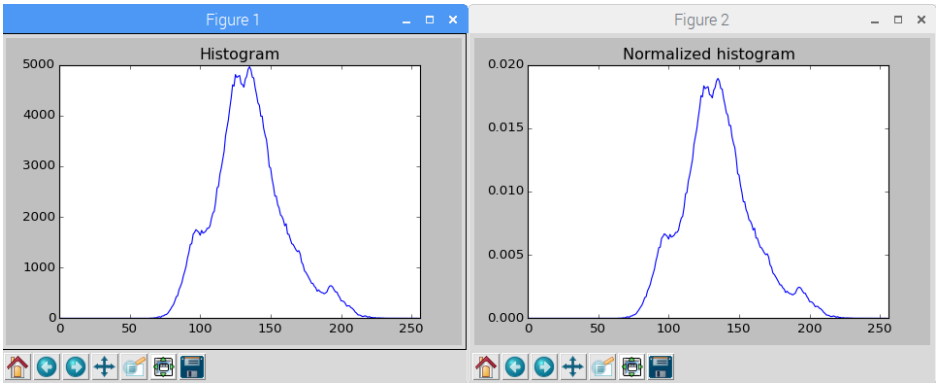
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('./Desktop/images/pentagon.png',0)

plt.hist(img.ravel(),256,[0,255]); plt.title('Histogram'); plt.figure()
plt.hist(img.ravel(),256,[0,255],True);
plt.title('Normalized histogram'); plt.show()
```



Εικόνα 4.8: Υπολογισμός και εμφάνιση ιστογραμμάτων με τη Matplotlib

Ένας άλλος τρόπος είναι να χρησιμοποιήσουμε τη συνάρτηση **matplotlib.pyplot.plot()**. Σε αυτή τη περίπτωση θα πρέπει να έχουμε υπολογίσει το ιστόγραμμα με τη συνάρτηση **cv2.calcHist()**. Αυτός ο τρόπος είναι ιδιαίτερα χρήσιμος όταν θέλουμε να υπολογίσουμε και εμφανίσουμε ιστογράμματα έγχρωμων εικόνων καθώς και στη περίπτωση που θέλουμε να έχουμε πρόσβαση σε συγκεκριμένες τιμές, κάτι που απαιτεί την ύπαρξη του ιστογράμματος ως πίνακα τιμών.

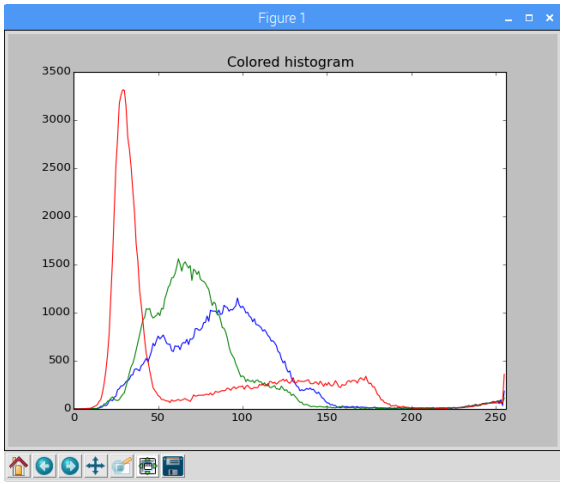
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('./Desktop/images/pentagon.png',0)

# Calculate histograms with OpenCV
# Calculate the histogram
hist = cv2.calcHist([img],[0],None,[256],[0,255])
# Calculate the normalized histogram
px = img.size
norm_hist = np.divide(hist, px)

# Plot histograms with Matplotlib
plt.plot(hist); plt.xlim([0,256]); plt.title('Histogram'); plt.figure()
plt.plot(norm_hist);        plt.xlim([0,256]);        plt.title('Normalized
histogram'); plt.show()
```



Εικόνα 4.9: Υπολογισμός και εμφάνιση ιστογραμμάτων με την OpenCV και τη Matplotlib αντίστοιχα

Τέλος παρατίθεται ένα παράδειγμα υπολογισμού και εμφάνισης του ιστογράμματος μιας έγχρωμης εικόνας.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('./Desktop/images/football.jpg')
color = ('b','g','r')
for i, col in enumerate(color):
    hist = cv2.calcHist([img],[i],None,[256],[0,256])
    plt.plot(hist, color = col);plt.xlim([0,256]); plt.title('Colored
histogram')
plt.show()
```



Εικόνα 4.10: Ιστόγραμμα έγχρωμης εικόνας

Σημείωση: Για λόγους απλότητας και συντομίας, στις επόμενες ενότητες θα παραλείπονται οι εντολές εισαγωγής βιβλιοθηκών και εμφάνισης εικόνων και ιστογραμμάτων.

4.4 Βελτιστοποίηση εικόνας

4.4.1 Επέκταση της αντίθεσης

Η επέκταση της αντίθεσης (contrast stretching) είναι η διαδικασία η οποία μεταβάλει το εύρος των φωτεινοτήτων μιας εικόνας και εφαρμόζεται σε εικόνες με χαμηλή αντίθεση.

Η OpenCV διαθέτει τη συνάρτηση **cv2.normalize()** με την οποία μπορούμε να κάνουμε επέκταση της αντίθεσης μιας εικόνας και συντάσσεται ως εξής:

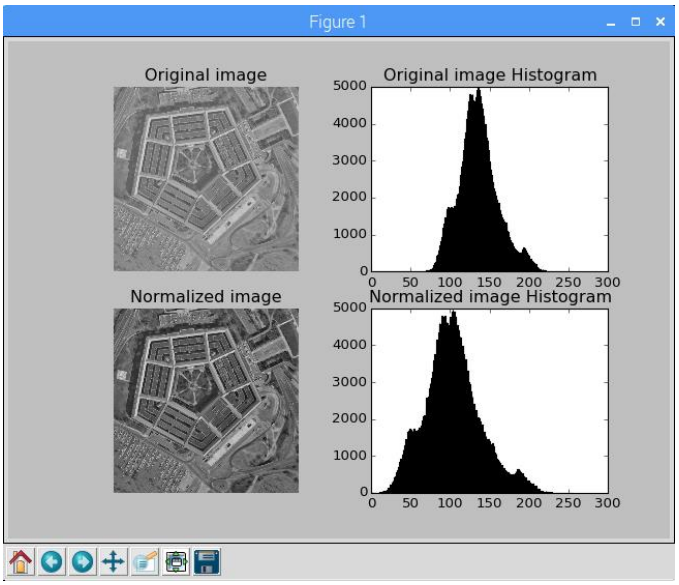
cv2.normalize(img, dst, alpha, beta, norm_type, dtype, mask)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **dst:** Εικόνα εξόδου.
- **alpha:** Ελάχιστη τιμή φωτεινότητας της εικόνας εξόδου.
- **beta:** Μέγιστη τιμή φωτεινότητας της εικόνας εξόδου.
- **norm_type:** Τύπος κανονικοποίησης. Για επέκταση της αντίθεσης χρησιμοποιείται η τιμή cv2.NORM_MINMAX
- **dtype:** Τύπος δεδομένων της εικόνας εξόδου.
- **mask:** Προαιρετική μάσκα λειτουργίας.

Η επέκταση της αντίθεσης σε μία εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/pentagon.png',0)
img_adj = np.copy(img)
cv2.normalize(img_adj, img_adj,0,255,cv2.NORM_MINMAX)
```



Εικόνα 4.11: Επέκταση της αντίθεσης

4.4.2 Κατωφλίωση

Η κατωφλίωση (thresholding) αποτελεί την απλούστερη μορφή του κοψίματος της φωτεινότητας. Χρησιμοποιείται μια τιμή κατωφλίου και ανάλογα με το τύπο της κατωφλίωσης οι τιμές των φωτεινοτήτων μεταβάλλονται αναλόγως.

Η OpenCV διαθέτει τη συνάρτηση **cv2.threshold()** για τη κατωφλίωση μιας εικόνας, ή οποία συντάσσεται ως εξής:

cv2.threshold(img, thr, maxval, type)

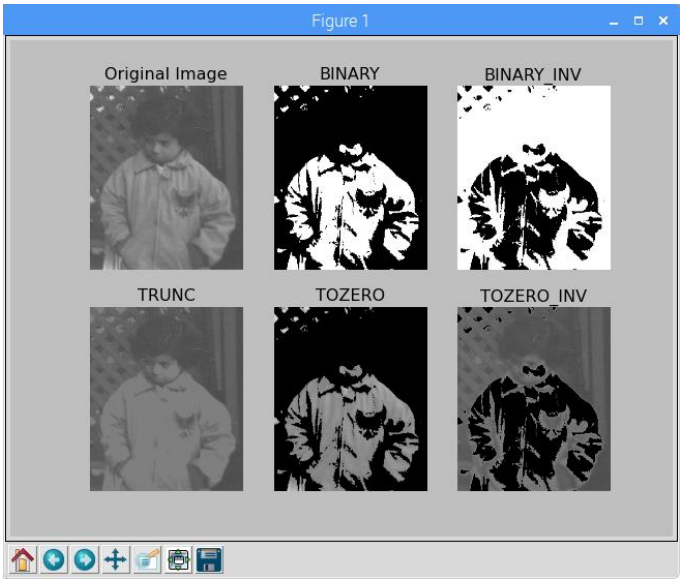
Παράμετροι:

- **img:** Εικόνα εισόδου.
- **thr:** Τιμή κατωφλίου.
- **maxval:** Τιμή φωτεινότητας της εικόνας εξόδου αν η τιμή φωτεινότητας της εικόνας εισόδου είναι μεγαλύτερη ή ίση της thr.
- **type:** Τύπος κατωφλίωσης. Δέχεται τις εξής τιμές:
  - cv2.THRESH_BINARY: $dst(x,y) = \begin{cases} maxval\ εάν\ src(x,y) > thresh \\ 0\ διαφορετικά \end{cases}$
  - cv2.THRESH_BINARY_INV: $dst(x,y) = \begin{cases} 0\ εάν\ src(x,y) > thresh \\ maxval\ διαφορετικά \end{cases}$
  - cv2.THRESH_TRUNC: $dst(x,y) = \begin{cases} threshold\ εάν\ src(x,y) > thresh \\ src(x,y)\ διαφορετικά \end{cases}$
  - cv2.THRESH_TOZERO: $dst(x,y) = \begin{cases} src(x,y)\ εάν\ src(x,y) > thresh \\ 0\ διαφορετικά \end{cases}$
  - cv2.THRESH_TOZERO_INV: $dst(x,y) = \begin{cases} 0\ εάν\ src(x,y) > thresh \\ src(x,y)\ διαφορετικά \end{cases}$

Η συνάρτηση cv2.threshold() επιστρέφει δύο τιμές, τη τιμή retval, η οποία ισοδυναμεί με τη τιμή thresh και την εικόνα μετά την εφαρμογή της κατωφλίωσης.

Η κατωφλίωση πραγματοποιείται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/pout.tif',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
```



Εικόνα 4.12: Κατωφλίωση

Στο παραπάνω παράδειγμα, χρησιμοποιήσαμε μια καθολική τιμή ως τιμή κατωφλίου, δηλαδή η φωτεινότητα κάθε εικονοστοιχείου μεταβάλλεται με βάση αυτή τη τιμή. Για το λόγο αυτό, η παραπάνω μέθοδος ονομάζεται και καθολική κατωφλίωση (global thresholding). Όμως, σε εικόνες οι οποίες έχουν διαφορετικές καταστάσεις φωτεινότητας (ένα κομμάτι της εικόνας είναι αρκετά σκοτεινό, ενώ ένα άλλο είναι αρκετά φωτεινό), η παραπάνω μέθοδος δε θα δώσει καλά αποτελέσματα. Για το λόγο αυτό εφαρμόζουμε προσαρμοσμένη κατωφλίωση, όπου υπολογίζεται η τιμή κατωφλίου για μια μικρή περιοχή μια εικόνας, έχοντας έτσι διαφορετικές τιμές κατωφλίου για διαφορετικές περιοχές στην ίδια εικόνα.

Η OpenCV διαθέτει τη συνάρτηση **cv2.adaptiveThreshold()** για τη προσαρμοσμένη κατωφλίωση μιας εικόνας, ή οποία συντάσσεται ως εξής:

cv2.adaptiveThreshold(img, maxval, adaptiveMethod, type, blockSize, C)

Παράμετροι:

- **adaptiveMethod:** Αλγόριθμος για τον υπολογισμό της τιμής κατωφλίου. Δέχεται τις εξής τιμές:
  - cv2.ADAPTIVE_THRESH_MEAN_C: η τιμή κατωφλίου του $I(x, y)$ ισούται με το μέσο όρο (mean) του $blockSize * blockSize$ μείων C.
  - cv2.ADAPTIVE_THRESH_GAUSSIAN_C: η τιμή κατωφλίου του $I(x, y)$ ισούται με το άθροισμα ειδικών βαρών (weighted sum) του $blockSize * blockSize$ μείων C.
- **type:** Τύπος κατωφλίωσης. Δέχεται τις εξής τιμές:

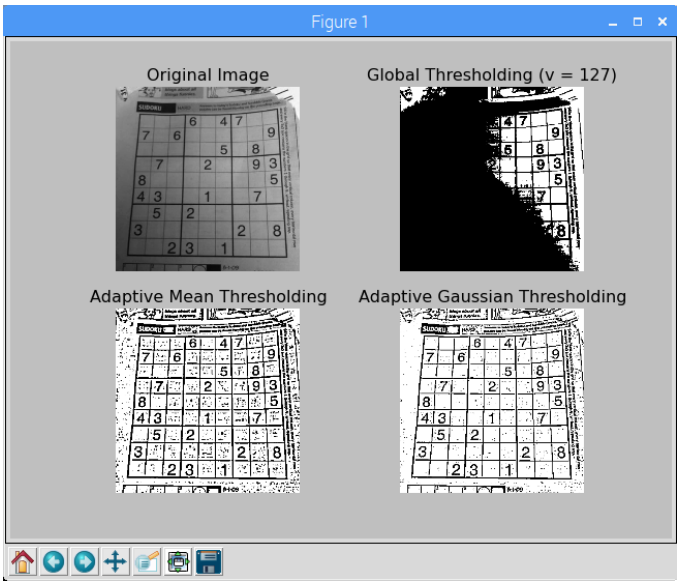
- cv2.THRESH_BINARY: $dst(x,y) = \begin{cases} maxval\ \varepsilon\acute{\alpha}\nu\ src(x,y) > thresh \\ 0\ \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$
- cv2.THRESH_BINARY_INV: $dst(x,y) = \begin{cases} 0\ \varepsilon\acute{\alpha}\nu\ src(x,y) > thresh \\ maxval\ \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases}$

- **blockSize:** Μέγεθος της γειτονικής περιοχής ενός εικονοστοιχείου και χρησιμοποιείται για τον υπολογισμό της τιμής κατωφλίου για το εικονοστοιχείο αυτό. Δέχεται τιμές μεγαλύτερες ή ίσες του 3.
- Οι υπόλοιπες παράμετροι είναι ίδιες με τις αντίστοιχες της **cv2.threshold()**.

Σε αντίθεση με τη συνάρτηση **cv2.threshold()**, η συνάρτηση cv2.adaptiveThreshold() επιστρέφει μια τιμή, την εικόνα μετά την εφαρμογή της προσαρμοσμένης κατωφλίωσης.

Στο παρακάτω παράδειγμα διαβάζουμε μια εικόνα φωτεινότητας και εφαρμόζουμε σε αυτή καθολική και προσαρμοσμένη κατωφλίωση.

```
img = cv2.imread('./Desktop/images/sudoku.jpg',0)
# Global thresholding
ret, thresh = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
# Adaptive mean thresholding
ad_thresh1=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.
THRESH_BINARY,11,2)
# Adaptive gaussian thresholding
ad_thresh2=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY,11,2)
```



Εικόνα 4.13: Προσαρμοσμένη κατωφλίωση

Τέλος, θα αναφερθούμε στη κατωφλίωση με τη τεχνική Otsu. Η κατωφλίωση με τη τεχνική Otsu εφαρμόζεται σε εικόνες οι οποίες περιέχουν δυο κλάσεις εικονοστοιχείων (οι εικόνες αυτές ονομάζονται bimodal). Η μια αφορά τα εικονοστοιχεία που βρίσκονται στο προσκήνιο (foreground pixels) και η άλλη τα εικονοστοιχεία που βρίσκονται στο παρασκήνιο της εικόνας (background pixels). Η μέθοδος αυτή υπολογίζει μια τιμή κατωφλίου που ελαχιστοποιεί την ενέργεια εντός της κλάσης (intra-class variance) το οποίο ορίζεται ως το άθροισμα με ειδικά βάρη των ενεργειών των δύο κλάσεων:

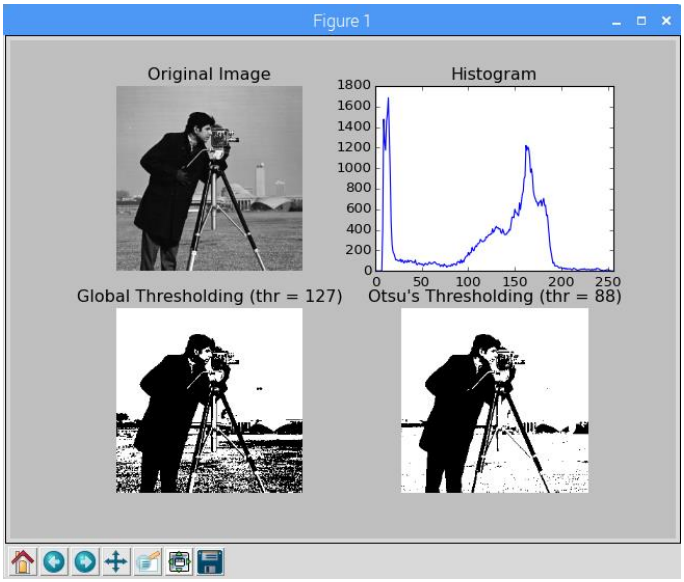
$$\sigma_{\omega}^2(t) = \omega_0(t) * \sigma_0^2(t) + \omega_1(t) * \sigma_1^2(t) \text{ όπου}$$

τα ω_0 και ω_1 είναι οι πιθανότητες των δύο κλάσεων που χωρίζονται από τη τιμή κατωφλίου t και τα σ_0^2 και σ_1^2 είναι οι ενέργειες των δυο κλάσεων.

Για τη κατωφλίωση με τη μέθοδο Otsu χρησιμοποιούμε τη συνάρτηση **cv2.threshold()** περνώντας μια επιπλέον τιμή, την cv2.THRESH_OTSU, και ορίζοντας τη παράμετρο thresh ίση με 0. Τέλος, η παράμετρος retval έχει τη τιμή κατωφλίου που υπολογίστηκε με τη τεχνική Otsu.

Στο παρακάτω παράδειγμα διαβάζουμε μια εικόνα φωτεινότητας και εφαρμόζουμε σε αυτή καθολική κατωφλίωση και κατωφλίωση με τη μέθοδο Otsu.

```
img = cv2.imread('./Desktop/images/cameraman.tif',0)
hist = cv2.calcHist([img],[0],None,[256],[0,255])
# Global thresholding
ret, thresh = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
# Otsu's thresholding
ret,otsu_thresh=cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTS
U)
```



Εικόνα 4.14: Κατωφλίωση με τη μέθοδο Otsu

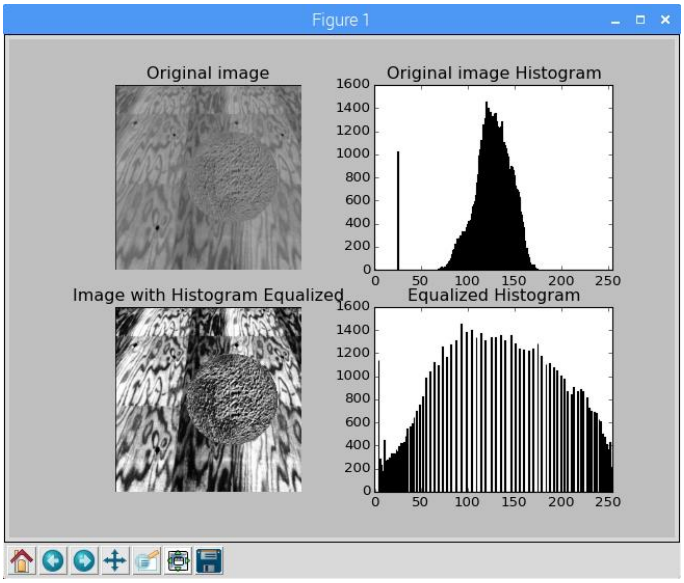
4.4.3 Εξισορρόπηση ιστογράμματος

Η εξισορρόπηση ιστογράμματος (histogram equalization) είναι μια τεχνική η οποία έχει ως σκοπό την ομοιόμορφη κατανομή των επιπέδων φωτεινότητας μιας εικόνας ή, με άλλα λόγια, να εμπλουτίσει την εικόνα με περισσότερες αποχρώσεις του γκρι. Η τεχνική αυτή εφαρμόζεται σε εικόνες, όπου το ιστόγραμμα τους είναι περιορισμένο σε μια στενή περιοχή φωτεινοτήτων, για παράδειγμα το ιστόγραμμα να είναι περιορισμένο κοντά στο μηδέν, οπότε η εικόνα να ναι σκοτεινή.

Η OpenCV διαθέτει τη συνάρτηση **cv2.equalizeHist()** για την υλοποίηση της εξισορρόπησης του ιστογράμματος η οποία δέχεται σαν μοναδικό όρισμα την εικόνα στην οποία θέλουμε να εφαρμόσουμε εξισορρόπηση ιστογράμματος.

Στο παρακάτω πρόγραμμα διαβάζουμε μια εικόνα φωτεινότητας και εφαρμόζουμε σε αυτή εξισορρόπηση του ιστογράμματος.

```
img = cv2.imread('./Desktop/images/sphere.png',0)
hist_equ = cv2.equalizeHist(img)
```



Εικόνα 4.15: Εξισορρόπηση ιστογράμματος

4.5 Φιλτράρισμα εικόνας

4.5.1 Εφαρμογή φίλτρων με τη συνάρτηση cv2.filter2D

Η OpenCV διαθέτει συναρτήσεις για την εφαρμογή αρκετών φίλτρων. Μπορούμε όμως να εφαρμόσουμε οποιοδήποτε φίλτρο, δημιουργώντας το σε μορφή μάσκας με τη συνάρτηση **cv2.filter2D()**, η οποία συντάσσεται ως εξής:

cv2.filter2D(img, dtype, kernel, anchor, delta, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **dtype:** Τύπος δεδομένων της εικόνας εξόδου. Με τιμή -1, έχει ίδιο τύπο δεδομένων με την εικόνα εισόδου. Ισχύουν τα εξής:
  - img.type() = CV_8U, dtype = -1/CV_16S/CV_32F/CV_64F
  - img.type() = CV_16U/CV_16S, dtype = -1/CV_32F/CV_64F
  - img.type() = CV_32F, dtype = -1/CV_32F/CV_64F
  - img.type() = CV_64F, dtype = -1/CV_64F
- **kernel:** Φίλτρο σε μορφή μάσκας.
- **anchor:** Σημείο της μάσκας όπου θα τοποθετείται το εικονοστοιχείο του οποίου η τιμή του θα τροποποιηθεί μετά το φιλτράρισμα. Η τιμή (-1,-1) δηλώνει πως το σημείο αυτό θα είναι το κέντρο της μάσκας.
- **delta:** Προαιρετική σταθερά που προστίθεται στο αποτέλεσμα του φιλτραρίσματος

- **borderType:** Μέθοδος επέκτασης της εικόνας.

4.5.2 Φίλτρο μέσης τιμής (Averaging or Mean Filter)

Το φίλτρο μέσης τιμής 3x3 έχει την εξής μορφή:

$$K = \frac{1}{a_1 + \cdots + a_9}\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}$$

όπου $a_1 = a_2 = a_3 = a_4 = a_5 = a_6 = a_7 = a_8 = a_9 = 1$

Αν ορισμένοι συντελεστές έχουν τιμή διαφορετική από 1, τότε το φίλτρο θα ήταν μέσης τιμής ειδικών βαρών (weighted averaging filter).

Για την εφαρμογή φίλτρου μέσης τιμής σε μια εικόνα, η OpenCV διαθέτει τη συνάρτηση **cv2.blur()** και συντάσσεται ως εξής:

cv2.blur(img, ksize, anchor, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου με τύπο δεδομένων CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
- **ksize:** Μέγεθος του φίλτρου.
- Οι υπόλοιπες παράμετροι είναι ίδιες με τις αντίστοιχες της συνάρτησης **cv2.filter2D()**.

Για την εφαρμογή φίλτρου μέσης τιμής ειδικών βαρών σε μια εικόνα, η OpenCV διαθέτει τη συνάρτηση **cv2.GaussianBlur()** και συντάσσεται ως εξής:

cv2.**GaussianBlur**(img, ksize, sigmaX, sigmaY, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου με τύπο δεδομένων CV_8U, CV_16U, CV_16S, CV_32F ή CV_64F.
- **ksize:** Μέγεθος του φίλτρου.
- **sigmaX:** Τυπική απόκλιση στον άξονα X.
- **sigmaY:** Τυπική απόκλιση στον άξονα Y.
- **borderType:** Βλέπε συνάρτηση **cv2.filter2D()**.

Η εφαρμογή φίλτρου μέσης τιμής σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/cameraman.tif', 0)
# Averaging filtering with cv2.filter2D()
kernel = np.ones((3,3),np.float32)/9 # Create the averaging filter
dst = cv2.filter2D(img,-1,kernel)
# Averaging filtering with cv2.blur()
blur = cv2.blur(img, (3,3))
```

Εικόνα 4.16: Εφαρμογή φίλτρου μέσης τιμής

Αντίστοιχα, η εφαρμογή φίλτρου μέσης τιμής ειδικών βαρών σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/cameraman.tif', 0)
# Gaussian filtering with cv2.filter2D()
kernel = np.array(([1, 2, 1], [2, 4, 2], [1, 2, 1]), np.float32)/16 #
Create the gaussian filter
dst = cv2.filter2D(img,-1,kernel)
# Gaussian filtering with cv2.blur()
blur = cv2.GaussianBlur(img, (3,3), 0)
```

Εικόνα 4.17: Εφαρμογή φίλτρου μέσης τιμής ειδικών βαρών

4.5.3 Φίλτρο δεύτερης παραγώγου (Laplacian Filter)

Το φίλτρο δεύτερης παραγώγου χρησιμοποιείται για τον εμπλουτισμό μιας εικόνας, αναδεικνύοντας τις απότομες μεταβολές φωτεινότητας της και ταυτόχρονα εξομαλύνοντας τις περιοχές φωτεινότητας που μεταβάλλονται ομαλά.

Το φίλτρο δεύτερης παραγώγου 3x3 έχει την εξής μορφή:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Η OpenCV διαθέτει τη συνάρτηση **cv2.Laplacian()** για την εφαρμογή φίλτρου δεύτερης παραγώγου σε μια εικόνα και συντάσσεται ως εξής:

cv2.Laplacian(img, dtype, ksize, scale, delta, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **dtype:** Τύπος δεδομένων της εικόνας εξόδου
- **ksize:** Το μέγεθος που θα χρησιμοποιηθεί για τον υπολογισμό του φίλτρου δεύτερης παραγώγου και δέχεται θετικές και περιττές τιμές. Με ksize > 1 ο υπολογισμός του φίλτρου γίνεται σύμφωνα με τον τελεστή Laplacian:

  $$\nabla^2 src = \frac{\partial^2 src}{\partial x^2} + \frac{\partial^2 src}{\partial y^2}$$

  Με ksize = 1 χρησιμοποιείται ως φίλτρο στην εικόνα ή εξής μάσκα:

  $$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

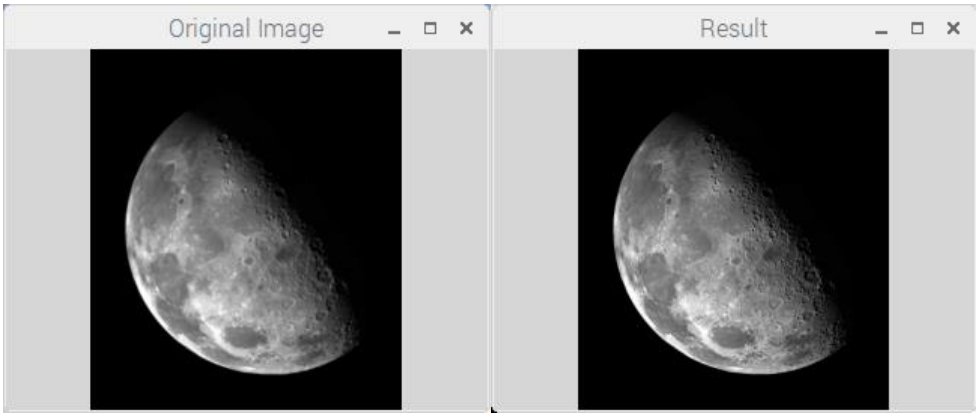
- **scale:** Προαιρετικός συντελεστής κλίμακας για τον υπολογισμό των δεύτερων παραγώγων.
- **delta:** Βλέπε συνάρτηση **cv2.filter2D()**.
- **borderType:** Βλέπε συνάρτηση **cv2.filter2D()**.

Η εφαρμογή φίλτρου δεύτερης παραγώγου σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/moon.tif',0)
# Laplacian filtering with cv2.filter2D()
kernel = np.array(([0, 1, 0], [1, -4, 1], [0, 1, 0]), np.float32)
dst = cv2.filter2D(img, cv2.CV_8U,kernel)
```

```
# Laplacian filtering with cv2.Laplacian()
laplacian = cv2.Laplacian(img,cv2.CV_8U,1)
res = cv2.subtract(img,laplacian)
```

Σημείωση: Αφαιρώντας από την αρχική εικόνα, την εικόνα που προέκυψε από την εφαρμογή του φίλτρου, παίρνουμε την νέα εμπλουτισμένη εικόνα



Εικόνα 4.18: Εφαρμογή φίλτρου δεύτερης παραγώγου

4.5.4 Φίλτρο πρώτης παραγώγου (Sobel Filter)

Το φίλτρο πρώτης παραγώγου χρησιμοποιείται συνήθως για την ανάδειξη των ακμών μιας εικόνας (edge detection), κάνοντας τες πιο εμφανείς.

Το φίλτρο αυτό αποτελείται από δυο μάσκες: (α) κάθετης παραγώγου, για την ανάδειξη των οριζόντιων ακμών και (β) οριζόντιας παραγώγου, για την ανάδειξη των κάθετων ακμών της εικόνας.

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

(α) (β)

Η OpenCV διαθέτει τη συνάρτηση **cv2.Sobel()** για την εφαρμογή φίλτρου πρώτης παραγώγου σε μια εικόνα και συντάσσεται ως εξής:

cv2.Sobel(img, dtype, dx, dy, ksize, scale, delta, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **dtype:** Τύπος δεδομένων της εικόνας εξόδου. Βλέπε συνάρτηση **cv2.filter2D()**.
- **dx:** Υπολογισμός οριζόντιας πρώτης παραγώγου (dx=1 και dy=0).

- **dy:** Υπολογισμός κάθετης πρώτης παραγώγου (dx=0 και dy=1).
- **Ksize:** Μέγεθος του φίλτρου. Ισχύουν τα εξής:
  - με ksize = 3, 5 ή 7, η μάσκα έχει μέγεθος ksize x ksize.
  - Με ksize = 1, χρησιμοποιείται μάσκα 3x1 ή 1x3.
  - Με ksize = 3 χρησιμοποιείται μια από τις δύο παρακάτω μάσκες
    - με dx = 1 και dy = 0 χρησιμοποιείται η πρώτη μάσκα
    - με dx = 0 και dy = 1 χρησιμοποιείται η δεύτερη μάσκα

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

  - με ksize = -1 χρησιμοποιείται το φίλτρο Scharr 3x3
    - με dx = 1 και dy = 0 χρησιμοποιείται η πρώτη μάσκα
    - με dx = 0 και dy = 1 χρησιμοποιείται η δεύτερη μάσκα

$$\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}$$

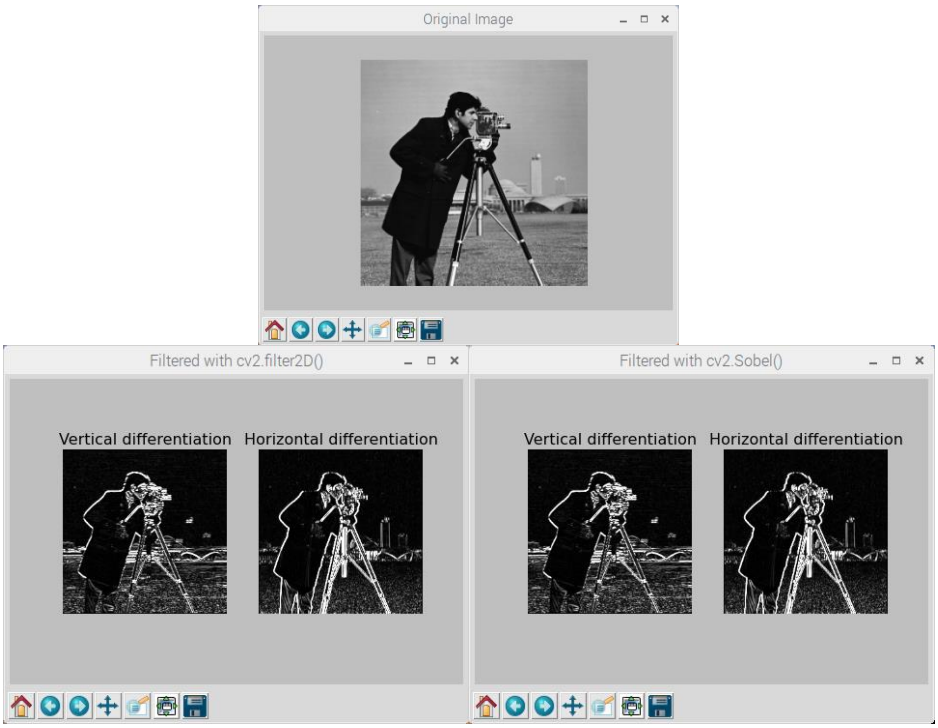
$$\begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

- **scale:** Προαιρετικός συντελεστής κλίμακας για τον υπολογισμό των πρώτων παραγώγων.
- **delta:** Βλέπε συνάρτηση **cv2.filter2D()**.
- **borderType:** Βλέπε συνάρτηση **cv2.filter2D()**.

Η εφαρμογή φίλτρου πρώτης παραγώγου σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/cameraman.tif', 0)
# Sobel filtering with cv2.filter2D()
# vertical differentiation
kernel_vd = np.array(([-1, -2, -1], [0, 0, 0], [1, 2, 1]), np.float32)
# horizontal differentiation
kernel_od = np.array(([-1, 0, 1], [-2, 0, 2], [-1, 0, 1]),np.float32)
# Save the images with dtype greater than uint8 to avoid overflows
```

```
dsty = cv2.filter2D(img, cv2.CV_64F,kernel_vd)
dstx = cv2.filter2D(img, cv2.CV_64F,kernel_od)
# calculate the absolute and convert back to uint8
dsty = cv2.convertScaleAbs(dsty)
dstx = cv2.convertScaleAbs(dstx)
# Sobel filtering with cv2.Sobel()
# Save the images with dtype greater than uint8 to avoid overflows
sobely=cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)# vertical differentiation
sobelx=cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)# horizontal differentiation
# calculate the absolute and convert back to uint8
sobely = cv2.convertScaleAbs(sobely)
sobelx = cv2.convertScaleAbs(sobelx)
```



Εικόνα 4.19: Εφαρμογή φίλτρου πρώτης παραγώγου

4.5.5 Φίλτρο μεσαίας τιμής (Median Filter)

Το φίλτρο μεσαίας τιμής χρησιμοποιείται κυρίως για την απαλοιφή θορύβου τύπου " salt & pepper ".

Η OpenCV διαθέτει τη συνάρτηση **cv2.medianBlur()** για την εφαρμογή φίλτρου μεσαίας τιμής σε μια εικόνα και συντάσσεται ως εξής:

cv2.medianBlur(img, ksize)

Παράμετροι:

- **img:** Εικόνα εισόδου. Με ksize = 3 ή 5, η εικόνα εισόδου μπορεί να έχει αριθμητικό τύπο δεδομένων CV_8U, CV_16U, ή CV_32F, ενώ με ksize > 5 μπορεί να έχει αριθμητικό τύπο δεδομένων μόνο CV_8U.
- **ksize:** Μέγεθος της μάσκας (πχ. για 5x5: ksize = 5).

Η εφαρμογή φίλτρου μεσαίας τιμής σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/cameraman.tif',0)
median = cv2.medianBlur(img,3)
```

Εικόνα 4.20: Εφαρμογή φίλτρου μεσαίας τιμής

4.5.6 Διμερές φίλτρο (Bilateral Filter)

Το διμερές φίλτρο χρησιμοποιείται κυρίως για την απαλοιφή κανονικού θορύβου.

Η OpenCV διαθέτει τη συνάρτηση **cv2.bilateralFilter()** για την εφαρμογή διμερούς φίλτρου σε μια εικόνα και συντάσσεται ως εξής:

cv2.bilateralFilter(img, d, sigmaColor, sigmaSpace, borderType)

Παράμετροι:

- **img:** Εικόνα εισόδου με τύπο δεδομένων CV_8U, CV_32F, ή CV_64F.
- **d:** Διάμετρος της γειτονίας.
- **sigmaColor:** Τιμή φιλτραρίσματος στο χρωματικό χώρο.
- **sigmaSpace:** Τιμή φιλτραρίσματος στο χώρο συχνοτήτων.
- **borderType:** βλέπε συνάρτηση **cv2.filter2D()**.

Η εφαρμογή διμερούς φίλτρου σε μια εικόνα γίνεται με τις εξής εντολές:

```
img = cv2.imread('./Desktop/images/cameraman.tif',0)
bilateral = cv2.bilateralFilter(img,9,75,75)
```

Εικόνα 4.21: Εφαρμογή διμερούς φίλτρου

ΚΕΦΑΛΑΙΟ 5

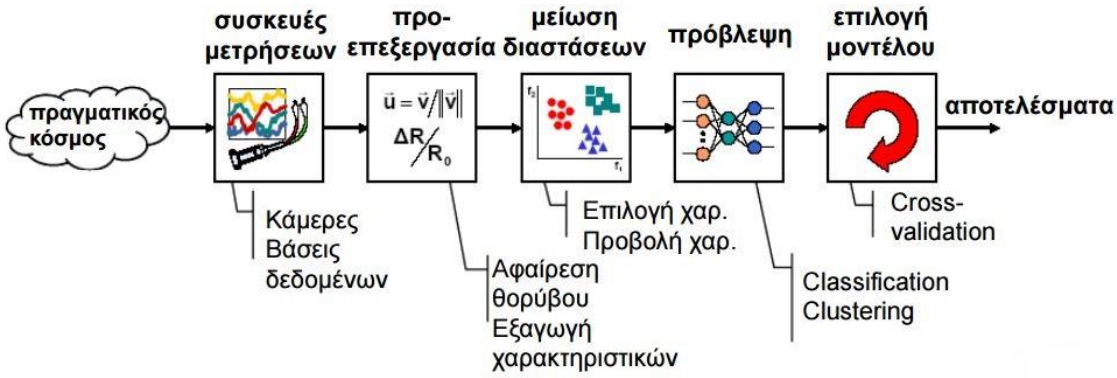
ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

5.1 Εισαγωγή

5.1.1 Τι είναι η αναγνώριση προτύπων

Η Αναγνώριση Προτύπων (Pattern Recognition) είναι ένα πεδίο έρευνας που μελετά τον σχεδιασμό και την υλοποίηση μεθόδων και συστημάτων που αναγνωρίζουν πρότυπα στα δεδομένα και αναλύουν την διαθέσιμη πληροφορία που προέρχεται από κάποιο φυσικό ή τεχνητό φαινόμενο με σκοπό την ερμηνεία του και την απόκτηση γνώσης. **[29]**

Ένα σύστημα ΑΠ περιλαμβάνει: έναν αισθητήρα, μια διαδικασία προ-επεξεργασίας, έναν μηχανισμό εξαγωγής χαρακτηριστικών, έναν αλγόριθμο αναγνώρισης προτύπων και ένα σετ εκπαίδευσης. **[28]**



Σχήμα 5.1: Σύστημα Αναγνώρισης Προτύπων **[28]**

Η ΑΠ βρίσκει εφαρμογή σε πολλές επιστήμες, όπως στην ιατρική (βιοϊατρική τεχνολογία, ανάλυση δεδομένων DNA), σε πεδία της πληροφορικής (υπολογιστική όραση) και σε πεδία της επιστήμης ηλεκτρονικού μηχανικού (ρομποτική). Επίσης χρησιμοποιείται σε συστήματα αναγνώρισης προσώπων (face recognition), σε συστήματα αναγνώρισης χαρακτήρων (optical character recognition – OCR), σε συστήματα αναγνώρισης φωνής (voice recognition), στην υποβοηθούμενη από τον υπολογιστή διάγνωση (computer aided diagnosis) και στην εξόρυξη δεδομένων και ανακάλυψη γνώσης (data mining and knowledge discovery). **[27]**

5.1.2 Αλγόριθμοι αναγνώρισης προτύπων

Οι αλγόριθμοι για την ΑΠ εξαρτώνται από τον τύπο της εξόδου, σχετικά με το εάν η μάθηση είναι επιβλεπόμενη ή όχι και με το εάν ο αλγόριθμος είναι στατιστικός ή μη. Οι στατιστικοί αλγόριθμοι μπορούν να κατηγοριοποιηθούν περαιτέρω σε παραγωγικούς και διακριτικούς. **[27]**

Υπάρχουν αρκετές κατηγορίες αλγορίθμων. Εμείς θα ασχοληθούμε με τους εξής τέσσερις:

- **Αλγόριθμους ταξινόμησης (classification algorithms)**

  Οι αλγόριθμοι ταξινόμησης αντιμετωπίζουν το πρόβλημα κατάταξης νέων αντικειμένων σε ήδη υπάρχουσες υποομάδες ενός δεδομένου συνόλου αντικειμένων (επιβλεπόμενη μάθηση). Στη κατηγορία αυτή βρίσκουμε τους εξής αλγορίθμους: Κ-κοντινότερου γείτονα (K-Nearest Neighbor - KNN), διάνυσμα υποστήριξης μηχανής (Support Vector Machine - SVM), νευρωνικά δίκτυα (Neural Networks) κλπ. **[27]**

- **Αλγόριθμοι ομαδοποίησης (clustering algorithms)**

  Οι αλγόριθμοι ομαδοποίησης αντιμετωπίζουν το πρόβλημα διαχωρισμού ενός υπάρχοντος συνόλου αντικειμένων σε κατηγορίες (clusters) με βάση κριτήρια ομοιότητας (μη επιβλεπόμενη μάθηση). Ο πιο γνωστός αλγόριθμος σε αυτή τη κατηγορία είναι ο αλγόριθμος ομαδοποίησης Κ-μέσων (K-Means clustering). **[27]**

- **Αλγόριθμοι παλινδρόμησης (regression algorithms)**

  Οι αλγόριθμοι παλινδρόμησης προβλέπουν τη τιμή μιας μεταβλητής συνεχών τιμών συναρτήσει άλλων μεταβλητών. Στη κατηγορία αυτή βρίσκουμε τους εξής αλγορίθμους: γραμμική παλινδρόμηση και επεκτάσεις, γκαουσιανή διαδικασία παλινδρόμησης, ανάλυση κύριων συνιστωσών (Principal component analysis - PCA) κλπ. **[27]**

- **Αλγόριθμοι συλλογικής μάθησης**

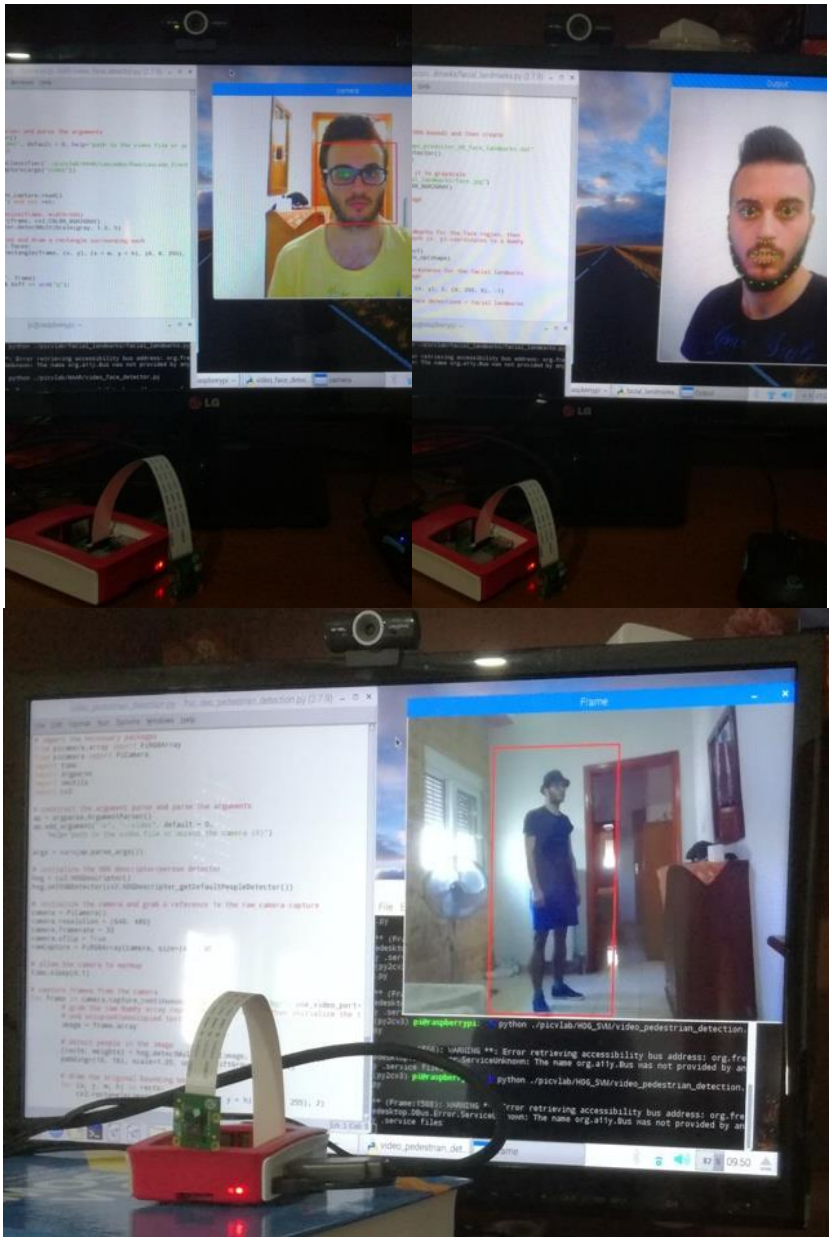
  Οι αλγόριθμοι συλλογικής μάθησης χρησιμοποιούνται στην εκπαίδευση ενός συστήματος αναγνώρισης προτύπων. Στη κατηγορία αυτή βρίσκουμε τους εξής αλγορίθμους: Bootstrap aggregating, Ensemble averaging, Ενίσχυση (Boosting) **[27]**

5.1.3 Ιστορική αναδρομή

Οι πρώτοι που έθεσαν τις βάσεις για την ΑΠ ήταν ο Αριστοτέλης και ο Πλάτωνας, οι οποίοι διαχώρισαν την ουσιώδη ιδιότητα (τι μοιράζονται τα μέλη μιας κατηγορίας μεταξύ τους) από την επουσιώδη ιδιότητα (τις διαφορές ανάμεσα στα μέλη μιας κατηγορίας). Ο Αριστοτέλης κατασκεύασε ένα σύστημα το οποίο ταξινομούσε τα ζώα σε σπονδυλωτά και ασπόνδυλα ενώ παρόμοιο σύστημα για τη ταξινόμηση των φυτών κατασκεύασε ο Θεόφραστος. Τον 18ο αιώνα ο Carolus Linnaeus, με χρήση των νέων γνώσεων που έφερε ο αιώνας των μεγάλων εξερευνήσεων, κατασκεύασε συστηματικές ταξινομήσεις για τη ταξινόμηση των ζωών, των φυτών, των πετρωμάτων και των ασθενειών. Ο Hertzprung και ο Russel ταξινόμησαν τα άστρα σε κατηγορίες με βάση τη θερμοκρασία και τη λαμπρότητα της επιφάνειας τους. Το 1936 έγινε η πρώτη προσπάθεια για τη μαθηματική μορφοποίηση του προβλήματος από τον Fisher. Το ερευνητικό ενδιαφέρον για την ΑΠ ξεκίνησε τη δεκαετία του 1960, η οποία αποτελεί τη πρώτη περίοδο ανάπτυξης της πληροφορικής και της τεχνητής νοημοσύνης. Μετά το 1970 έγιναν προσπάθειες για τη ταχύτερη εξέλιξη του τομέα, ενώ το 1976 ιδρύθηκε η Διεθνής Ένωση Αναγνώρισης Προτύπων (International Association for Pattern Recognition – IAPR). **[14] [27]**

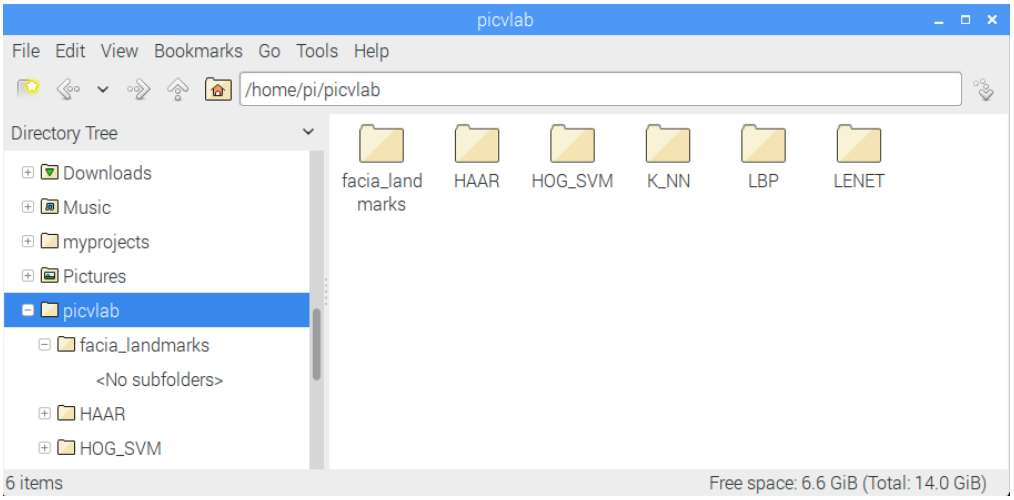
5.2 Αναγνώριση προτύπων με το Raspberry Pi 3 Model B

Στην ενότητα αυτή θα σχεδιάσουμε συστήματα αναγνώρισης προτύπων και θα τα εφαρμόσουμε στο RPi. Σε ορισμένες υλοποιήσεις θα γίνει χρήση κάμερας (RPi Camera και usb webcam). Τα συστήματα αυτά θα υλοποιηθούν με χρήση της γλώσσας Python και των βιβλιοθηκών που αναφέρθηκαν στο Κεφάλαιο 3.



Εικόνα 5.1: Αναγνώριση προτύπων με το RPi

Προκειμένου να έχουμε όλα τα συστήματα συγκεντρωμένα, θα δημιουργήσουμε στο /home/pi ένα κατάλογο με όνομα picvlab, ο οποίος θα έχει τους εξής πέντε καταλόγους: KNN, HAAR, LBP, HOG_SVM, LENET και facial_landmarks



Σχήμα 5.2: Δομή του καταλόγου picvlab

5.2.1 Αναγνώριση προτύπων με την Python και την OpenCV

Στην ενότητα αυτή θα αναπτύξουμε εφαρμογές αναγνώρισης προτύπων με χρήση της Python και της OpenCV. Συγκεκριμένα θα παρουσιάσουμε τους αλγορίθμους K-Κοντινότερου γείτονα και Histogram of Oriented Gradients (HOG) καθώς και τη μέθοδο εντοπισμού αντικειμένων κατά Viola & Jones.

K-Nearest Neighbors (KNN)

Στην αναγνώριση προτύπων, ο αλγόριθμος KNN αποτελεί μια μη παραμετρική μέθοδος η οποία χρησιμοποιείται και για την ταξινόμηση και για την παλινδρόμηση. Και στις δύο περιπτώσεις, η είσοδος αποτελείται από τα *k* κοντινότερα παραδείγματα εκπαίδευσης στο χώρο των χαρακτηριστικών. Η έξοδος εξαρτάται από το εάν ο KNN χρησιμοποιείται για ταξινόμηση ή παλινδρόμηση:

- Στην ταξινόμηση KNN, η έξοδος είναι μέλος της κλάσης. Ένα αντικείμενο ταξινομείται βάση της πλειοψηφίας των γειτόνων του, με το αντικείμενο να κατατάσσεται στην κλάση που είναι πιο κοινή μεταξύ των *k* κοντινότερων γειτόνων του (το k είναι ένας θετικός ακέραιος, συνήθως μικρός). Εάν k = 1, τότε το αντικείμενο απλώς κατατάσσεται στην κλάση του κοντινότερου γείτονα.
- Στην παλινδρόμηση KNN, η έξοδος είναι η τιμή ιδιότητας για το αντικείμενο. Αυτή η τιμή είναι ο μέσος όρος των τιμών των k πλησιέστερων γειτόνων του. **[43]**

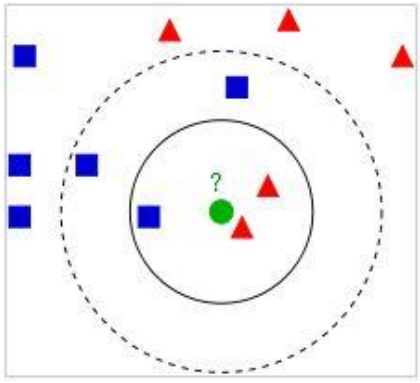
Ο KNN είναι ένας αλγόριθμος μηχανικής μάθησης βασισμένος σε παραδείγματα όπου η λειτουργία προσεγγίζεται μόνο τοπικά και όλος ο υπολογισμός αναβάλλεται μέχρι την ταξινόμηση. Ο αλγόριθμος KNN είναι από τους απλούστερους όλων των αλγορίθμων μηχανικής μάθησης. **[43]**

Τόσο για ταξινόμηση όσο και για παλινδρόμηση, είναι χρήσιμο να αποδοθεί βάρος στις συνεισφορές των γειτόνων, έτσι ώστε οι πλησιέστεροι γείτονες να συνεισφέρουν περισσότερο στο μέσο όρο από ό, τι οι πιο μακρινοί. Για παράδειγμα, μια τιμή βάρους που μπορεί να δοθεί σε κάθε γείτονα ένα ίση με $1 / d$, όπου d είναι η απόσταση από τον γείτονα. **[43]**

Οι γείτονες λαμβάνονται από ένα σύνολο αντικειμένων για τα οποία η κλάση (για την ταξινόμηση KNN) ή η τιμή ιδιότητας αντικειμένου (για παλινδρόμηση KNN) είναι γνωστά. Αυτό μπορεί να θεωρηθεί ως το σετ εκπαίδευσης για τον αλγόριθμο, αν και δεν απαιτείται ρητή εκπαίδευση. **[43]**

Τα παραδείγματα εκπαίδευσης είναι διανύσματα σε ένα πολυδιάστατο χώρο χαρακτηριστικών, όπου καθένα από αυτά διαθέτει μια ετικέτα κλάσης. Η φάση εκπαίδευσης του αλγορίθμου συνίσταται μόνο στην αποθήκευση των διανυσματικών χαρακτηριστικών και των ετικετών κλάσης των δειγμάτων εκπαίδευσης. **[43]**

Στη φάση ταξινόμησης, το k είναι μια σταθερά καθορισμένη από το χρήστη, και ένα μη επισημασμένο διάνυσμα ταξινομείται με την ανάθεση της ετικέτας κλάσης η οποία είναι πιο συχνή μεταξύ των k κοντινότερων δειγμάτων εκπαίδευσης στο εν λόγω σημείο αναζήτησης. **[43]**



Σχήμα 5.3: Αλγόριθμος K-NN **[43]**

Μια κοινώς χρησιμοποιούμενη μονάδα μέτρησης της απόστασης για συνεχείς μεταβλητές είναι η Ευκλείδεια απόσταση. Για διακριτές μεταβλητές, όπως για την ταξινόμηση κειμένου, μπορεί να χρησιμοποιηθεί μια άλλη μονάδα, όπως η απόσταση Hamming. **[43]**

Η OpenCV διαθέτει συναρτήσεις τόσο για την εκπαίδευση όσο και για την εφαρμογή του αλγορίθμου KNN για τη ταξινόμηση ενός νέου δείγματος σε μια κλάση. Αρχικά πρέπει να δημιουργηθεί ένα μοντέλο τύπου KNN ως εξής:

knn = **cv2.ml.KNearest_create()**

Στη συνέχεια, εκπαιδεύουμε το σύστημα μας ως εξής:

knn.**train**(trainData, cv2.ml.ROW_SAMPLE, trainLabels)

Παράμετροι:

- **trainData:** Δείγματα για την εκπαίδευση του συστήματος.
- **trainLabels:** Ετικέτες των κλάσεων
- **cv2.ml.ROW_SAMPLE:** Υποχρεωτική τιμή για την ορθή λειτουργία της συνάρτησης. Σε εκδόσεις της OpenCV < 3.0.0 δεν ήταν υποχρεωτική η χρήση της.

Τέλος, εφαρμόζουμε τον αλγόριθμο για τη ταξινόμηση των εξεταζόμενων δειγμάτων στις κλάσεις ως εξής:

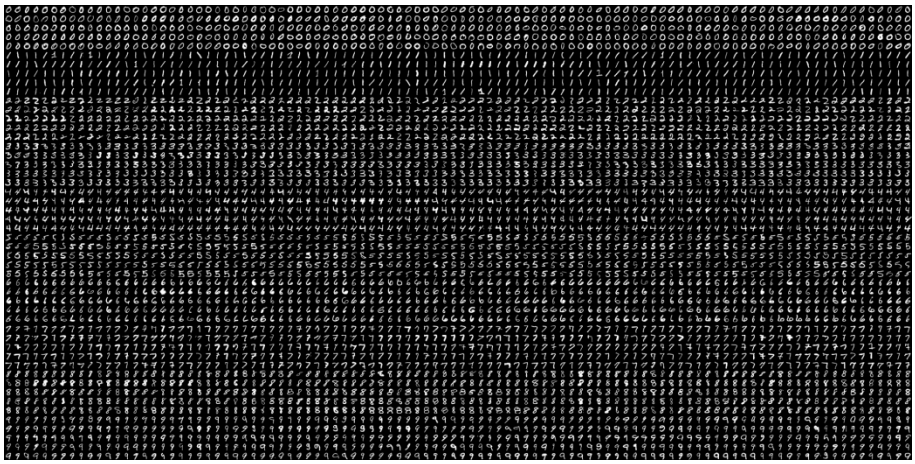
ret, result, neighbors, dist = knn.**findNearest**(testData, k)

Παράμετροι:

- **testData:** Δείγματα προς εξέταση.
- **k:** Αριθμός των κοντινότερων γειτόνων
- **result:** Πίνακας με τα αποτελέσματα της πρόβλεψης για κάθε δείγμα προς εξέταση.
- **neighbors:** Προαιρετικές τιμές εξόδου για τους αντίστοιχους γείτονες.
- **dist:** Προαιρετικές τιμές απόστασης από τα δείγματα προς εξέταση με τους αντίστοιχους γείτονες.

Στο παρακάτω παράδειγμα, θα δημιουργήσουμε ένα σύστημα αναγνώρισης χαρακτήρων (Optical Character Recognition – OCR) το οποίο θα αναγνωρίζει

χειρόγραφα ψηφία με τον αλγόριθμο KNN. Αρχικά διαβάζουμε την εικόνα digits.png, η οποία έχει 5000 χειρόγραφα ψηφία (500 για κάθε ψηφίο). Κάθε ψηφίο είναι μια εικόνα 20x20. Στη συνέχεια χωρίζουμε αυτήν την εικόνα σε 5000 διαφορετικά ψηφία. Για κάθε ψηφίο, το χωρίζουμε σε μια μόνο σειρά με 400 εικονοστοιχεία. Τέλος, χρησιμοποιούμε τα πρώτα 250 δείγματα από κάθε ψηφίο ως δείγματα για την εκπαίδευση του συστήματος και τα επόμενα 250 δείγματα ως δείγματα προς εξέταση.

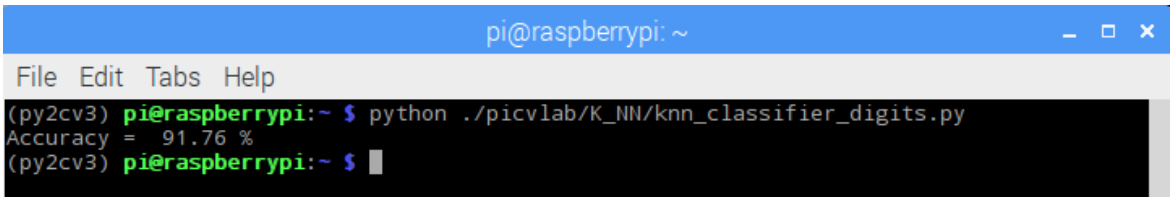


Σχήμα 5.4: Εικόνα digits.png

Δημιουργούμε στο /home/pi/picvlab/KNN ένα αρχείο knn_classifier_digits.py και γράφουμε το παρακάτω κώδικα. Τοποθετούμε την εικόνα digits.png στο κατάλογο αυτό και εκτελούμε το αρχείο με την εντολή:
python ./picvlab/KNN/ knn_classifier_digits.py

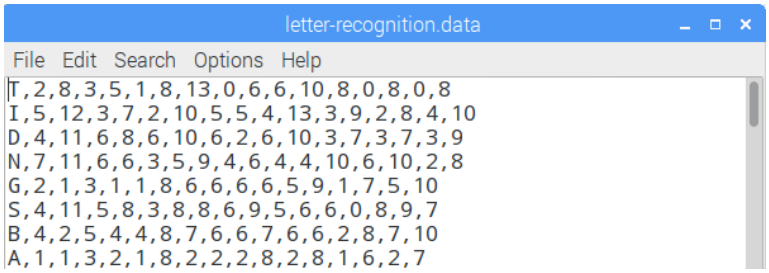
```
import numpy as np
import cv2

img = cv2.imread('./picvlab/KNN/digits.png')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

# Now we split the image to 5000 cells, each 20x20 size
cells = [np.hsplit(row,100) for row in np.vsplit(gray,50)]

# Make it into a Numpy array. It size will be (50,100,20,20)
x = np.array(cells)
```

```
# Now we prepare trainData and testData.
# trainData size = (2500,400)
trainData = x[:,:50].reshape(-1,400).astype(np.float32)
# testData size = (2500,400)
testData = x[:,50:100].reshape(-1,400).astype(np.float32)

# Create labels for train and test data
k = np.arange(10)
trainLabels = np.repeat(k,250)[:,np.newaxis]
testLabels = trainLabels.copy()

# Initiate kNN, train the data, then test it with test data for k=5
knn = cv2.ml.KNearest_create()
knn.train(trainData,cv2.ml.ROW_SAMPLE,trainLabels)
ret, result, neighbours, dist = knn.findNearest(testData, k=5)

# Now we check the accuracy of classification
# For that, compare the result with testLabels and check
# which are wrong
matches = result==testLabels
correct = np.count_nonzero(matches)
accuracy = correct*100.0/result.size
print "Accuracy = ", accuracy,"%"
```



Εικόνα 5.2: Αποτέλεσμα του προγράμματος knn_classifier_digits.py

Το παραπάνω πρόγραμμα μας έδωσε ακρίβεια σχεδόν 92%. Ας κάνουμε τώρα το ίδιο για την αναγνώριση λατινικών χαρακτήρων. Εδώ, αντί για εικόνα, θα διαβάσουμε το αρχείο letter-recognition.data. Το αρχείο αυτό περιέχει 20000 δείγματα επομένως τα μισά θα χρησιμοποιηθούν ως δείγματα για την εκπαίδευση του συστήματος και υπόλοιπα ως δείγματα προς εξέταση. Να σημειωθεί ότι πρέπει να μετατρέψουμε τους λατινικούς χαρακτήρες σε χαρακτήρες ascii.



Σχήμα 5.5: Τμήμα του αρχείου letter-recognition.data

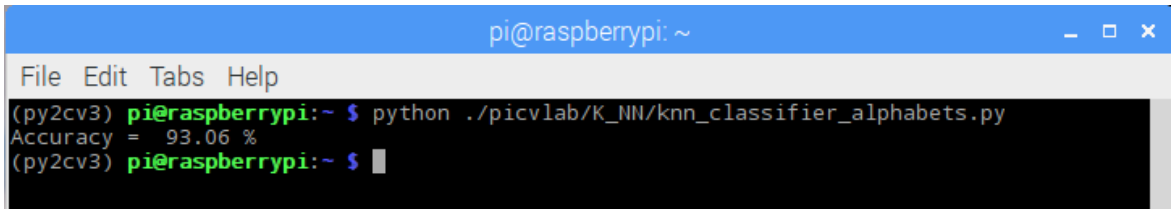
Σε κάθε γραμμή, η πρώτη στήλη είναι ένα γράμμα του αλφαβήτου, το οποίο αποτελεί την ετικέτα και οι υπόλοιπες 16 τιμές αποτελούν τα διαφορετικά χαρακτηριστικά του. Τα χαρακτηριστικά αυτά προέρχονται από το UCI Mchine Learning Repository.

Δημιουργούμε στο /home/pi/picvlab/KNN ένα αρχείο knn_classifier_alphabets.py και γράφουμε το παρακάτω κώδικα. Τοποθετούμε το αρχείο letter-recognition.data στο κατάλογο αυτό και εκτελούμε το αρχείο με την εντολή:

python ./picvlab/KNN/ knn_classifier_alphabets.py

```
import numpy as np
import cv2

# Load the data, converters convert the letter to a number
data= np.loadtxt('./Desktop/letter-recognition.data',
dtype= 'float32', delimiter = ',',
converters= {0: lambda ch: ord(ch)-ord('A')})

# split the data to two, 10000 each for train and test
train, test = np.vsplit(data,2)

# split trainData and testData to features and responses
trainLabels, trainData = np.hsplit(train,[1])
testLabels, testData = np.hsplit(test,[1])

# Initiate the kNN, classify and measure accuracy.
knn = cv2.ml.KNearest_create()
knn.train(trainData,cv2.ml.ROW_SAMPLE,trainLabels)
ret, result, neighbors, dist = knn.findNearest(testData, k=5)
```

```
correct = np.count_nonzero(result == testLabels)
accuracy = correct*100.0/10000
print "Accuracy = ", accuracy,"%"
```



Εικόνα 5.3: Αποτέλεσμα του προγράμματος knn_classifier_alphabets.py

Το παραπάνω πρόγραμμα μας έδωσε ακρίβεια 93%.

Εντοπισμός αντικειμένων κατά Viola & Jones

Το 2001, οι Paul Viola και Michael Jones παρουσίασαν στο έγγραφο Rapid Object Detection using a Boosted Cascade of Simple Features μια μέθοδο εντοπισμού αντικειμένων. Παρόλο που μπορεί να εκπαιδευτεί για την ανίχνευση διαφόρων κατηγοριών αντικειμένων, ήταν υποκινούμενο κυρίως από το πρόβλημα της ανίχνευσης προσώπου. Η μέθοδος αυτή, βασίζεται στην εκμάθηση μηχανής, όπου το σύστημα διδάσκεται με ένα σύνολο θετικών εικόνων, δηλαδή εικόνων που περιέχουν το αντικείμενο προς εντοπισμό, και αρνητικών εικόνων, δηλαδή εικόνων που δεν περιέχουν το αντικείμενο προς εντοπισμό. **[44]**

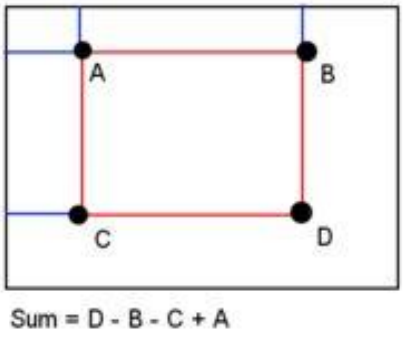
Η μέθοδος χωρίζεται στις εξής τέσσερις φάσεις:

1. **Δημιουργία της ολοκληρωτικής εικόνας**

   Στόχος της μεθόδου που παρουσίασαν οι Viola και Jones είναι μια γρήγορη και ακριβής ανίχνευση αντικείμενων. Για το λόγο αυτό χρησιμοποίησαν την ολοκληρωτική εικόνα (integral image), η οποία αυξάνει τη ταχύτητα της ανίχνευσης και διευκολύνει τις υπολογιστικές διαδικασίες, και ορίζεται ως εξής:

$$I_{int}(x,y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} I(x',y')$$

   δηλαδή, η τιμή της ολοκληρωτικής εικόνας στο σημείο (x, y) ισούται με το άθροισμα των τιμών των σημείων πάνω και αριστερά από το σημείο αυτό, συμπεριλαμβανομένου του σημείου αυτού. **[44]**

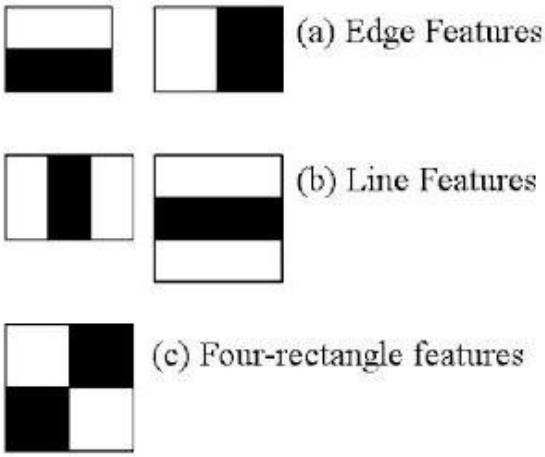


Σχήμα 5.6: Ολοκληρωτική εικόνα

2. Επιλογή του συνόλου χαρακτηριστικών

Η σάρωση όλων των εικονοστοιχείων μιας εικόνας για την ανίχνευση ενός αντικειμένου αποτελεί μια αργή διαδικασία η οποία απαιτεί μεγάλη υπολογιστική ισχύ. Για το λόγο αυτό, προτιμάται η χρήση ενός συνόλου χαρακτηριστικών (feature set) των τιμών των εικονοστοιχείων. Με την χρήση των χαρακτηριστικών αυτών, η εικόνα χωρίζεται σε ορθογώνιες περιοχές και υπολογίζεται το άθροισμα των εικονοστοιχείων των τετραγώνων μέσω της ολοκληρωτικής εικόνας.

Τα χαρακτηριστικά που πρότειναν οι Viola και Jones ονομάζονται χαρακτηριστικά Haar και είναι κάποια ορθογώνια σχήματα, των οποίων οι τιμές των ισούται με τη διαφορά του αθροίσματος των εικονοστοιχείων του λευκού τμήματος από το άθροισμα των εικονοστοιχείων του μαύρου τμήματος. **[44]**



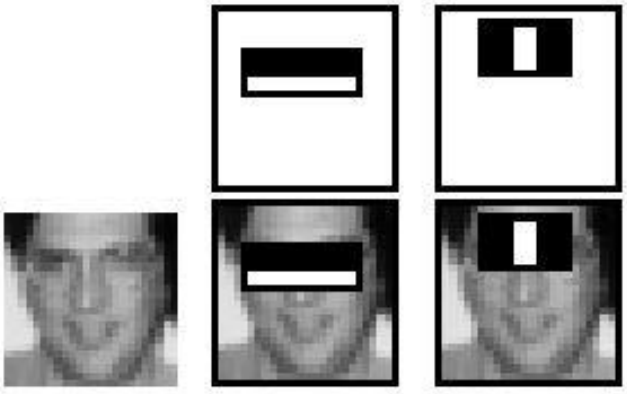
Σχήμα 5.7: Χαρακτηριστικά Haar

3. Εκπαίδευση του συστήματος

Ο αλγόριθμος που χρησιμοποιείται για την εκπαίδευση του συστήματος είναι ο αλγόριθμος AdaBoost, ο οποίος χρησιμοποιείται τόσο για την επιλογή των καλύτερων χαρακτηριστικών όσο και για την εκπαίδευση των ταξινομητών που τα χρησιμοποιούν. Ο αλγόριθμος αυτός κατασκευάζει έναν δυνατό ταξινομητή ο οποίος προκύπτει από το άθροισμα ειδικών βαρών των απλών αδύναμων ταξινομητών. Ένας απλός ταξινομητής ορίζεται ως εξής:

$$h_i(x, f_i, p_i, \theta_i) = \begin{cases} 1, \alpha\nu\ p_i f_i(x) < p_i \theta_i \\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

όπου $f_i(x)$ είναι το διάνυσμα ενός χαρακτηριστικού υπολογισμένο στο παράθυρο x, p_i είναι η πιθανότητα και θ_i είναι το κατώφλι. Στα παρακάτω σχήματα βλέπουμε ένα παράδειγμα χαρακτηριστικών που έχουν επιλεχθεί από τον αλγόριθμο AdaBoost καθώς και τα βήματα του. **[44]**



Σχήμα 5.8: Χαρακτηριστικά που έχουν επιλεχθεί από τον αλγόριθμο AdaBoost **[45]**

Input: Εικόνες εκπαίδευσης $(x_1, \ldots, x_n)$ και οι αντίστοιχες ετικέτες $(y_1, \ldots, y_n)$, όπου $y_i \in \{0, 1\}$ για αρνητικά και θετικά δείγματα αντίστοιχα. Ο αριθμός των αρνητικών δειγμάτων είναι m και αυτός των θετικών δειγμάτων $l = n - m$

1: Αρχικοποίηση των n βαρών ως $(2m)^{-1}$ για $y_i = 0$ και $(2l)^{-1}$ για $y_i = 1$

2: **for** $t = 1, \ldots, T$ **do**

3: Κανονικοποίηση των βαρών $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

4: Επιλογή του καλύτερου αδύναμου ταξινομητή ως προς το σφάλμα με βάρη $\epsilon_t = \min_{f,p,\theta} \sum_i w_i \left| h(x_i, f, p, \theta) - y_i \right|$

5: Ορισμός της $h_t(x) = h(x, f_t, p_t, \theta_t)$, όπου f_t, p_t και θ_t οι τιμές των μεταβλητών που ελαχιστοποιούν το ϵ_t

6: Ανανέωση των βαρών $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ όπου $e_i = 0$ αν το δείγμα x_i ταξινομείται σωστά, $e_i = 1$ αν το δείγμα x_i ταξινομείται λανθασμένα και $\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$

7: **end for**

8: Ο δυνατός ταξινομητής με $\alpha_t = \log \frac{1}{\beta_t}$ ορίζεται ως

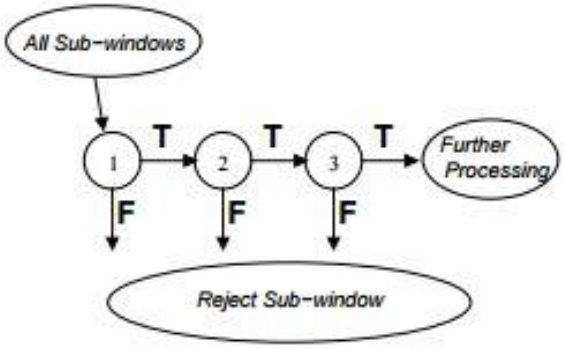
$$h_x = \begin{cases} 1, & \text{αν } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, & \text{αλλιώς} \end{cases}$$

Σχήμα 5.9: Λειτουργία του αλγόριθμου AdaBoost **[15]**

4. Εφαρμογή του ταξινομητή για τον εντοπισμό των αντικειμένων

Η μέθοδος εντοπισμού αντικειμένου κατά Viola και Jones χρησιμοποιεί μια διαφορετική λογική όσον αφορά των εντοπισμό ενός αντικειμένου. Σύμφωνα με αυτή, αντί η μέθοδος να εντοπίζεται το αντικείμενο, απορρίπτεται το μη αντικείμενο, καθώς η διαδικασία απόρριψης μη αντικειμένων είναι ταχύτερη από τη διαδικασία εύρεσης των. Για το λόγο αυτό, οι Viola και Jones εισήγαγαν την έννοια του ταξινομητή Cascade.

Ο ταξινομητής αυτός, αποτελείται από μια σειρά δυνατών ταξινομητών. Κάθε δυνατός ταξινομητής καθορίζει κατά πόσο ένα παράθυρο εισόδου είναι σίγουρα μη αντικείμενο ή πιθανό αντικείμενο. Αν κάποιος δυνατός ταξινομητής το ταξινομήσει ως μη αντικείμενο, τότε απορρίπτεται, αλλιώς συνεχίζει στον επόμενο δυνατό ταξινομητή. Όσους περισσότερους δυνατούς ταξινομητές καταφέρει να περάσει χωρίς να απορριφθεί, τόσες περισσότερες πιθανότητες έχει να περιλαμβάνει το αντικείμενο. **[44]**



Σχήμα 5.10: Ταξινομητής Cascade **[45]**

Input: Βαθμός false-positive f, βαθμός ανίχνευσης d και στόχος βαθμού false-positive F_{target}. Σύνολο P θετικών δειγμάτων (προσώπων) και σύνολο N αρνητικών δειγμάτων (μη-προσώπων)

Αρχικοποίηση $F_0 = 1$, $D_0 = 1$, $i = 0$
while $F_i > F_{target}$ και $n_i < N$ **do**
  $i = i + 1$, $n_i = 0$ και $F_i = F_{i-1}$
  while $F_i > F_{i-1}$ **do**
    $n_i = n_i + 1$
    Εκτέλεση αλγορίθμου AdaBoost με P, N και τα n_i χαρακτηριστικά
    Υπολογισμός του τρέχοντος ταξινομητή για το σύνολο που προέκυψε ώστε να καθοριστούν τα F_i και D_i
    Μείωση της τιμής κατωφλίου του $i-$οστού ταξινομητή ώστε ο βαθμός ανίχνευσης να γίνει τουλάχιστον $d \cdot D_{i-1}$
  end while
  $N = 0$
  if $F_i > F_{target}$ **then**
    Υπολογισμός του τρέχοντος ανιχνευτή cascade για τις εικόνες μη-προσώπου και τοποθέτηση τυχόντων λανθασμένων ανιχνεύσεων στο σύνολο N
  end if
end while

Σχήμα 5.11: Λειτουργία του ταξινομητή Cascade **[15]**

Η OpenCV διαθέτει προ-εγκατεστημένους Cascade ταξινομητές για τον εντοπισμό προσώπων και χαρακτηριστικών των σε μορφή XML αρχείων. Τα αρχεία αυτά βρίσκονται στο κατάλογο /home/pi/opencv-3.1.0/date/haarcascades. Η ανάγνωση των αρχείων αυτών γίνεται με τη συνάρτηση **cv2.CascadeClassifier()**. Ο εντοπισμός των αντικειμένων σε μια εικόνα γίνεται με τη συνάρτηση **cv2.CascadeClassifier.detectMultiScale()**, όπου **cv2.CascadeClassifier()** ο Cascade ταξινομητές που έχει αναγνωσθεί προηγουμένως, και συντάσσεται ως εξής:

cv2.CascadeClassifier.detectMultiScale(img, scaleFactor, minNeigbors, minSize, maxSize)

Παράμετροι:

- **img:** Εικόνα στην οποία θα εντοπιστούν τα αντικείμενα με τύπο δεδομένων CV_8U.
- **scaleFactor:** Παράμετρος που καθορίζει πόσο μειώνεται το μέγεθος της εικόνας σε κάθε κλίμακα εικόνας.
- **minNeigbors:** Η παράμετρος που καθορίζει τον αριθμό των γειτόνων που κάθε υποψήφιο ορθογώνιο πρέπει να διατηρήσει.
- **minSize:** Ελάχιστο δυνατό μέγεθος αντικειμένου που θα εντοπίζεται.
- **maxSize:** Μέγιστο δυνατό μέγεθος αντικειμένου που θα εντοπίζεται.

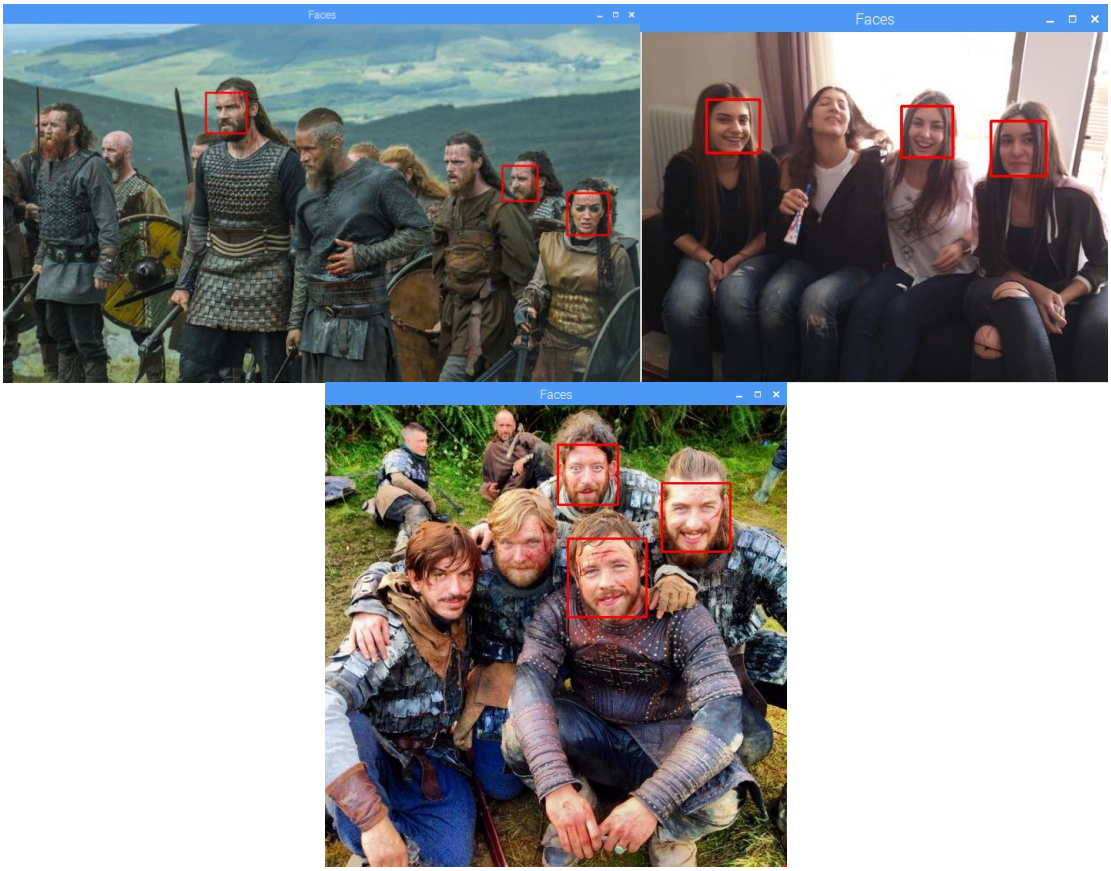
Τα παρακάτω προγράμματα αποτελούν τρία συστήματα ανίχνευσης προσώπων (face_recognition.py, video_face_recognition.py και video_face_recognition2.py. Τα προγράμματα αυτά βρίσκονται στο /home/pi/picvlab/HAAR). Στη πρώτη υλοποίηση εφαρμόζουμε ανίχνευση προσώπων σε φωτογραφίες που είτε διαβάζονται από το δίσκο είτε λαμβάνονται από την RPi Camera και εφαρμόζονται σε αυτές χωρίς να έχουν αποθηκευτεί στο δίσκο. Στη δεύτερη υλοποίηση εφαρμόζουμε ανίχνευση προσώπων είτε σε βίντεο που διαβάζονται από τον δίσκο είτε σε πραγματικό χρόνο με χρήση webcam. Στη τρίτη υλοποίηση εφαρμόζουμε ανίχνευση προσώπων σε πραγματικό χρόνο με χρήση της RPi Camera.

Αρχικά, εφαρμόζουμε αναγνώριση προτύπων σε τρείς εικόνες που βρίσκονται στο δίσκο. Τρέχουμε το πρόγραμμα τρείς φορές με την εξής εντολή:
python ./picvlab/HAAR/face_recognition.py --image ./picvlab/HAAR/images/faces/img
όπου img οι εικόνες face-1.png, face-2.png και face-3.png

Να τονίσουμε σε αυτό το σημείο ότι, όπως φαίνεται και στο πρόγραμμα, η παράμετρος scaleFactor έχει τιμή 1.3 και η παράμετρος minNeighbors έχει τιμή 5.

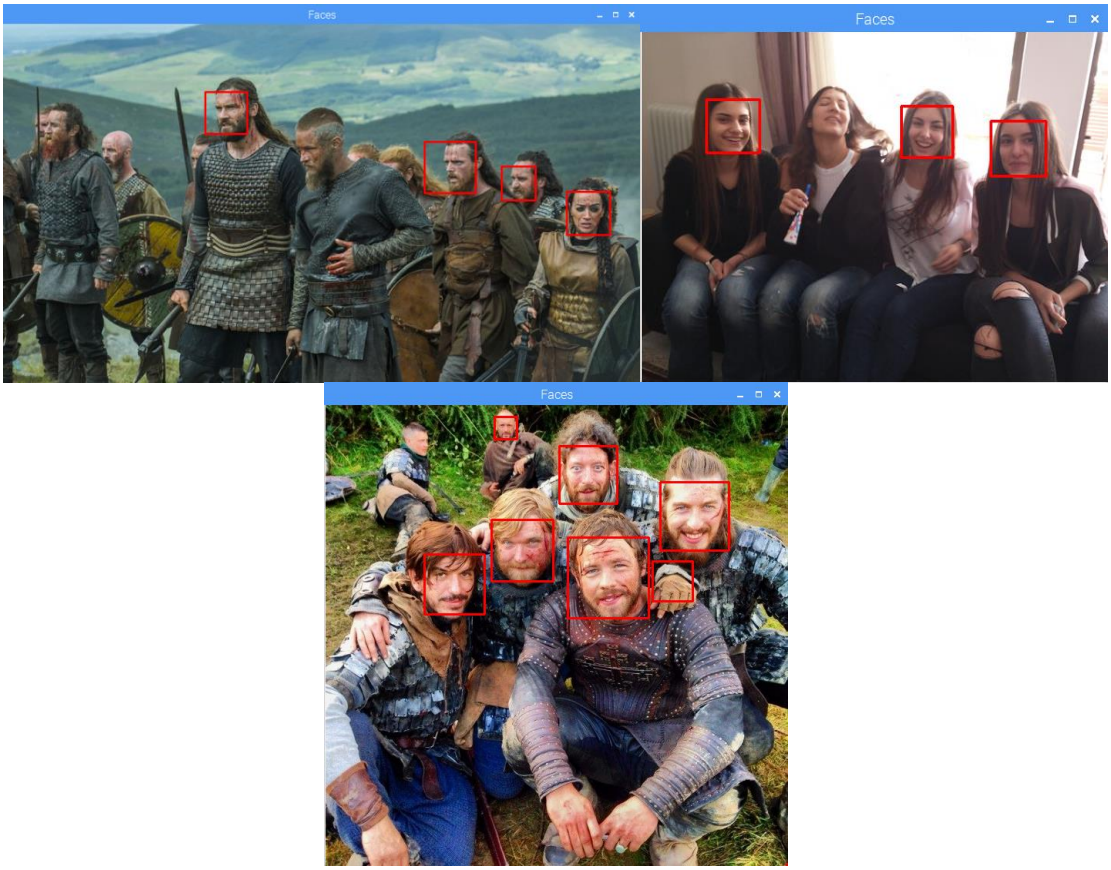
```
# import the necessary packages
import numpy as np
import argparse
import cv2
```

```
import time
from picamera import PiCamera
from picamera.array import PiRGBArray

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", default="None",
    help="path to the input image or access the camera (None value)")
ap.add_argument("-c", "--cascade",

default="./picvlab/HAAR/cascades/haarcascade_frontalface_default.xml",
    help="path to face detector haar cascade")
args = vars(ap.parse_args())

if args["image"] != "None":
    # load the input image and convert it to grayscale
    image = cv2.imread(args["image"])
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

else:
    # Take a capture from the RPi Camera
    camera = PiCamera()
    camera.resolution = (640,480)
    rawCapture = PiRGBArray(camera, size= (640,480))
    time.sleep(0.1)
    camera.capture(rawCapture, format = "bgr")
    image = rawCapture.array
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# load the face detector Haar cascade, then detect faces
# in the input image
detector = cv2.CascadeClassifier(args["cascade"])
rects = detector.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)

# loop over the faces and draw a rectangle surrounding each
for (i, (x, y, w, h)) in enumerate(rects):
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

# show the detected faces
cv2.imshow("Faces", image)
cv2.waitKey(0)
```



Εικόνα 5.4: Ανίχνευση προσώπων με scaleFactor = 1.3, minNeighbors = 5

Ας δοκιμάσουμε να τρέξουμε ξανά το πρόγραμμα, αυτή τη φορά αλλάζοντας τη τιμή του scaleFactor σε 1.1

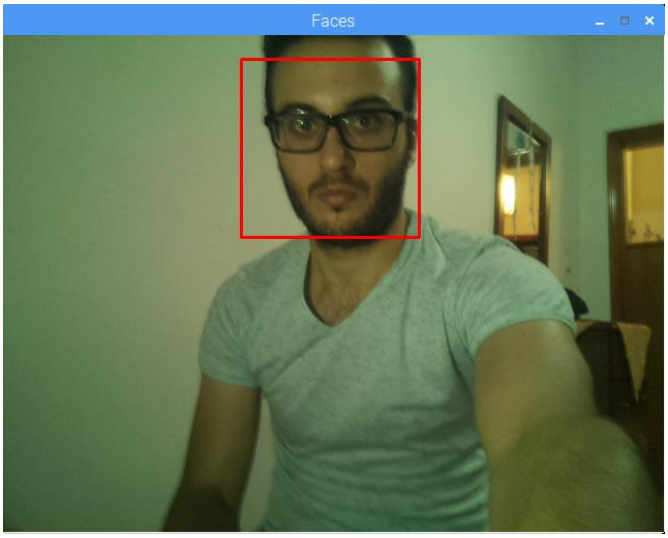


Εικόνα 5.5: Ανίχνευση προσώπων με scaleFactor = 1.1, minNeighbors = 5

Παρατηρούμε ότι στην εικόνα face-1.png ανιχνεύσαμε ένα πρόσωπο παραπάνω και στην εικόνα face-3.png τρία επιπλέον πρόσωπα συν μια λάθος ανίχνευση. Συμπεραίνουμε λοιπόν πώς για να πετύχουμε το βέλτιστο δυνατό αποτέλεσμα, θα πρέπει να τροποποιούμε κατάλληλα τις τιμές των παραμέτρων scaleFactor και minNeighbors ανάλογα με την εικόνα. Οι περισσότερο χρησιμοποιούμενες τιμές είναι οι 1.3 και 5 αντίστοιχα, αλλά δε δίνουν πάντα το βέλτιστο δυνατό αποτέλεσμα.

Στη συνέχεια εφαρμόζουμε αναγνώριση προτύπων σε φωτογραφία που θα ληφθεί με τη RPi Camera. Τρέχουμε το πρόγραμμα με την εξής εντολή:

```
python ./picvlab/HAAR/face_recognition.py --image None
```

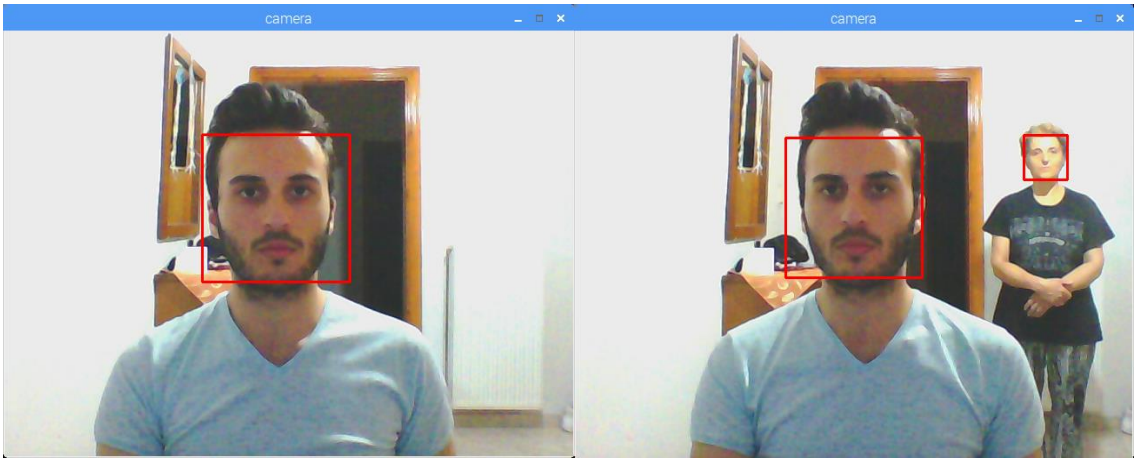


Εικόνα 5.6: Ανίχνευση προσώπου σε φωτογραφία που λήφθηκε με χρήση της RPi Camera

Τώρα, θα εφαρμόσουμε ανίχνευση προσώπων σε πραγματικό χρόνο με χρήση usb webcam. Τρέχουμε το πρόγραμμα με την εξής εντολή:
python ./picvlab/HAAR/video_face_recognition.py

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

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", default = 0, help="path to the video file or access the camera (0)")
args = vars(ap.parse_args())

face_detector=
cv2.CascadeClassifier('./picvlab/HAAR/cascades/haarcascade_frontalface_default.xml')
video_capture = cv2.VideoCapture(args["video"])


while (True):
    (ret, frame) = video_capture.read()
    if args.get("video") and not ret:
```

```
        break
    #frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)

    # loop over the faces and draw a rectangle surrounding each
    for (x, y, w, h) in faces:
        img = cv2.rectangle(frame,(x, y),(x + w, y + h),(0, 0, 255),2)

    # show the frames
    cv2.imshow("Camera", frame)
    if cv2.waitKey(1) & 0xff == ord("q"):
        break

video_capture.release(); cv2.destroyAllWindows()
```

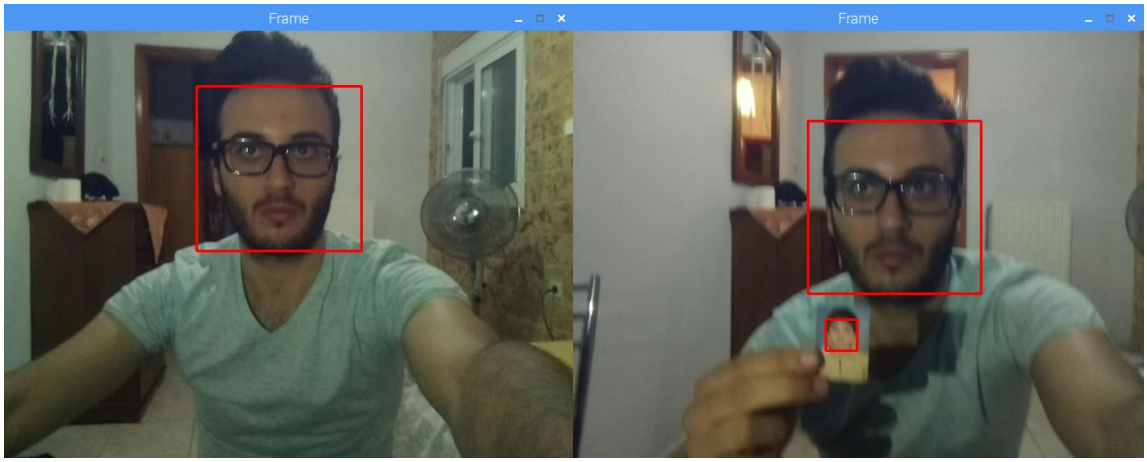


Εικόνα 5.7: Ανίχνευση προσώπων σε πραγματικό χρόνο με χρήση usb webcam

Τέλος, θα εφαρμόσουμε ανίχνευση προσώπων σε πραγματικό χρόνο με χρήση της RPi Camera. Τρέχουμε το πρόγραμμα με την εξής εντολή:
python ./picvlab/HAAR/video_face_recognition2.py

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import numpy as np
import cv2
```

```
face_detector=cv2.CascadeClassifier('./picvlab/HAAR/cascades/haarcascade
_frontalface_default.xml')

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))

# allow the camera to warmup
time.sleep(0.1)

# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    # grab the raw NumPy array representing the image, then initialize
    the timestamp
    # and occupied/unoccupied text
    image = frame.array
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
    # loop over the faces and draw a rectangle surrounding each
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # show the frame
    cv2.imshow("Camera", image)
    key = cv2.waitKey(1) & 0xFF

    # clear the stream in preparation for the next frame
    rawCapture.truncate(0)
     # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break
```



Εικόνα 5.8: Ανίχνευση προσώπων σε πραγματικό χρόνο με χρήση της RPi Camera

Histogram of Oriented Gradients (HOG)

Ο HOG είναι ένας περιγραφέας χαρακτηριστικών ο οποίος χρησιμοποιείται στην μηχανική όραση και την επεξεργασία εικόνας με στόχο την ανίχνευση αντικειμένων. Η τεχνική μετρά περιστατικά προσανατολισμού κλίσης σε τοπικά τμήματα μιας εικόνας. **[46]**

Ο πρώτος που περιέγραψε τις έννοιες πίσω από το HOG ήταν ο Robert K. McConnell, χωρίς να χρησιμοποιήσει τον όρο HOG, σε μια πατέντα ευρεσιτεχνίας το 1986. Το 1994 οι έννοιες αυτές χρησιμοποιήθηκαν από τα Mitsubishi Electric Research Laboratories. Ωστόσο, η χρήση τους έγινε ευρέως διαδεδομένη το 2005, όταν ο Navneet Dalal και ο Bill Triggs, ερευνητές του Γαλλικού Εθνικού Ινστιτούτου Έρευνας Πληροφορικής και Αυτοματισμού (INRIA), παρουσίασαν τη συμπληρωματική τους εργασία σχετικά με τον HOG στο συνέδριο Υπολογιστικής Όρασης και Αναγνώρισης Προτύπων (CVPR). Σε αυτή την εργασία επικεντρώθηκαν στην ανίχνευση πεζών σε στατικές εικόνες, παρόλο που έκτοτε επέκτειναν τις δοκιμές τους ώστε να συμπεριλάβουν την ανίχνευση ανθρώπων σε βίντεο, καθώς και σε μια ποικιλία κοινών ζώων και οχημάτων σε στατικές εικόνες. **[46]**

Η βασική σκέψη πίσω από το HOG είναι ότι η εμφάνιση ενός τοπικού αντικειμένου και του σχήματος του μέσα σε μια εικόνα μπορούν να περιγραφούν από τη κατανομή βαθμίδων έντασης ή ακμών. Η εικόνα χωρίζεται σε μικρές συνδεδεμένες περιοχές που ονομάζονται κελιά, και για τα εικονοστοιχεία μέσα σε κάθε κελί, καταρτίζεται το ιστόγραμμα των προσανατολισμένων διαβαθμίσεων. Ο περιγραφέας είναι η αλληλουχία αυτών των ιστογραμμάτων. Για βέλτιστη

ακρίβεια, τα τοπικά ιστογράμματα μπορούν να κανονικοποιηθούν, υπολογίζοντας ένα μέτρο της έντασης σε μια μεγαλύτερη περιοχή της εικόνας, που ονομάζεται μπλοκ, και στη συνέχεια χρησιμοποιώντας αυτήν την τιμή για να κανονικοποιηθούν όλα τα κελιά εντός του μπλοκ. Αυτή η κανονικοποίηση έχει ως αποτέλεσμα την καλύτερη ανανέωση των αλλαγών στον φωτισμό και τη σκίαση. **[46]**

Ο αλγόριθμος HOG αποτελείται από τα εξής βήματα:

1. **Υπολογισμός των τιμών κλίσης**

   Το πρώτο βήμα είναι ο υπολογισμός των τιμών κλίσης. Η πιο συνηθισμένη μέθοδος είναι η εφαρμογή μιας διακριτής μάσκας σε έναν ή και στους δυο άξονες (οριζόντιο και κάθετο άξονα). Συγκεκριμένα, αυτή η μέθοδος απαιτεί το φιλτράρισμα της εικόνας με τα εξής φίλτρα:

   $$[-1,0,1] \; \kappa\alpha\iota \; [-1,0,1]^T$$ **[46]**

2. **Συγκέντρωση προσανατολισμού**

   Το δεύτερο βήμα είναι η δημιουργία των ιστογραμμάτων για κάθε κελί. Κάθε εικονοστοιχείο εντός του κελιού εκπέμπει μια σταθμισμένη ψήφο για ένα ιστόγραμμα βάσει των τιμών που βρέθηκαν στον υπολογισμό κλίσης. Τα ίδια τα κελιά μπορούν να είναι είτε ορθογώνια είτε ακτινωτά, και τα ιστογράμματα κατανέμονται ομοιόμορφα σε 0 έως 180^o ή 0 έως 360^o, ανάλογα με το αν η κλίση είναι "μη υπογεγραμμένη" ή "υπογεγραμμένη". **[46]**

3. **Δημιουργία μπλοκ περιγραφών**

   Για να ληφθούν υπόψη οι αλλαγές στον φωτισμό και την αντίθεση, οι δυνάμεις της κλίσης πρέπει να ομαλοποιηθούν τοπικά, πράγμα που απαιτεί την ομαδοποίηση των κελιών σε μεγαλύτερα, χωρικά συνδεδεμένα μπλοκ. Αυτά τα μπλοκ τυπικά αλληλεπικαλύπτονται, πράγμα που σημαίνει ότι κάθε κύτταρο συνεισφέρει περισσότερο από μία φορά στον τελικό περιγραφέα. Υπάρχουν δύο είδη μπλοκ: τα ορθογώνια μπλοκ (R-HOG) και κυκλικά μπλοκ (C-HOG). **[46]**

4. **Κανονικοποίηση των μπλοκ**

   Οι Dalal και Triggs διερεύνησαν τέσσερις διαφορετικές μεθόδους για την ομαλοποίηση των μπλοκ. Έστω u το μη κανονικοποιημένο διάνυσμα που περιέχει όλα τα ιστογράμματα σε ένα δεδομένο μπλοκ, $\|u\|_k$ το

κανονικοποιημένο διάνυσμα για $k = 1, 2\ και\ e$. Ο συντελεστής κανονικοποίησης μπορεί να είναι ένας από τους ακόλουθους:

L2-norm: $f = \frac{u}{\sqrt{\|u\|_2^2 + e^2}}$

L2-hys: αποτελεί τον L2-norm ακολουθούμενο από το περιορισμό των μέγιστων τιμών του u σε 0.2 και την εφαρμογή κανονικοποίησης για δεύτερη φορά.

L1-norm: $f = \frac{u}{\sqrt{\|u\|_1 + e}}$

L1-sqrt: $f = \sqrt{\frac{u}{\sqrt{\|u\|_1 + e}}}$ **[46]**

5. **Τροφοδοσία του περιγραφέα σε ένα σύστημα εντοπισμού**

   Το τελευταίο βήμα είναι να τροφοδοτήσουμε τον περιγραφέα σε κάποιο σύστημα εντοπισμού. Συνήθως το σύστημα που επιλέγεται είναι το SVM. Φυσικά, αντί του SVM μπορεί να επιλεγεί κάποιο νευρωνικό δίκτυο. **[46]**

Η OpenCV διαθέτει συναρτήσεις για την υλοποίηση του αλγορίθμου HOG. Αρχικά θα πρέπει να δημιουργήσουμε ένα περιγραφέα HOG ως εξής:

hog = **cv2.HOGDescriptor()**

Στη συνέχεια τροφοδοτούμε το περιγραφέα μαζί με τον ταξινομητή στο SVM (η OpenCV διαθέτει προ-εκπαιδευμένους ταξινομητές για την ανίχνευση πεζών (pedestrian detection)) ως εξής:

hog.**setSVMDetector**(cv2.HOGDescriptor_getDefaultPeopleDetector())

όπου **cv2.HOGDescriptor_getDefaultPeopleDetector()** ο προ-εκπαιδευμένος ταξινομητής για την ανίχνευση πεζών.

Τέλος εφαρμόζουμε τον αλγόριθμο για την ανίχνευση πεζών ως εξής:

foundLocations, foundWeights = **hog.detectMultiScale**(img, hitThreshold, winStride, padding, scale, finalThreshold, useMeanshiftGrouping)

Παράμετροι:

- **img:** Εικόνα εισόδου.
- **hitThreshold:** Τιμή κατωφλιού για την απόσταση μεταξύ των χαρακτηριστικών και του SVM. Εξ' ορισμού δε χρησιμοποιείται κάποια τιμή.
- **winStride:** Βήμα παραθύρου.

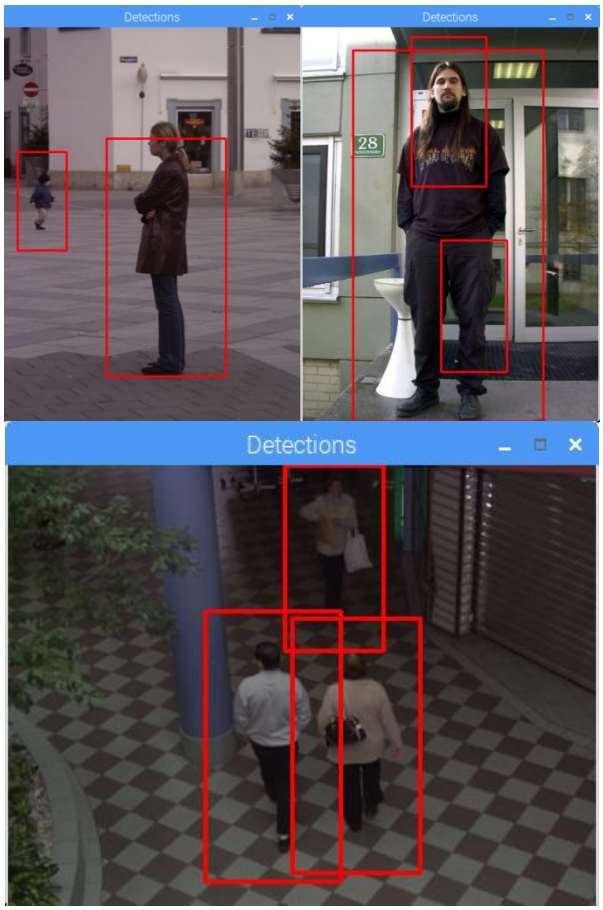
- **padding:** Τιμή που υποδεικνύει τον αριθμό των εικονοστοιχείων στην κατεύθυνση x και y στην οποία το συρόμενο παραθύρου ROI είναι "γεμισμένο" πριν από την εξαγωγή χαρακτηριστικών HOG.
- **scale:** Συντελεστής του ανοίγματος παραθύρου ανίχνευσης.
- **finalThreshold:** Προαιρετική τιμή η οποία συνήθως δε χρησιμοποιείται.
- **useMeanshiftGrouping:** Λογική τιμή που υποδεικνύει εάν πρέπει να γίνει ομαδοποίηση με μέση μετατόπιση (mean shift) για να αντιμετωπιστούν δυνητικά επικαλυπτόμενα κουτιά οριοθέτησης.

Τα παρακάτω προγράμματα αποτελούν δυο συστήματα ανίχνευσης πεζών με χρήση του αλγορίθμου HOG (pedestrian_detection.py και video_pedestrian_detection.py Τα προγράμματα αυτά βρίσκονται στο /home/pi/picvlab/HOG_SVM). Στη πρώτη υλοποίηση εφαρμόζουμε ανίχνευση πεζών σε φωτογραφίες που διαβάζονται από το δίσκο, ενώ στη δεύτερη σε πραγματικό χρόνο με χρήση της RPi Camera.
Εκτελούμε το πρώτο κώδικα με την εντολή:
python ./picvlab/HOG_SVM/pedestrian_detection.py --image ./picvlab/HOG_SVM/images/img
όπου img οι εικόνες person-1.bmp, person-2.bmp και person-3.bmp.

Να τονίσουμε σε αυτό το σημείο ότι, όπως φαίνεται και στο πρόγραμμα, η παράμετρος winStride έχει τιμή (4,4), ή παράμετρος padding έχει τιμή (8,8), και η παράμετρος scale έχει τιμή 1.05.

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

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to the input image")
args = vars(ap.parse_args())

# initialize the HOG descriptor/person detector
```

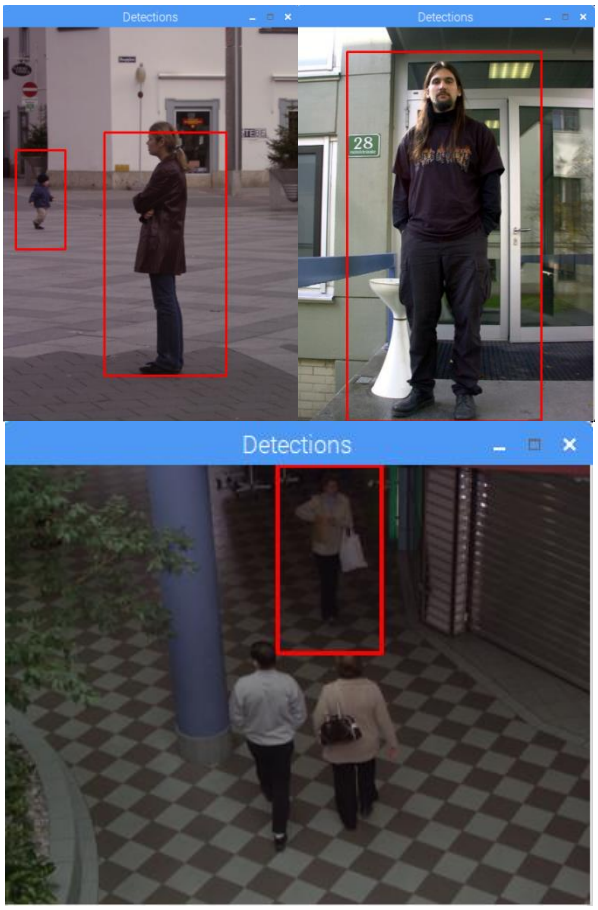
```
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# load the image and resize it
image = cv2.imread(args["image"])
image = imutils.resize(image, width=min(400, image.shape[1]))

# detect people in the image
(rects,weights)=hog.detectMultiScale(image,
winStride=(4,4),padding=(8,8), scale=1.05, useMeanshiftGrouping=False)

# draw the original bounding boxes
for (x, y, w, h) in rects:
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

# show the output image
cv2.imshow("Detections", image); cv2.waitKey(0)
```



Εικόνα 5.9: Ανίχνευση πεζών με winStride=(4,4), padding=(8,8) και scale=1.05

Παρατηρούμε σωστή ανίχνευση στις εικόνες person-1.bmp και person-3.bmp ενώ στην εικόνα person-2.bmp, ενώ έχει γίνει σωστή ανίχνευση, υπάρχουν και δυο λάθος ανιχνεύσεις εντός της σωστής.

Ας δοκιμάσουμε να τρέξουμε ξανά το πρόγραμμα, αυτή τη φορά αλλάζοντας τις τιμές των winStride και padding σε (8,8) και (16,16) αντίστοιχα



Εικόνα 5.10: Ανίχνευση πεζών με winStride=(8,8), padding=(16,16) και scale=1.05

Παρατηρούμε μια ελαφρώς καλύτερη ενός πεζού στη εικόνα person-1.bmp, απολύτως σωστή ανίχνευση στην εικόνα person-2.bmp καθώς και απώλεια ανίχνευσης δύο πεζών στην εικόνα person-3.bmp.

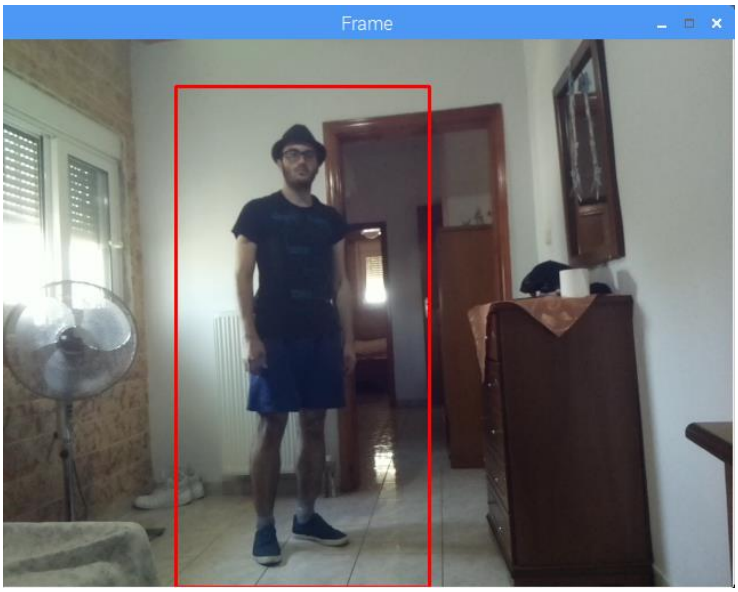
Συμπεραίνουμε λοιπόν πώς για να πετύχουμε το βέλτιστο δυνατό αποτέλεσμα, θα πρέπει να τροποποιούμε κατάλληλα τις τιμές των παραμέτρων winStride, padding και scale ανάλογα με την εικόνα. Οι περισσότερο χρησιμοποιούμενες τιμές είναι οι (8,8), (16,16) και 1.05 αντίστοιχα, αλλά δε δίνουν πάντα το βέλτιστο δυνατό αποτέλεσμα.

Εκτελούμε το δεύτερο κώδικα με την εντολή:

python ./picvlab/HOG_SVM/pedestrian_detection.py

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import argparse
import imutils
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", default = 0,
    help="path to the video file or access the camera (0)")
args = vars(ap.parse_args())

# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
# apply vertical flip if the camera is upside down
camera.vflip = True
rawCapture = PiRGBArray(camera, size=(640, 480))

# allow the camera to warmup
time.sleep(0.1)

# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
      # grab the raw NumPy array representing the image, then initialize
      the timestamp
      # and occupied/unoccupied text
      image = frame.array
      # detect people in the image
```

```
(rects, weights) = hog.detectMultiScale(image, winStride=(8, 8),
      padding=(16, 16), scale=1.05, useMeanshiftGrouping=False)

  # draw the original bounding boxes
  for (x, y, w, h) in rects:
      cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

      # show the frame
      cv2.imshow("Frame", image)
      key = cv2.waitKey(1) & 0xFF

      # clear the stream in preparation for the next frame
      rawCapture.truncate(0)

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



Εικόνα 5.11: Ανίχνευση πεζών σε πραγματικό χρόνο με χρήση της RPi Camera

Τέλος, δοκιμάζοντας ανίχνευση πεζών σε πραγματικό χρόνο με χρήση webcam παρατηρήθηκε μεγάλο lag μεταξύ των frames, επομένως η χρήση webcam δε προτιμάται για το σκοπό αυτό.

5.2.2 Αναγνώριση προτύπων με την Python και την scikit-image

Στην ενότητα αυτή θα αναπτύξουμε ένα σύστημα αναγνώρισης αντικειμένων χρησιμοποιώντας τον αλγόριθμο LBP. Το σύστημα αυτό θα αναπτυχθεί με χρήση της γλώσσας Python και της βιβλιοθήκης scikit-image.

Local Binary Patterns (LBP)

Ο LBP είναι ένας τύπος οπτικού περιγραφέα που χρησιμοποιείται για ταξινόμηση στην μηχανική όραση και περιγράφηκε για πρώτη φορά το 1994. Έχει προσδιοριστεί ότι όταν ο LBP συνδυαστεί με τον HOG, βελτιώνει σημαντικά την απόδοση ανίχνευσης σε ορισμένα σύνολα δεδομένων. Μια σύγκριση διαφόρων βελτιώσεων του αρχικού LBP στο πεδίο της αφαίρεσης υποβάθρου έγινε το 2015 από τους C. Silva, T. Bouwmans και C. Frelicot στο έγγραφο: an eXtended Center-Symmetric Local Binary Pattern for Background Modeling and Subtraction in Videos. Μια πλήρη επισκόπηση των διαφορετικών εκδόσεων του LBP μπορεί να βρεθεί στο έγγραφο: On the Role and the Importance of Features for Background Modeling and Foreground Detection. **[47]**

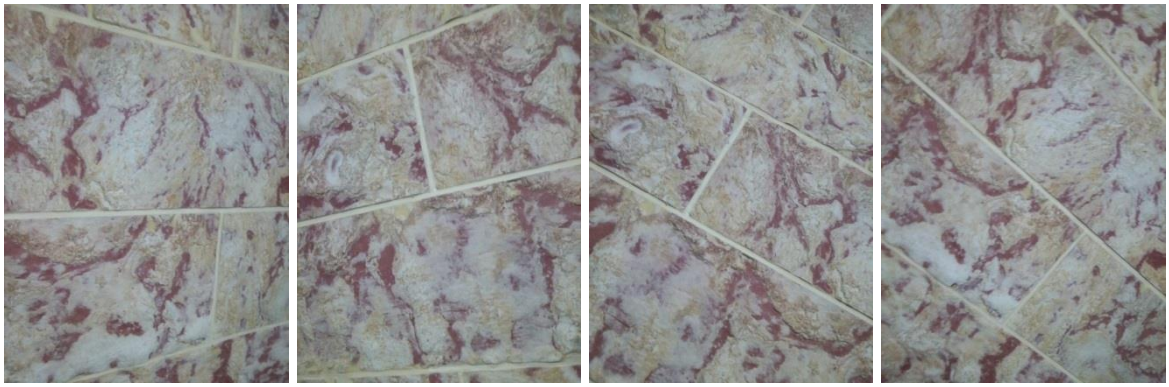
Ο αλγόριθμος LBP, στην απλούστερη μορφή του, αποτελείται από τα εξής βήματα:

1. Διαχωρίζει το εξεταζόμενο παράθυρο σε κελιά (π.χ. 16x16 εικονοστοιχεία για κάθε κελί).
2. Για κάθε εικονοστοιχείο σε ένα κελί, συγκρίνει το εικονοστοιχείο με κάθε ένα από τα 8 γειτονικά του (στην αριστερή του κορυφή, στο αριστερό μεσαίο, στο αριστερό κάτω μέρος, στη δεξιά πλευρά, κλπ.).
3. Όταν η τιμή του κεντρικού εικονοστοιχείου είναι μεγαλύτερη από την τιμή του γειτονικού του, γράφει "0". Διαφορετικά, γράφει "1". Αυτό δίνει έναν 8-ψήφιο δυαδικό αριθμό.
4. Υπολογίζει το ιστόγραμμα, πάνω από το κελί, της συχνότητας κάθε "αριθμού" που εμφανίζεται (δηλαδή, κάθε συνδυασμός των οποίων τα εικονοστοιχεία είναι μικρότερα και τα οποία είναι μεγαλύτερα από το κέντρο). Αυτό το ιστόγραμμα μπορεί να θεωρηθεί ως διάνυσμα χαρακτηριστικών 256 διαστάσεων.
5. Προαιρετικά γίνεται κανονικοποίηση του ιστογράμματος.
6. Συγχρονίζει τα (κανονικοποιημένα) ιστογράμματα όλων των κελιών. Αυτό δίνει ένα διάνυσμα χαρακτηριστικών για ολόκληρο το παράθυρο

Το διάνυσμα χαρακτηριστικών μπορεί τώρα να υποβληθεί σε επεξεργασία χρησιμοποιώντας SVM ή κάποιον άλλον αλγόριθμο μηχανικής μάθησης για την ταξινόμηση εικόνων. Τέτοιοι ταξινομητές μπορούν να χρησιμοποιηθούν για την αναγνώριση προσώπου ή την ανάλυση υφής. **[47]**

Η εφαρμογή του αλγορίθμου LBP μπορεί να γίνει τόσο με τη OpenCV, όσο και με το scikit-image. Θα προτιμήσουμε το scikit-image, αφού με την OpenCV μπορεί να γίνει εφαρμογή του LBP αυστηρά για αναγνώριση προσώπου.

Πριν ξεκινήσουμε να εφαρμόζουμε τον LBP σε εικόνες, πρέπει πρώτα να δημιουργήσουμε ένα σετ δεδομένων με 20 δείγματα (5 δείγματα για κάθε κατηγορία: wall, painting, playstation και football). Τα 16 από αυτές (4 για κάθε κατηγορία) θα χρησιμοποιηθούν για την εκπαίδευση του συστήματος μας ενώ τα υπόλοιπες ως δείγματα προς εξέταση.



Εικόνα 5.12: Δείγματα εκπαίδευσης για τη κατηγορία wall

Εικόνα 5.13: Δείγματα εκπαίδευσης για τη κατηγορία painting

Εικόνα 5.14: Δείγματα εκπαίδευσης για τη κατηγορία playstation

Εικόνα 5.15: Δείγματα εκπαίδευσης για τη κατηγορία football

Εικόνα 5.16: Δείγματα προς εξέταση

Παρακάτω παρατίθεται η δομή της εφαρμογής μας (βρίσκεται στο /home/pi/picvlab/LBP):

```
--- algorithm
|     |--- localbinarypatterns.py
|--- recognize.py
```

Σχήμα 5.12: Δομή της εφαρμογής

Ο algorithm είναι το module το οποίο περιέχει την υλοποίηση του αλγορίθμου (localbinarypatterns.py):

```
# import the necessary packages
from skimage import feature
import numpy as np

class LocalBinaryPatterns:
    def __init__(self, numPoints, radius):
        # store the number of points and radius
        self.numPoints = numPoints
        self.radius = radius

    def describe(self, image, eps=1e-7):
        # compute the Local Binary Pattern representation
        # of the image, and then use the LBP representation
        # to build the histogram of patterns
        lbp = feature.local_binary_pattern(image, self.numPoints,
            self.radius, method="uniform")
        (hist, _) = np.histogram(lbp.ravel(),
            bins = np.arange(0, self.numPoints + 3),
            range = (0, self.numPoints + 2))

        # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + eps)

        # return the histogram of Local Binary Patterns
        return hist
```

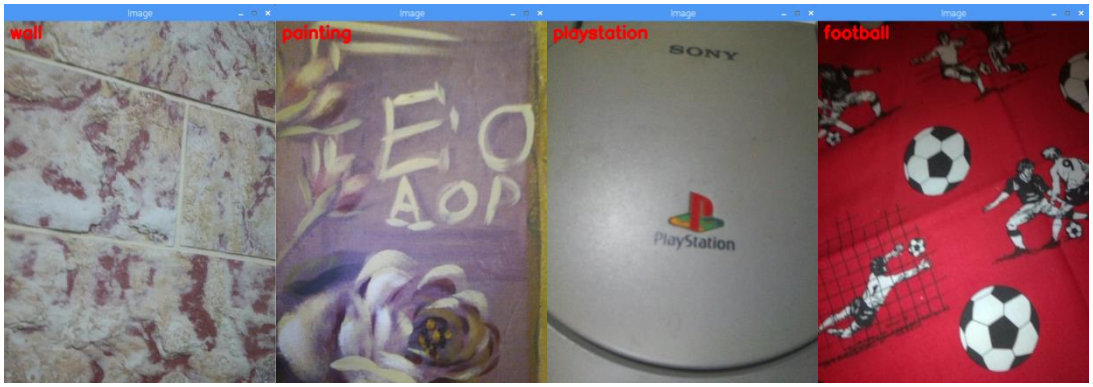
Στο recognize.py γίνεται αρχικά η εκπαίδευση του συστήματος μας και εφαρμόζεται ο αλγόριθμος στα δείγματα προς εξέταση. Τρέχουμε το πρόγραμμα με την εντολή:

python ./picvlab/LBP/recognize.py --training ./picvlab/LBP/images/training --testing ./picvlab/LBP/images/testing

```
# import the necessary packages
from algorithm.localbinarypatterns import LocalBinaryPatterns
from sklearn.svm import LinearSVC
```

```
from imutils import paths
import argparse
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-t", "--training", required=True,
    help="path to the training images")
ap.add_argument("-e", "--testing", required=True,
    help="path to the testing images")
args = vars(ap.parse_args())

# initialize the local binary patterns descriptor along with
# the data and label lists
desc = LocalBinaryPatterns(24, 8)
data = []
labels = []
# loop over the training images
for imagePath in paths.list_images(args["training"]):
    # load the image, convert it to grayscale, and describe it
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = desc.describe(gray)

    # extract the label from the image path, then update the
    # label and data lists
    labels.append(imagePath.split("/")[-2])
    data.append(hist)

# train a Linear SVM on the data
model = LinearSVC(C=100.0, random_state=42)
model.fit(data, labels)

# loop over the testing images
for imagePath in paths.list_images(args["testing"]):
    # load the image, convert it to grayscale, describe it,
    # and classify it
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    hist = desc.describe(gray)
    prediction = model.predict(hist)[0]
```

```
# display the image and the prediction
cv2.putText(image, prediction, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
    1.0, (0, 0, 255), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
```



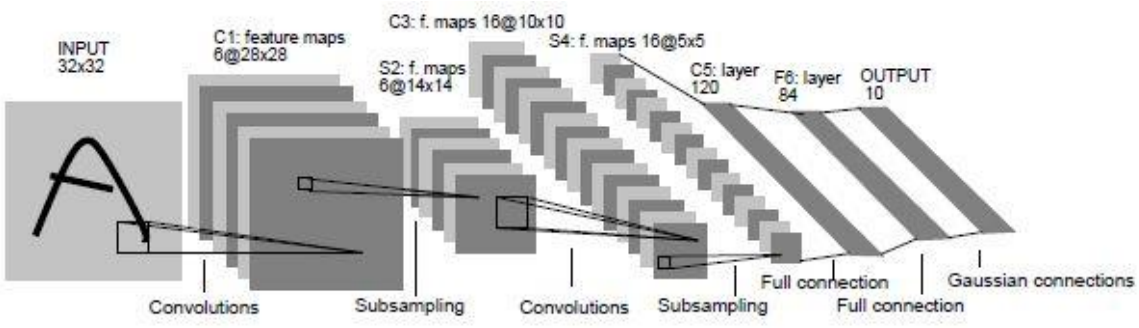
Εικόνα 5.17: Αποτέλεσμα του προγράμματος recognize.py

5.2.3 Αναγνώριση προτύπων με την Python και την Keras

Στην ενότητα αυτή θα αναπτύξουμε ένα σύστημα αναγνώρισης χαρακτήρων χρησιμοποιώντας τον αλγόριθμο LeNet-5. Το σύστημα αυτό θα αναπτυχθεί με χρήση της γλώσσας Python και της βιβλιοθήκης Keras.

LeNet-5

Το LeNet-5 αποτελεί ένα πρωτοποριακό Συνελικτικό Νευρωνικό δίκτυο εφτά επιπέδων σχεδιασμένο για την αναγνώριση χειρόγραφων και εκτυπωμένων χαρακτήρων, το οποίο προτάθηκε από τους Yann LeCun, Leon Bottou, Yoshua Bengio και Patrick Haffner το 1998 στο έγγραφο Gradient Based Learning Applied to Document Recognition. **[48]**



Σχήμα 5.13: Αρχιτεκτονική του LeNet-5 **[49]**

Στο Σχήμα – διακρίνονται τα εφτά επίπεδα του LeNet-5 (δεν υπολογίζεται η είσοδος), όπου κάθε επίπεδο περιέχει παραμέτρους εκπαίδευσης. Η είσοδος είναι ψηφιακή εικόνα με διαστάσεις 32x32. **[49]**

Το επίπεδο C1 αποτελεί ένα επίπεδο συνέλιξης με έξι χάρτες χαρακτηριστικών (feature maps) με διαστάσεις 28x28. Κάθε μονάδα σε κάθε χάρτη χαρακτηριστικών είναι συνδεδεμένη με μια γειτονία 5x5 στην είσοδο. Το επίπεδο αυτό διαθέτει 156 παραμέτρους εκπαίδευσης και 122.304 συνδέσεις. **[49]**

Το επίπεδο S2 αποτελεί ένα επίπεδο υποδειγματοληψίας με έξι χάρτες χαρακτηριστικών με διαστάσεις 14x14. Κάθε μονάδα σε κάθε χάρτη χαρακτηριστικών είναι συνδεδεμένη με μια γειτονία 2x2 στον αντίστοιχο χάρτη χαρακτηριστικών του επιπέδου C1. Το επίπεδο αυτό διαθέτει 12 παραμέτρους εκπαίδευσης και 5.880 συνδέσεις. **[49]**

Το επίπεδο C3 αποτελεί ένα επίπεδο συνέλιξης με δεκαέξι χάρτες χαρακτηριστικών. Κάθε μονάδα σε κάθε χάρτη χαρακτηριστικών είναι συνδεδεμένη με μια γειτονία 5x5 σε πανομοιότυπες θέσεις σε ένα υποσύνολο των χαρτών χαρακτηριστικών του επιπέδου S2. Το επίπεδο αυτό διαθέτει 1.516 παραμέτρους εκπαίδευσης και 151.600 συνδέσεις. **[49]**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Σχήμα 5.14: Σύνδεση των χαρτών χαρακτηριστικών του επιπέδου S2 με αυτούς του επιπέδου C3 **[49]**

Το επίπεδο S4 αποτελεί ένα επίπεδο υποδειγματοληψίας με δεκαέξι χάρτες χαρακτηριστικών με διαστάσεις 5x5. Κάθε μονάδα σε κάθε χάρτη χαρακτηριστικών είναι συνδεδεμένη με μια γειτονία 2x2 στον αντίστοιχο χάρτη χαρακτηριστικών του επιπέδου C3, με παρόμοιο τρόπο με τα επίπεδα C1 και S2. Το επίπεδο αυτό διαθέτει 32 παραμέτρους εκπαίδευσης και 2.000 συνδέσεις. **[49]**

Το επίπεδο C5 αποτελεί ένα επίπεδο συνέλιξης με 120 χάρτες χαρακτηριστικών με διαστάσεις 1x1. Κάθε μονάδα είναι συνδεδεμένη με μια γειτονία 5x5 σε κάθε

χάρτη χαρακτηριστικών του επιπέδου S4. Το επίπεδο αυτό διαθέτει 48.120 παραμέτρους εκπαίδευσης. **[49]**

Το επίπεδο F6 περιέχει 84 μονάδες και είναι πλήρως συνδεδεμένο με το επίπεδο C5. Το επίπεδο αυτό διαθέτει 10.164 παραμέτρους εκπαίδευσης. **[49]**

Παρακάτω παρατίθεται η δομή της εφαρμογής μας (βρίσκεται στο /home/pi/picvlab/LENET):

```
|--- output
|--- algorithm
|    |--- __init__.py
|    |--- cnn
|    |    |--- __init__.py
|    |    |--- networks
|    |    |    |--- __init__.py
|    |    |    |--- lenet.py
|--- lenet_mnist.py
```

Σχήμα 5.15: Δομή της εφαρμογής

Για να διατηρήσουμε τον οργανωμένο κώδικα, θα ορίσουμε ένα module με όνομα algorithm. Και μέσα σε αυτό, θα δημιουργήσουμε ένα sub-module με όνομα cnn, όπου θα αποθηκεύσουμε τις υλοποιήσεις του CNN, μαζί με τα βοηθητικά προγράμματα που σχετίζονται με τα CNN. Μέσα στο cnn, θα δημιουργήσουμε ένα sub-module με όνομα networks, το οποίο περιέχει την υλοποίηση του αλγορίθμου (lenet.py):

```
# import the necessary packages
from keras.models import Sequential
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense

class LeNet:
    @staticmethod
    def build(width, height, depth, classes, weightsPath=None):
        # initialize the model
        model = Sequential()

        # first set of CONV => RELU => POOL
```

```
model.add(Convolution2D(20, 5, 5, border_mode="same",
    input_shape=(depth, height, width)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# second set of CONV => RELU => POOL
model.add(Convolution2D(50, 5, 5, border_mode="same"))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# set of FC => RELU layers
model.add(Flatten())
model.add(Dense(500))
model.add(Activation("relu"))

# softmax classifier
model.add(Dense(classes))
model.add(Activation("softmax"))

# if a weights path is supplied (inicating that the model was
# pre-trained), then load the weights
if weightsPath is not None:
    model.load_weights(weightsPath)

# return the constructed network architecture
return model
```

Στο lenet_mnist.py γίνεται αρχικά η εκπαίδευση του δικτύου (ή η φόρτωση του μοντέλου εάν το δίκτυό μας είναι προ-εκπαιδευμένο) και αξιολογείται η απόδοση του δικτύου στο σύνολο δεδομένων MNIST. Τέλος στο module output θα αποθηκεύσουμε το μοντέλο, αφού το εκπαιδεύσουμε, έτσι ώστε να μπορούμε να ταξινομήσουμε ψηφία όταν εφαρμόσουμε ξανά τον αλγόριθμο, χωρίς να ξανά-εκπαιδεύσουμε το μοντέλο.

Σχήμα 5.16: Τμήμα από το σύνολο δεδομένων MNIST

Εκτελούμε το κώδικα προκειμένου να εκπαιδεύουμε το μοντέλο με την εντολή:

$ python ./picvlab/LENET/lenet_mnist.py --save-model 1 --weights ./picvlab/LENET/output/lenet_weights.hdf5

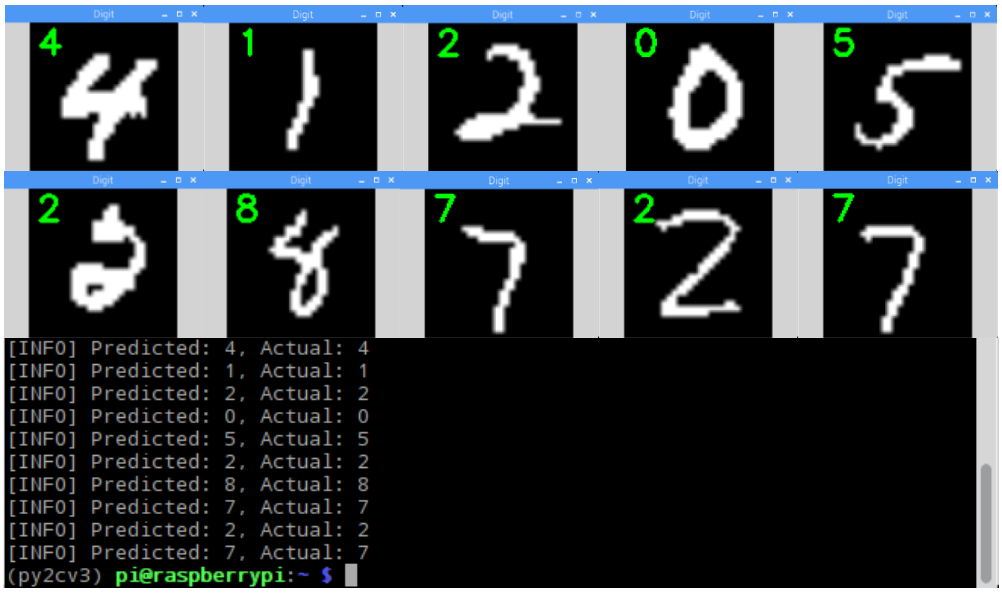
Αν το δίκτυο είναι προ-εκπαιδευμένο, εκτελούμε τον κώδικα με την εξής εντολή:

$ python ./picvlab/LENET/lenet_mnist.py --load-model 1 --weights ./picvlab/LENET/output/lenet_weights.hdf5

```
# import the necessary packages
from algorithm.cnn.networks import LeNet
from sklearn.cross_validation import train_test_split
from sklearn import datasets
from keras.optimizers import SGD
from keras.utils import np_utils
import numpy as np
import argparse
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-s", "--save-model", type=int, default=-1,
    help="(optional) whether or not model should be saved to disk")
ap.add_argument("-l", "--load-model", type=int, default=-1,
    help="(optional) whether or not pre-trained model should be loaded")
ap.add_argument("-w", "--weights", type=str,
    help="(optional) path to weights file")
args = vars(ap.parse_args())

# grab the MNIST dataset (if this is your first time running this
# script, the download may take a minute -- the 55MB MNIST dataset
# will be downloaded)
print("[INFO] downloading MNIST...")
dataset = datasets.fetch_mldata("MNIST Original")

# reshape the MNIST dataset from a flat list of 784-dim vectors, to
# 28 x 28 pixel images, then scale the data to the range [0, 1.0]
# and construct the training and testing splits
data = dataset.data.reshape((dataset.data.shape[0], 28, 28))
data = data[:, np.newaxis, :, :]
```

```
(trainData, testData, trainLabels, testLabels) = train_test_split(
    cv2.divide(data,255), dataset.target.astype("int"), test_size=0.33)

# transform the training and testing labels into vectors in the
# range [0, classes] -- this generates a vector for each label,
# where the index of the label is set to `1` and all other entries
# to `0`; in the case of MNIST, there are 10 class labels
trainLabels = np_utils.to_categorical(trainLabels, 10)
testLabels = np_utils.to_categorical(testLabels, 10)

# initialize the optimizer and model
print("[INFO] compiling model...")
opt = SGD(lr=0.01)
model = LeNet.build(width=28, height=28, depth=1, classes=10,
    weightsPath=args["weights"] if args["load_model"] > 0 else None)
model.compile(loss="categorical_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# only train and evaluate the model if we *are not* loading a
# pre-existing model
if args["load_model"] < 0:
    print("[INFO] training...")
    model.fit(trainData, trainLabels, batch_size=128, nb_epoch=20,
        verbose=1)

    # show the accuracy on the testing set
    print("[INFO] evaluating...")
    (loss, accuracy) = model.evaluate(testData, testLabels,
        batch_size=128, verbose=1)
    print("[INFO] accuracy: {:.2f}%".format(accuracy * 100))
# check to see if the model should be saved to file
if args["save_model"] > 0:
    print("[INFO] dumping weights to file...")
    model.save_weights(args["weights"], overwrite=True)

# randomly select a few testing digits
for i in np.random.choice(np.arange(0, len(testLabels)), size=(10,)):
    # classify the digit
    probs = model.predict(testData[np.newaxis, i])
    prediction = probs.argmax(axis=1)
```

```
# resize the image from a 28 x 28 image to a 96 x 96 image so we
# can better see it
image = (testData[i][0] * 255).astype("uint8")
image = cv2.merge([image] * 3)
image = cv2.resize(image, (96, 96), interpolation=cv2.INTER_LINEAR)
cv2.putText(image, str(prediction[0]), (5, 20),
    cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)

# show the image and prediction
print("[INFO] Predicted: {}, Actual: {}".format(prediction[0],
    np.argmax(testLabels[i])))
cv2.imshow("Digit", image)
cv2.waitKey(0)
```



Εικόνα 5.18: Αποτέλεσμα του προγράμματος lenet_mnist.py

5.2.4 Αναγνώριση προτύπων με τη Python και τη Dlib

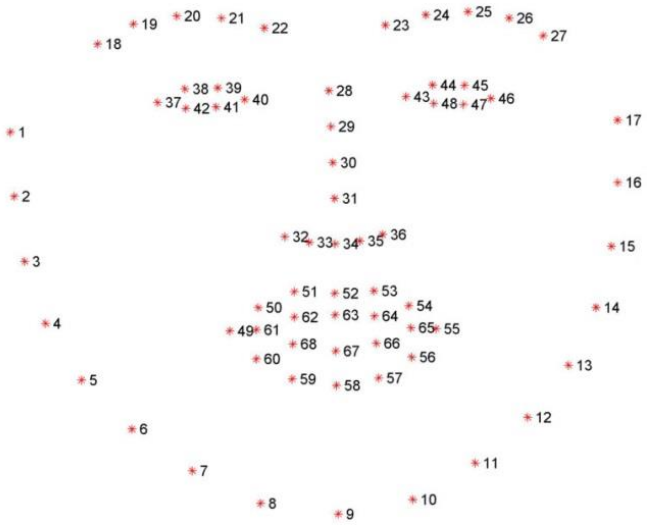
Το παρακάτω πρόγραμμα αποτελεί ένα σύστημα ανίχνευσης ορόσημων του προσώπου (facial landmarks). Τα ορόσημα προσώπου χρησιμοποιούνται για τον εντοπισμό και την εκπροσώπηση σημαντικών περιοχών του προσώπου, όπως τα μάτια, τα φρύδια, τη μύτη, το στόμα και το σαγόνι. Τα ορόσημα προσώπου έχουν εφαρμοστεί με επιτυχία στην ευθυγράμμιση προσώπου, στην εκτίμησης θέσης (πόζας) του κεφαλιού, στην ανίχνευση ανοιγοκλείματος των ματιών και πολλά άλλα. **[50]**

Για την ανίχνευση των ορόσημων του προσώπου θα πρέπει πρώτα να ανιχνεύσουμε το πρόσωπο. Αυτό μπορεί να γίνει με πολλούς τρόπους. Στη συγκεκριμένη υλοποίηση θα χρησιμοποιήσουμε έναν προ-εγκατεστημένο περιγραφέα της Dlib, ο οποίος βασίζεται σε HOG + SVM. Αφού εντοπίσουμε το πρόσωπο, αναζητούμε τα ορόσημα προσώπου εντός του προσώπου που ανιχνεύσαμε προηγουμένως. Για το σκοπό αυτό, χρησιμοποιούμε έναν προ-εκπαιδευμένο περιγραφέα της Dlib (shape_predictor_68_face_landmarks.dat), ο οποίος βασίζεται στο έγγραφο του Kazemi και Sullivan: One Millisecond Face Alignment with an Ensemble of Regression Trees. **[50]**

Σύμφωνα με το έγγραφο, χρησιμοποιείται ένα εκπαιδευτικό σετ με επισημανθέντα ορόσημα προσώπου σε μια εικόνα. Αυτές οι εικόνες φέρουν χειροκίνητη ετικέτα, καθορίζοντας συγκεκριμένες συντεταγμένες (x, y) των περιοχών που περιβάλλουν κάθε δομή του προσώπου. Δεδομένων αυτών των δεδομένων εκπαίδευσης, ένα σύνολο από δέντρα παλινδρόμησης εκπαιδεύονται για να εκτιμήσουν τις θέσεις ορόσημο του προσώπου απευθείας από τις ίδιες τις εντάσεις των εικονοστοιχείων. **[50]**

Το τελικό αποτέλεσμα είναι ένας ανιχνευτής ορόσημων προσώπου που μπορεί να χρησιμοποιηθεί για την ανίχνευση σημείων προσώπου σε πραγματικό χρόνο με προβλέψεις υψηλής ποιότητας. **[50]**

Ο προ-εκπαιδευμένος ανιχνευτής ορόσημων προσώπου της Dlib χρησιμοποιείται για να εκτιμηθεί η θέση των 68 συντεταγμένων (x, y) που αντιστοιχούν στις δομές του προσώπου. **[50]**



Σχήμα 5.17: Αναπαράσταση των 68 συντεταγμένων (x, y) που αντιστοιχούν στις δομές του προσώπου **[50]**

Αρχικά για την υλοποίηση του συστήματος εκτός της βιβλιοθήκης Dlib θα χρησιμοποιήσουμε και την imutils και συγκεκριμένα τη συνάρτηση shape_to_np() που βρίσκεται στο αρχείο face_utils.py. η συνάρτηση αυτή παίρνει την έξοδο του ανιχνευτή ορόσημων προσώπου, η οποία είναι ένα αντικείμενο με τις συντεταγμένες των σημείων αυτών και το μετατρέπει σε πίνακα τιμών της NumPy.

```
def shape_to_np(shape, dtype="int"):
    # initialize the list of (x, y)-coordinates
    coords = np.zeros((68, 2), dtype=dtype)

    # loop over the 68 facial landmarks and convert them
    # to a 2-tuple of (x, y)-coordinates
    for i in range(0, 68):
        coords[i] = (shape.part(i).x, shape.part(i).y)

    # return the list of (x, y)-coordinates
    return coords
```

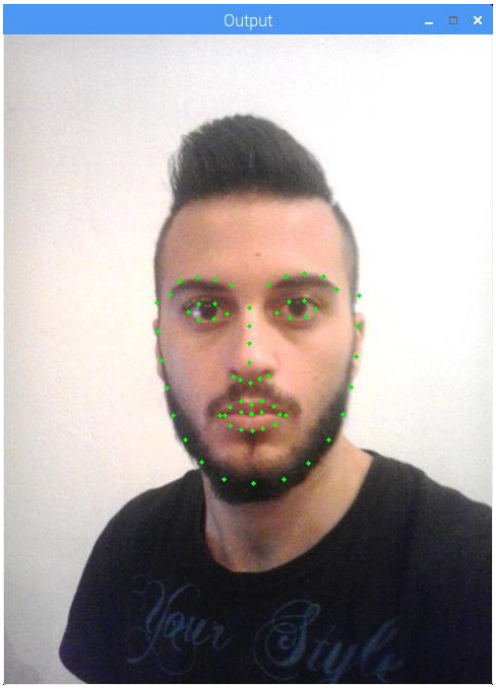
Στη συνέχεια παραθέτουμε το κώδικα που υλοποιεί το σύστημα ανίχνευσης ορόσημων προσώπου (facial_landmarks.py. Το πρόγραμμα αυτό βρίσκεται στο /home/pi/picvlab/facial_landmarks). Αρχικά, ανιχνεύουμε το πρόσωπο

χρησιμοποιώντας έναν προ-εγκατεστημένο περιγραφέα βασισμένο σε HOG + SVM και στη συνέχεια

Εκτελούμε το κώδικα με την εξής εντολή:

$ python ./picvlab/facial_landmarks/facia_landmarks.py

```
# import the necessary packages
from imutils import face_utils
import dlib
import cv2
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
p = "./picvlab/facial_landmarks/shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(p)

# load the input image and convert it to grayscale
image = cv2.imread("./picvlab/facial_landmarks/face.jpg")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# detect faces in the grayscale image
rects = detector(gray, 0)

# loop over the face detections
for (i, rect) in enumerate(rects):
    # determine the facial landmarks for the face region, then
    # convert the facial landmark (x, y)-coordinates to a NumPy
    # array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)

    # loop over the (x, y)-coordinates for the facial landmarks
    # and draw them on the image
    for (x, y) in shape:
        cv2.circle(image, (x, y), 2, (0, 255, 0), -1)

# show the output image with the face detections + facial landmarks
cv2.imshow("Output", image)
cv2.waitKey(0)
```



Εικόνα 5.19: Ανίχνευση ορόσημων προσώπου

Τέλος, δοκιμάζοντας ανίχνευση οροσήμων προσώπου σε πραγματικό χρόνο με χρήση RPi Camera και usb webcam, παρατηρήθηκε αρκετά μεγάλο lag μεταξύ των frames της ροής βίντεο. Αυτό συμβαίνει διότι ο εντοπισμός προσώπου με τη Dlib αποτελεί μια σχετικά αργή διαδικασία και επειδή το RPi δεν είναι τόσο δυνατό στην αναγνώριση προτύπων σε πραγματικό χρόνο με χρήση της Dlib.

ΚΕΦΑΛΑΙΟ 6

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

6.1 Συμπεράσματα

Στη παρούσα εργασία αναπτύχθηκαν αλγόριθμοι επεξεργασίας εικόνας και συστήματα αναγνώρισης προτύπων στο RPi 3 Model B. Συγκεκριμένα, στο αντικείμενο της επεξεργασίας εικόνας παρουσιάσαμε ορισμένες συναρτήσεις της OpenCV και της Matplotlib για χειρισμό εικόνων και εφαρμογής τεχνικών όπως επέκταση της αντίθεσης και φιλτράρισμα εικόνας. Στο αντικείμενο της αναγνώρισης προτύπων αναπτύχθηκαν: δύο συστήματα αναγνώρισης χαρακτήρων (ψηφίων και αγγλικού αλφαβήτου) με χρήση του αλγορίθμου KNN, ένα σύστημα ανίχνευσης προσώπων με τη μέθοδο εντοπισμού αντικειμένων κατά Viola & Jones, ένα σύστημα ανίχνευσης πεζών με χρήση του αλγορίθμου HOG, ένα σύστημα αναγνώρισης μοτίβων με χρήση του αλγορίθμου LBP, και ένα σύστημα αναγνώρισης ψηφίων με χρήση του αλγορίθμου LeNet-5. Για τα συστήματα αυτά χρησιμοποιήθηκαν οι βιβλιοθήκες OpenCV, scikit-image και Keras. Επίσης, χρησιμοποιήθηκε η βιβλιοθήκη Dlib για την ανάπτυξη ενός συστήματος ανίχνευσης ορόσημων προσώπου.

Με την ολοκλήρωση της παρούσας εργασίας, προκύπτουν αρκετά συμπεράσματα, θετικά και αρνητικά, τα οποία χρήζουν ανάλυσης. Συγκεκριμένα:

Στο αντικείμενο της επεξεργασίας εικόνας δεν αντιμετωπίσαμε κάποιο ιδιαίτερο πρόβλημα, ούτε ως προς τα αποτελέσματα των προγραμμάτων ούτε ως προς την απόδοση του RPi κατά την εκτέλεση των. Αυτό οφείλεται στο γεγονός ότι η OpenCV και η Matplotlib διαθέτουν αρκετά βελτιστοποιημένες συναρτήσεις, κάτι που τις καθιστούν ικανές να “τρέχουν” σε συστήματα με όχι δυνατό υλικό, όπως το RPi.

Στο αντικείμενο της αναγνώρισης προτύπων προκύπτουν τα εξής συμπεράσματα:

- Στα συστήματα αναγνώρισης χαρακτήρων με χρήση του αλγορίθμου KNN είχαμε ποσοστό επιτυχίας 91,76 % και 93,06% αντίστοιχα. Θεωρώ ότι αυτό οφείλεται στο γεγονός ότι τόσο τα δείγματα εκπαίδευσης, όσο και τα δείγματα προς εξέταση προέρχονται από τα ίδια αρχεία, και έχουν αρκετές

ομοιότητες μεταξύ τους και για το λόγο αυτό, αντιστοιχίζονται με μεγάλη επιτυχία.

- Στο σύστημα ανίχνευσης προσώπων με χρήση της μεθόδου κατά Viola & Jones είχαμε ικανοποιητικά αποτελέσματα αν αναλογιστούμε πως η μέθοδος αυτή επηρεάζεται σημαντικά από τις συνθήκες φωτισμού. Επίσης είχαμε τη δυνατότητα να δοκιμάσουμε ανίχνευση προσώπων σε πραγματικό χρόνο τόσο με χρήση της RPi Camera, όσο και με χρήση usb webcam με αποτελέσματα επίσης ικανοποιητικά. Το μόνο αρνητικό στοιχείο που παρατηρήθηκε ήταν μια όχι συχνή αύξηση θερμοκρασίας του RPi κατά τη χρήση της RPi Camera.
- Στο σύστημα ανίχνευσης πεζών με χρήση του αλγορίθμου HOG είχαμε ικανοποιητικά αποτελέσματα αν αναλογιστούμε πως η μέθοδος αυτή επηρεάζεται σημαντικά από τις συνθήκες φωτισμού. Επίσης είχαμε τη δυνατότητα να δοκιμάσουμε ανίχνευση πεζών σε πραγματικό χρόνο με χρήση της RPi Camera με αποτελέσματα επίσης ικανοποιητικά. Με χρήση usb webcam παρατηρήθηκε σημαντικό lag μεταξύ των frames της ροής του βίντεο, καθιστώντας την ανίχνευση πεζών μη αξιόλογη. Τέλος παρατηρήθηκε σημαντική αύξηση της θερμοκρασίας του RPi κατά τη χρήση της RPi Camera.
- Στο σύστημα αναγνώρισης μοτίβων με χρήση του αλγορίθμου LBP είχαμε ποσοστό επιτυχίας 100%. Αυτό οφείλεται στο γεγονός ότι τα δείγματα προς εξέταση έχουν πάρα πολλά κοινά με τα δείγματα εκπαίδευσης.
- Στο σύστημα αναγνώρισης χαρακτήρων με χρήση του αλγορίθμου LeNet-5 τα αποτελέσματα ήταν άκρως ικανοποιητικά με ποσοστά επιτυχίας μεγαλύτερα του 90%. Το μόνο αρνητικό στοιχείο που παρατηρήθηκε ήταν το γεγονός ότι το RPi είναι αρκετά αδύναμο για την εκπαίδευση του δικτύου λόγω του υλικού του και για το λόγο αυτό χρησιμοποιήθηκε προ-εκπαιδευμένο μοντέλο.
- Στο σύστημα ανίχνευσης ορόσημων προσώπου με χρήση της βιβλιοθήκης Dlib ήταν αρκετά ικανοποιητικά αν και η ταχύτητα εκτέλεσης ήταν σχετικά αργή. Αυτό οφείλεται στο γεγονός ότι η ανίχνευση προσώπου με τη Dlib είναι αργή. Τέλος διαπιστώθηκε πως το RPi δεν είναι ικανό για αναγνώριση προτύπων σε πραγματικό χρόνο με χρήση της Dlib καθώς τόσο με την RPi

Camera, όσο και με τη usb webcam παρατηρήθηκε σημαντικό lag μεταξύ των frames της ροής του βίντεο καθιστώντας την ανίχνευση μη αξιόλογη.

6.2 Προοπτικές

Η ολοκλήρωση της παρούσας πτυχιακής εργασίας σε ένα τόσο ενδιαφέρον αντικείμενο όπως αυτό της επεξεργασίας εικόνας και της αναγνώρισης προτύπων, μόνο κίνητρα για μελλοντική ενασχόληση θα μπορούσε να προσφέρει, τόσο στη βελτίωση και εξέλιξη των υπαρχόντων συστημάτων όσο και στην μελέτη επιπρόσθετων αλγορίθμων. Μερικές από τις προοπτικές ενασχόλησης στο μέλλον που πηγάζουν από τη συγκεκριμένη εργασία είναι:

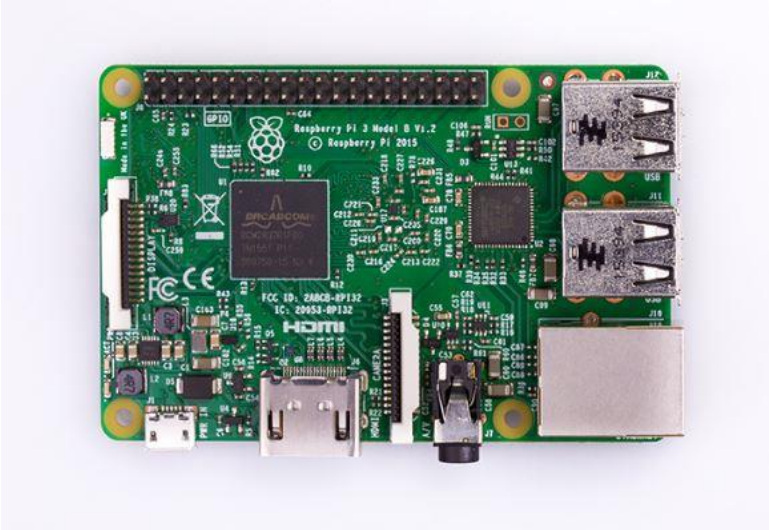
- Ανάπτυξη διεπαφής για χειρισμό εικόνων και εφαρμογή αλγορίθμων επεξεργασίας εικόνας.
- Ανάπτυξη συστήματος ανίχνευσης αντικειμένων (όχι προσώπου, χαρακτηριστικών του και πεζών) με χρήση είτε της μεθόδου εντοπισμού αντικειμένων κατά Viola & Jones, είτε με χρήση του αλγορίθμου HOG και βελτιστοποίηση του για εφαρμογή σε πραγματικό χρόνο.
- Ανάπτυξη συστήματος αναγνώρισης χαρακτήρων σε πραγματικό χρόνο και με δυνατότητα εγγραφής των αποτελεσμάτων σε αρχείο με χρήση του αλγορίθμου LeNet-5.

Μελέτη αλγορίθμων αναγνώρισης προτύπων που δεν αναφέρθηκαν στη παρούσα εργασία, για παράδειγμα: αλγόριθμος ομαδοποίησης K-μέσων, δίκτυα Bayes κλπ., και ανάπτυξη συστημάτων με χρήση αυτών.

ΠΑΡΑΡΤΗΜΑ Α

Α.1 Χαρακτηριστικά του Raspberry Pi 3 Model B

Παρακάτω βλέπουμε το RPi 3 Model B και παραθέτουμε τα χαρακτηριστικά του (specifications).



Σχήμα Α.1: Raspberry Pi 3 Model B

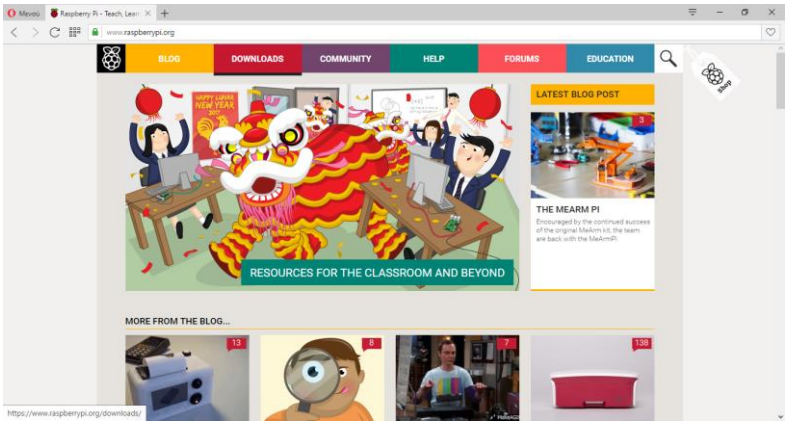
Χαρακτηριστικά – Specifications

- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

A.2 Εγκατάσταση του λειτουργικού Raspbian Jessie στο Raspberry Pi 3 Model B χρησιμοποιώντας τα Windows 10

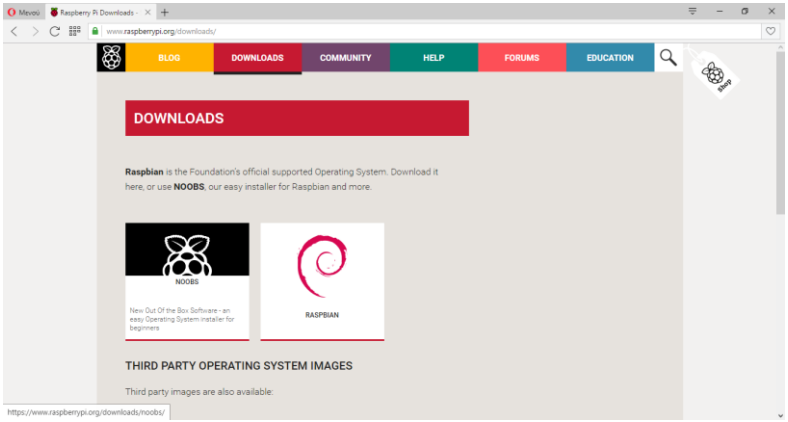
Για την εγκατάσταση του Raspbian Jessie στο RPi ακολουθούμε τη παρακάτω διαδικασία:

Ξεκινώντας θα πρέπει να κατεβάσουμε το Raspbian. Αυτό θα το κάνουμε πηγαίνοντας στη σελίδα www.raspberrypi.org και κάνοντας κλικ στο DOWNLOADS στο πάνω μέρος της σελίδας.



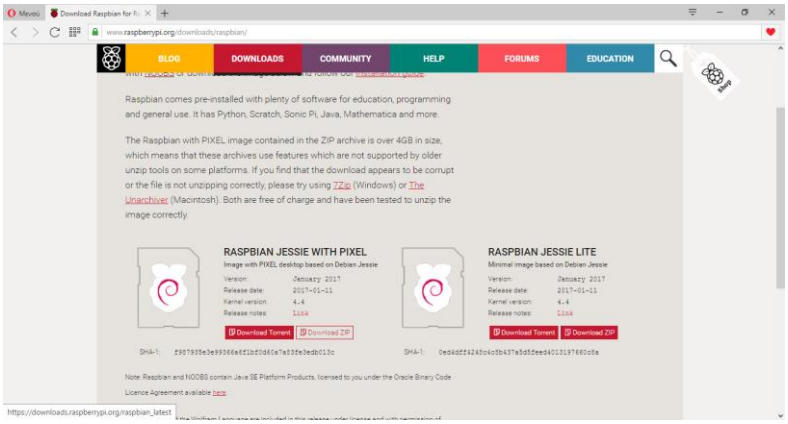
Σχήμα A.2: Αρχική Σελίδα (Home Page)

Στη συνέχεια κάνουμε κλικ στο NOOBS.



Σχήμα A.3: Σελίδα Λήψεων (Downloads)

Κατεβάζουμε το Raspbian Jessie κάνοντας κλικ στο Download ZIP



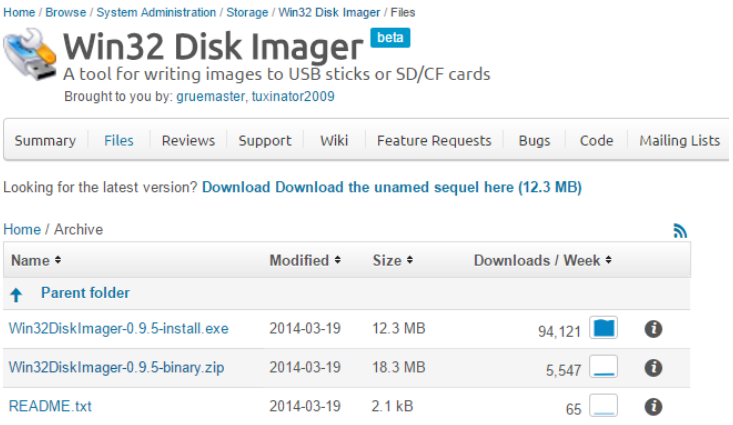
Σχήμα Α.4: Λήψη του Raspbian ως αρχείο ZIP

Όταν γίνει η λήψη του zip αρχείου το αποσυμπιέζουμε (για την ορθή αποσυμπίεση του αρχείου η ιστοσελίδα μας προτείνει να χρησιμοποιήσουμε το πρόγραμμα 7Zip). Το αρχείο που θα προκύψει θα είναι ένα αρχείο εικόνας δίσκου (disc image file).

Τοποθετούμε τη κάρτα μνήμης στη κατάλληλη υποδοχή (SD card reader) του υπολογιστή και ελέγχουμε ποιο γράμμα (drive letter) έχει ανατεθεί στη κάρτα μνήμης (π.χ. F:). Αν ο υπολογιστής δεν διαθέτει SD card reader μπορούμε να συνδέσουμε έναν αντάπτορα (SD adapter) σε μια θύρα USB.

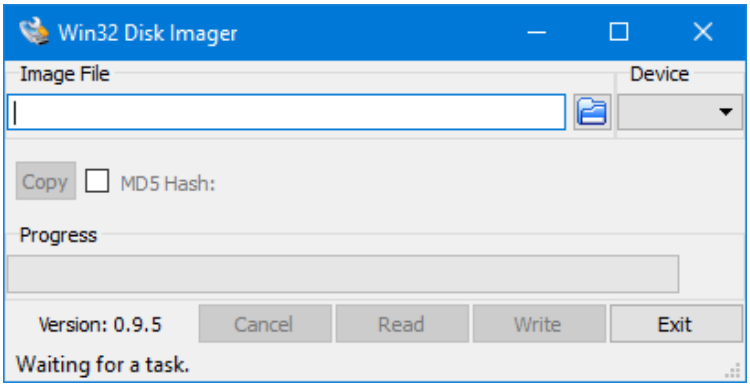
Κατεβάζουμε το Win32DiskImager από το:

https://sourceforge.net/projects/win32diskimager/files/Archive/ ως zip.



Σχήμα Α.5: Λήψη του Win32DiskImager-0.9.5-binary.zip

Αποσυμπιέζουμε το αρχείο και τρέχουμε το Win32DiskImager.exe με δικαιώματα διαχειριστή (Δεξί κλικ -> Εκτέλεση ως διαχειριστής).



Σχήμα Α.6: Το γραφικό περιβάλλον του Win32DiskImager

Στο Image File επιλέγουμε το αρχείο εικόνας δίσκου.

Στο Device επιλέγουμε το γράμμα που αντιστοιχεί στη κάρτα μνήμης.

Κάνουμε κλικ στο Write και περιμένουμε μέχρι να ολοκληρωθεί το γράψιμο.

Κλείνουμε την εφαρμογή και κάνουμε εξαγωγή της κάρτας μνήμης με ασφαλή κατάργηση.

Αυτό ήταν. Το μόνο που μένει να κάνουμε είναι να τοποθετήσουμε τη κάρτα μνήμης στο Raspberry Pi, να το ανοίξουμε και μετά να ενημερώσουμε το firmware

Γράφοντας στο LXTerminal τις εξής εντολές:

$ sudo apt-get update

$ sudo apt-get upgrade

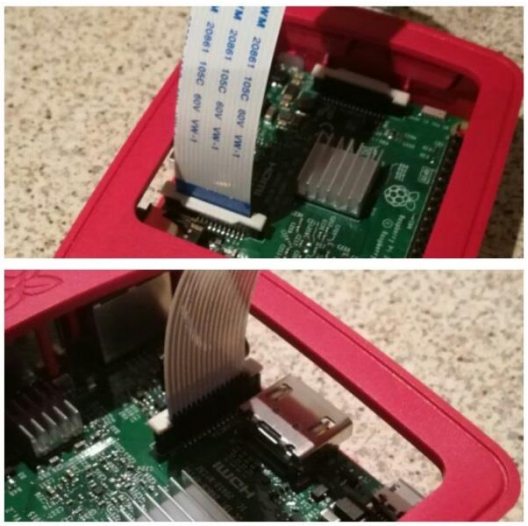
Μόλις ολοκληρωθεί η ενημέρωση κάνουμε επανεκκίνηση.

Α.3 Σύνδεση της Raspberry Pi Camera V2 στο Raspberry Pi 3 Model B

Η σύνδεση της Raspberry Pi Camera V2 αποτελεί μια απλή διαδικασία ή οποία είναι η εξής:

Έχοντας το RPi κλειστό και αποσυνδεδεμένο από τη πρίζα συνδέουμε σε αυτό τη κάμερα όπως στις παρακάτω εικόνες.

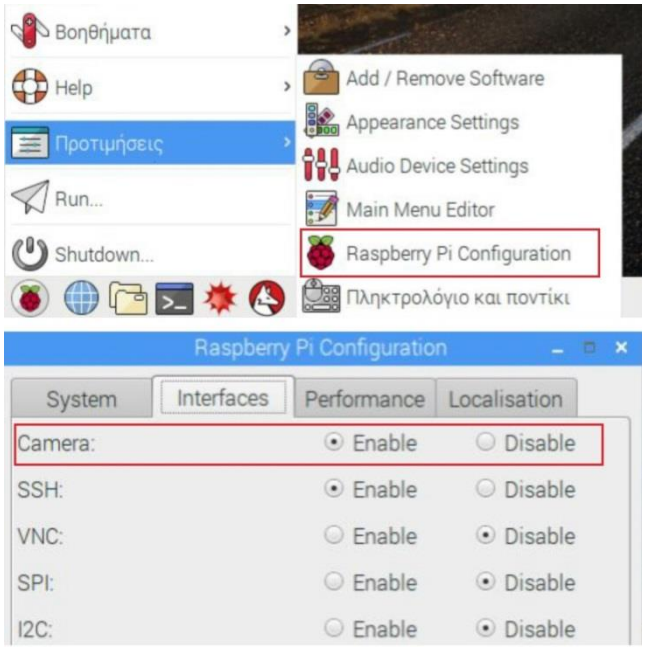
Εικόνα Α.1: Η σύνδεση της RPi Camera V2 στο RPi 3 Model B



Εικόνα Α2: Η υποδοχή της κάμερας βρίσκεται ανάμεσα στη θύρα HDMI και στην Audio Jack. Η μπλε γραμμή θα πρέπει να είναι στη μεριά της Audio Jack

Ανοίγουμε το RPi.

Πηγαίνουμε στο Raspberry Pi Configuration (Έναρξη -> Προτιμήσεις -> Raspberry Pi Configuration -> Interfaces (Tab)) και ελέγχουμε αν η Camera είναι Enabled. Αν δεν είναι επιλέγουμε το Enabled και κάνουμε επανεκκίνηση.



Σχήμα Α.7: Ενεργοποίηση της RPi Camera

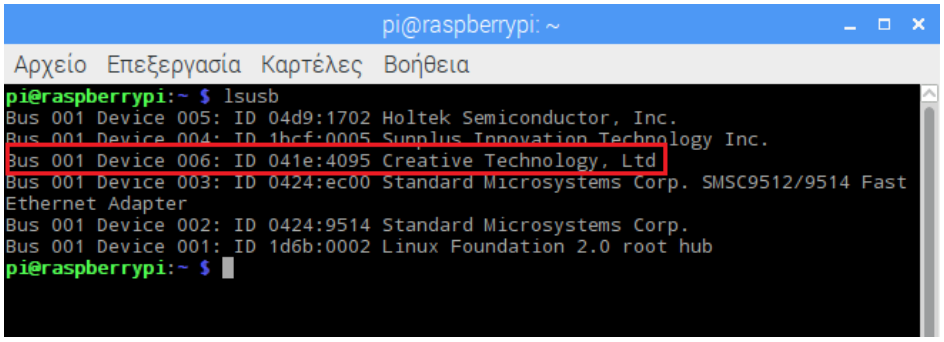
Τέλος κάνουμε εγκατάσταση του πακέτου (package) picamera, ενός πακέτου με το οποίο θα μπορούμε να χειριστούμε τη Raspberry Pi Camera χρησιμοποιώντας τη γλώσσα Python. Η εγκατάσταση γίνεται γράφοντας στο LXTerminal την εντολή: $ pip install “picamera[array]”. Θα δούμε αναλυτικά την εγκατάσταση του πακέτου στο Παράρτημα Β.

Α.4 Σύνδεση usb webcam στο Raspberry Pi 3 Model B

Αντί να χρησιμοποιήσουμε τη Raspberry Pi Camera, μπορούμε να χρησιμοποιήσουμε μια usb webcam. Σε αυτό το σημείο πρέπει να επισημάνουμε ότι δεν είναι όλες οι usb webcams συμβατές με το Raspberry Pi καθώς και ότι δεν είναι βέβαιο ότι θα δουλέψουν σωστά ακόμα και αν είναι συμβατές. Η σύνδεση μιας usb webcam είναι μια διαδικασία πολύ απλή και είναι η εξής:

Συνδέουμε τη webcam σε μια υποδοχή usb του RPi.

Ελέγχουμε αν η webcam είναι συμβατή με το RPi ανοίγοντας το LXTerminal και γράφοντας την εντολή lsusb. Αν στο αποτέλεσμα της εντολής αυτής δούμε τη webcam που έχουμε συνδέσει τότε είναι συμβατή. Η webcam που θα χρησιμοποιήσουμε είναι η Creative Live! Cam Sync HD.



Εικόνα Α.3: Έλεγχος συμβατότητας της webcam

Κάνουμε εγκατάσταση του προγράμματος fswebcam γράφοντας στο LXTerminal την εντολή: $ sudo apt-get install fswebcam.

Αφού ολοκληρωθεί η εγκατάσταση μπορούμε να χρησιμοποιήσουμε τη κάμερα. Μπορούμε να τραβήξουμε μια φωτογραφία με την εντολή:

$ fswebcam image.jpg.

Η φωτογραφία θα αποθηκευτεί στο τρέχοντα κατάλογο ~ (ο τρέχον κατάλογος είναι ο /home/pi).

A.5 Picamera Documentation

Το documentation για το πακέτο Picamera είναι διαθέσιμο στο παρακάτω ιστότοπο:

http://picamera.readthedocs.io/en/release-1.13/

Αυτή τι στιγμή, η τελευταία έκδοση του πακέτου είναι η Version 1.13 και είναι συμβατή με την έκδοση Python 2.7 (ή νεότερη) και με την έκδοση Python 3.2 (ή νεότερη).

A.6 fswebcam Documentation

Το documentation για το πακέτο fswebcam είναι διαθέσιμο στο παρακάτω ιστότοπο:

https://manpages.debian.org/jessie/fswebcam/fswebcam.1.en.html

Αυτή τι στιγμή, η τελευταία έκδοση του πακέτου είναι η Version 20140113 και είναι συμβατή με το λειτουργικό Raspbian Jessie. Επίσης μπορούμε να δούμε το documentation από το LXTerminal με την εντολή:

$ man fswebcam

ΠΑΡΑΡΤΗΜΑ Β

Β.1 Εγκατάσταση της βιβλιοθήκης OpenCV στο RPi 3 Model B

Στην ενότητα αυτή παρουσιάζεται βήμα-βήμα η εγκατάσταση της βιβλιοθήκης OpenCV στο RPi 3 Model B.

Βήμα #1: Εγκατάσταση εξαρτήσεων

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να ενημερώσουμε το firmware του Raspberry Pi.

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

Μόλις ολοκληρωθεί η ενημέρωση θα πρέπει να κάνουμε επανεκκίνηση.

```
$ sudo reboot
```

Στη συνέχεια θα πρέπει να εγκαταστήσουμε κάποια εργαλεία προγραμματιστή (developer tools).

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

Τώρα θα εγκαταστήσουμε κάποια πακέτα εισόδου/εξόδου εικόνας (image I/O packages) τα οποία μας επιτρέπουν να φορτώνουμε διάφορους τύπους αρχείων εικόνας, όπως JPEG, PNG, TIFF κλπ.

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

Μετά εγκαθιστούμε τα αντίστοιχα πακέτα για βίντεο (video I/O packages).

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

Εγκαθιστούμε το GTK development library, έτσι ώστε να μπορέσουμε να μεταγλωττίσουμε το highgui.

```
$ sudo apt-get install libgtk2.0-dev
```

Εγκαθιστούμε κάποιες επιπλέον εξαρτήσεις για εκτέλεση πράξεων εντός της OpenCV (όπως πράξεις πινάκων).

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

Τέλος εγκαθιστούμε τα Python 2.7 και Python 3 header files έτσι ώστε να μπορέσουμε να μεταγλωττίσουμε την OpenCV με διεπαφές της Python (OpenCV + Python bindings)

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

Βήμα #2: Λήψη του πηγαίου κώδικα (source code) της OpenCV

Σε αυτό το σημείο, έχοντας εγκαταστήσει όλα τα προαπαιτούμενα, ας κατεβάσουμε τη OpenCV 3.1.0 .

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

Για τη πλήρη εγκατάσταση της OpenCV 3 θα πρέπει να κατεβάσουμε και το opencv_contrib

(θα πρέπει οι εκδόσεις τόσο του opencv όσο και του opencv_contrib να είναι οι ίδιες αλλιώς θα υπάρξουν σφάλματα κατά τη μεταγλώττιση)

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

(οι δύο πρώτες γραμμές μπορούν να γραφτούν και στην ίδια σειρά του LXTerminal)

Βήμα #3: Ρύθμιση της Python

Το πρώτο πράγμα που πρέπει να κάνουμε για να ρυθμίσουμε τη Python για τη μεταγλώττιση της OpenCV είναι να εγκαταστήσουμε το pip

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

Η επόμενη εγκατάσταση που θα κάνουμε είναι προαιρετική και δεν αποτελεί προαπαιτούμενο προκειμένου να χρησιμοποιήσουμε την OpenCV στο Raspberry Pi. Πρόκειται για την εγκατάσταση του virtualenv και του virtualenvwrapper. Τα δυο αυτά πακέτα μας επιτρέπουν να τρέχουμε πολλαπλές εκδόσεις της Python, με διαφορετικές εκδόσεις πακέτων εγκατεστημένα σε κάθε εικονικό περιβάλλον, κάτι που αποτελεί βασική αρχή στη κοινωνία της Python.

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

Μετά την εγκατάσταση του virtualenv και του virtualenvwrapper θα ανοίξουμε το αρχείο .profile (βρίσκεται στο /home/pi) με την εντολή nano .profile και να προσθέσουμε τις παρακάτω γραμμές στο τέλος του αρχείου.

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

Αφού το αρχείο .profile ενημερώθηκε θα πρέπει να το επαναφορτώσουμε έτσι ώστε να λειτουργεί με τις νέες αλλαγές με την εντολή source.

```
$ source ~/.profile
```

Στη συνέχεια θα δημιουργήσουμε το εικονικό περιβάλλον της Python με το οποίο θα δουλέψουμε. Το περιβάλλον αυτό θα τρέχει Python 2 και θα έχει όνομα py2cv3.

```
$ mkvirtualenv py2cv3
```

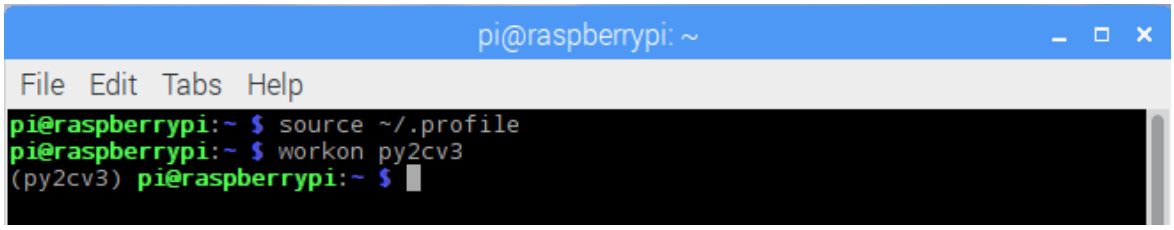
Σε αυτό το σημείο να τονίσουμε ότι το εικονικό περιβάλλον py2cv3 είναι εντελώς ανεξάρτητο από τα προ-εγκατεστημένα περιβάλλοντα που υπάρχουν στο Raspbian Jessie.

Κάθε φορά που κάνουμε αποσύνδεση, επανεκκίνηση ή ανοίγουμε ένα καινούριο τερματικό θα πρέπει να γράφουμε τις παρακάτω εντολές προκειμένου να έχουμε πρόσβαση στο εικονικό περιβάλλον cv, αλλιώς θα χρησιμοποιούμε την έκδοση Python του συστήματος.

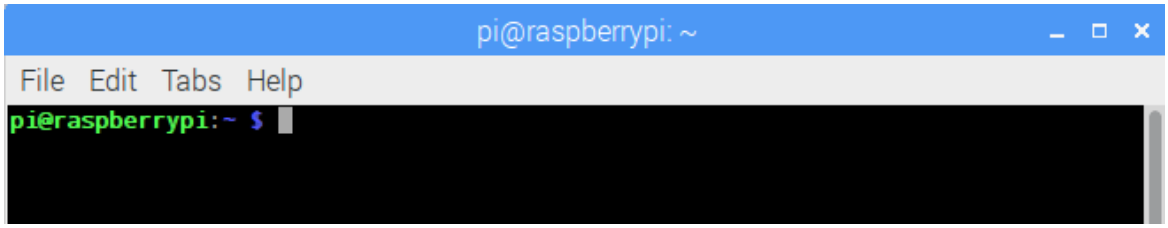
$ source ~/.profile

$ workon py2cv3

Μπορούμε να βεβαιωθούμε αν βρισκόμαστε στο εικονικό περιβάλλον cv ελέγχοντας τη γραμμή εντολών. Αν δούμε το κείμενο “(py2cv3)” σημαίνει ότι βρισκόμαστε στο εικονικό περιβάλλον.



Εικόνα Β.1: Βρισκόμαστε στο εικονικό περιβάλλον py2cv3



Εικόνα Β.2: Δεν βρισκόμαστε στο εικονικό περιβάλλον py2cv3

Βήμα #4: Μεταγλώττιση και εγκατάσταση της OpenCV

Σε αυτό το σημείο είμαστε έτοιμοι να μεταγλωττίσουμε την OpenCV.

Αρχικά βεβαιωνόμαστε ότι βρισκόμαστε στο εικονικό περιβάλλον py2cv3. Αν δεν είμαστε γράφουμε τις παρακάτω εντολές:

$ source ~/.profile

$ workon py2cv3

Στη συνέχεια κάνουμε τις απαραίτητες ρυθμίσεις γράφοντας τα παρακάτω:

$ cd ~/opencv-3.1.0/

$ mkdir build

$ cd build

$ cmake –D CMAKE_BUILD_TYPE=RELEASE \

-D CMAKE_INSTALL_PREFIX=/usr/local \

-D INSTALL_PYTHON_EXAMPLES=ON \

-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \

-D BUILD_EXAMPLES=ON ..

Πριν συνεχίσουμε, ελέγχουμε την έξοδο του CMake. Κατεβαίνουμε μέχρι να βρούμε το κομμάτι με τίτλο Python 2. Θα πρέπει το αποτέλεσμα να είναι ίδιο με το παρακάτω

Python 2:

Interpreter:	/usr/bin/python2.7 (ver 2.7.9)
Libraries:	/usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.9)
numpy:	/usr/lib/python2.7/dist-packages/numpy/core/include
package path	/lib/python2.7/dist-packages

Τώρα είμαστε έτοιμοι για τη μεταγλώττιση της OpenCV.
$ make -j4

Το -j4 δηλώνει τον αριθμό των πυρήνων που θα χρησιμοποιηθούν κατά τη μεταγλώττιση της OpenCV.

Το βήμα αυτό είναι το πιο χρονοβόρο της όλης διαδικασίας αφού η μεταγλώττιση της OpenCV με τέσσερις πυρήνες θα διαρκέσει μιάμιση ώρα.

Αν η εντολή make βγάλει σφάλματα, θα πρέπει να ξεκινήσουμε τη μεταγλώττιση από την αρχή, χρησιμοποιώντας μόνο ένα πυρήνα.
$ make clean
$ make

Χρησιμοποιώντας μόνο ένα πυρήνα, η μεταγλώττιση θα διαρκέσει περισσότερη ώρα αλλά ελαχιστοποιούνται οι πιθανότητες να προκύψει κάποιο σφάλμα.
Αφού ολοκληρωθεί η μεταγλώττιση χωρίς κάποιο σφάλμα, εγκαθιστούμε την OpenCV στο σύστημα μας.
$ sudo make install
$ sudo ldconfig

Βήμα #5: Τελειώνοντας την εγκατάσταση
Η OpenCV θα πρέπει να έχει εγκατασταθεί στο:
/usr/local/lib/python2.7/dist-packages

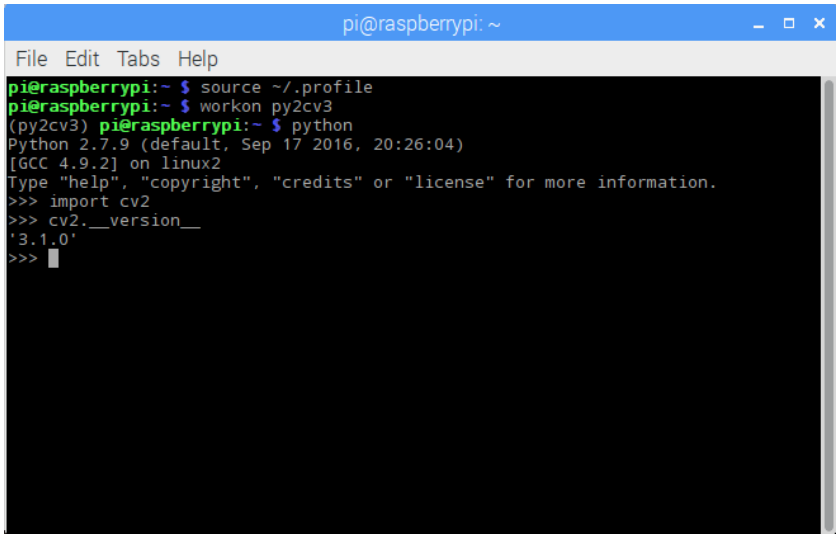
Ας το ελέγξουμε με τη παρακάτω εντολή:
$ ls /usr/local/lib/python2.7/dist-packages/

Αν έχει εγκατασταθεί εκεί θα εμφανιστεί ένα αρχείο με όνομα cv2.so
Αν δεν έχει εγκατασταθεί εκεί ελέγχουμε στο:
/usr/local/lib/python2.7/site-packages

Τέλος θα φτιάξουμε ένα συμβολικό σύνδεσμο του cv2.so στο εικονικό περιβάλλον py2cv3.

$ cd ~/.virtualenvs/py2cv3/lib/python2.7/dist-packages/
$ ln -s /usr/local/lib/python2.7/dist-packages/cv2.so cv2.so

Βήμα #6: Επαλήθευση της εγκατάστασης της OpenCV
Σε αυτό το σημείο, η OpenCV θα πρέπει να έχει εγκατασταθεί στο Raspberry Pi μας. Ας ελέγξουμε ότι όντως η OpenCV δουλεύει έχοντας πρόσβαση στο εικονικό περιβάλλον cv:
$ source ~/.profile
$ workon py2cv3
$ python
>>> import cv2
>>> cv2.__version__



Εικόνα Β.3: Επαλήθευση της εγκατάστασης της OpenCV στο εικονικό περιβάλλον py2cv3

Στη παραπάνω εικόνα, βλέπουμε ότι το εικονικό περιβάλλον py2cv3 τρέχει Python 2 και πως η OpenCV έχει εγκατασταθεί επιτυχώς.

Β.2 Εγκατάσταση απαραίτητων βιβλιοθηκών

Στην ενότητα αυτή παρουσιάζουμε το σωστό τρόπο εγκατάστασης βασικών βιβλιοθηκών επεξεργασίας εικόνας, αναγνώρισης προτύπων και μηχανικής μάθησης που χρησιμοποιήθηκαν στη παρούσα εργασία.

Βήμα #1: Σύνδεση στο εικονικό περιβάλλον

Πριν ξεκινήσουμε πρέπει να συνδεθούμε στο εικονικό περιβάλλον στο οποίο θέλουμε να δουλέψουμε με τις συγκεκριμένες βιβλιοθήκες. Επομένως γράφουμε στο LXTerminal τις εξής εντολές:

$ source ~/.profile

$ workon py2cv3

Βήμα #2: Εγκατάσταση των βιβλιοθηκών

Αφού συνδεθούμε στο εικονικό περιβάλλον εγκαθιστούμε τις απαραίτητες βιβλιοθήκες γράφοντας τις εξής εντολές στο LXTerminal:

$ pip install numpy

$ pip install scipy

$ pip install matplotlib

$ pip install scikit-learn
$ pip install scikit-image
$ pip install pillow
$ pip install h5py
$ pip install imutils
$ pip install argparse
$ pip install "picamera[array]"

Β.3 Εγκατάσταση της βιβλιοθήκης Keras στο RPi 3 Model B

Στην ενότητα αυτή παρουσιάζεται βήμα-βήμα η εγκατάσταση της βιβλιοθήκης Keras στο RPi 3 Model B.

Βήμα #1: Σύνδεση στο εικονικό περιβάλλον
Πριν ξεκινήσουμε την εγκατάσταση πρέπει να συνδεθούμε στο εικονικό περιβάλλον στο οποίο θέλουμε να δουλέψουμε με τη βιβλιοθήκη Keras. Επομένως γράφουμε στο LXTerminal τις εξής εντολές:
$ source ~/.profile
$ workon py2cv3

Βήμα #2: Εγκατάσταση της βιβλιοθήκης Keras
Πριν εγκαταστήσουμε την βιβλιοθήκη Keras θα πρέπει να εγκαταστήσουμε κάποιες εξαρτήσεις (εάν δεν έχουν εγκατασταθεί παλαιότερα):
$ pip install numpy
$ pip install scipy
$ pip install scikit-learn
$ pip install pillow
$ pip install h5py

Επίσης, πρέπει να εγκαταστήσουμε τη βιβλιοθήκη Theano. Η Theano είναι μια βιβλιοθήκη για τη γλώσσα Python η οποία επιτρέπει τον ορισμό και την αποτελεσματική βελτιστοποίηση και αξιολόγηση μαθηματικών εκφράσεων που περιλαμβάνουν πολυδιάστατους πίνακες.
$ pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git

Αφού εγκατασταθούν οι απαραίτητες εξαρτήσεις μπορούμε να εγκαταστήσουμε τη βιβλιοθήκη Keras:
$ pip install keras

Βήμα #3: Τροποποίηση του αρχείου ρυθμίσεων keras.json
Τώρα θα πρέπει να τροποποιήσουμε το αρχείο ρυθμίσεων keras.json έτσι ώστε να ορίσουμε ως backend τη βιβλιοθήκη Theano. Αρχικά, όντας εντός του εικονικού περιβάλλοντος, τρέχουμε τις εξής εντολές:
>>> import keras
>>> quit()
Με τον τρόπο αυτό θα δημιουργηθεί το αρχείο ρυθμίσεων το οποίο θα βρίσκεται στο: ~/.keras/keras.json
Ανοίγουμε το αρχείο ως εξής:
nano ~/.keras/keras.json
το αρχείο θα πρέπει να έχει αυτή τη μορφή:

```
{
    "epsilon": 1e-07,
    "floatx": "float32",
    "image_data_format": "channels_last",
    "backend": "tensorflow"
}
```

Αλλάζουμε τη τιμή του "image_data_format" σε "channels_first" και τη τιμή του "backend" σε "theano". Αποθηκεύουμε και κλείνουμε το αρχείο

Η εγκατάσταση έχει ολοκληρωθεί. Μπορούμε να συνδεθούμε στο εικονικό περιβάλλον και να κάνουμε import τη βιβλιοθήκη έτσι ώστε να ελέγξουμε για τυχόν σφάλματα.

Β.4 Εγκατάσταση της βιβλιοθήκης Dlib στο RPi 3 Model B

Στην ενότητα αυτή παρουσιάζεται βήμα-βήμα η εγκατάσταση της βιβλιοθήκης Dlib στο RPi 3 Model B.

Η Dlib είναι μια βιβλιοθήκη που απαιτεί αρκετούς υπολογιστικούς πόρους κατά την εγκατάσταση της, κάτι που το RPi δε μπορεί να προσφέρει αυτή τη στιγμή. Αν επιχειρήσουμε να εγκαταστήσουμε τη Dlib θα προκύψει σφάλμα λόγω ανεπάρκειας μνήμης και η εγκατάσταση θα σταματήσει. Επομένως, πριν την εγκατάσταση της dlib θα πρέπει να αυξήσουμε τη διαθέσιμη μνήμη για τη μεταγλώττιση της βιβλιοθήκης.

Βήμα #1: Προετοιμασία του RPi για τη μεταγλώττιση της Dlib

Για τη προετοιμασία του RPi για τη μεταγλώττιση της Dlib απαιτούνται οι εξής τρείς ρυθμίσεις:

Αύξηση μεγέθους του swap file

Το αρχείο ρυθμίσεων του swap file βρίσκεται στο: /etc/dphys-swapfile. Ανοίγουμε το αρχείο με την εντολή: $ sudo nano /etc/dphys-swapfile και τροποποιούμε τη τιμή του CONF_SWAPSIZE από 100 σε 1024.

Μετά εκτελούμε τις εξής δυο εντολές για την επανεκκίνηση του swap service:

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

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

Αλλαγή των ρυθμίσεων εκκίνησης

Εκτελούμε την εντολή: $ sudo raspi-config

Και επιλέγουμε: Boot Options => Desktop / CLI => Console Autologin

Ενημέρωση της μνήμης της GPU

Εξ' ορισμού, το RPi διαθέτει 128MB μνήμης για τη GPU. Θα μειώσουμε τον αριθμό αυτό σε 16MB.

Εκτελούμε την εντολή: $ sudo raspi-config

Και επιλέγουμε: Boot Options => Memory Split

και μετατρέπουμε το 128 σε 16

Μόλις ολοκληρώσουμε τις ρυθμίσεις, κάνουμε επανεκκίνηση.

Βήμα #2: Εγκατάσταση εξαρτήσεων

Η βιβλιοθήκη Dlib απαιτεί τέσσερις εξαρτήσεις:

1. Boost
2. Boost.Python
3. CMake
4. X11

Αυτές μπορούν να εγκατασταθούν με τις εξής εντολές:

$ sudo apt-get update

$ sudo apt-get install build-essential cmake

$ sudo apt-get install libgtk-3-dev

$ sudo apt-get install libboost-all-dev

Βήμα #3: Σύνδεση στο εικονικό περιβάλλον

Πριν ξεκινήσουμε την εγκατάσταση πρέπει να συνδεθούμε στο εικονικό περιβάλλον στο οποίο θέλουμε να δουλέψουμε με τη βιβλιοθήκη Keras. Επομένως γράφουμε στο LXTerminal τις εξής εντολές:

$ source ~/.profile

$ workon py2cv3

Βήμα #4: Εγκατάσταση της βιβλιοθήκης Dlib

Πριν εγκαταστήσουμε τη Dlib εγκαθιστούμε τις βιβλιοθήκες NumPy, Scipy και scikit-image (εάν δεν έχουν εγκατασταθεί παλαιότερα):

$ pip install numpy

$ pip install scipy

$ pip install scikit-image

Αφού ολοκληρωθεί η εγκατάσταση μπορούμε να εγκαταστήσουμε τη Dlib:

$ pip install dlib

Αφού ολοκληρωθεί η εγκατάσταση, μπορούμε να συνδεθούμε στο εικονικό περιβάλλον και να κάνουμε import τη βιβλιοθήκη έτσι ώστε να ελέγξουμε για τυχόν σφάλματα.

Βήμα #5: Επαναφορά των ρυθμίσεων μνήμης

Είναι σημαντικό να επαναφέρουμε τις ρυθμίσεις που έγιναν στο **Βήμα #1** στην αρχική τους κατάσταση. Αυτό μπορεί να γίνει ακολουθώντας τις οδηγίες του **Βήμα #1**. Αφού επαναφέρουμε τις ρυθμίσεις αυτές κάνουμε επανεκκίνηση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία:

[1] Ι.Ν. Έλληνας, "ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ & ΒΙΝΤΕΟ: Από τη Θεωρία στη Πράξη", 2010

[2] Rafael C. Gonzales, Richard E. Woods, "Digital Image Processing 3rd edition", 2008

[3] Gary Bradski and Adrian Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library", 2008

[4] Jan Erik Solem, "Programming Computer Vision with Python", 2012

[5] Joseph Howse, "OpenCV Computer Vision with Python", 2013

[6] Simon Monk, "Raspberry Pi Cookbook", 2014

[7] Κ. Μαγκούτης, Χ. Νικολάου, "ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ PYTHON", 2015

[8] ΜΑΝΗΣ ΓΕΩΡΓΙΟΣ, "ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ ΑΡΩΓΟ ΤΗ ΓΛΩΣΣΑ PYTHON", 2015

Εγχειρίδια:

[9] Αθανασία Κολοβού (Ε,Τ.Ε.Π), "ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ: ΠΑΡΑΔΕΙΓΜΑΤΑ ΜΕ ΧΡΗΣΗ MATLAB", Πανεπιστήμιο Αθηνών, 2012

[10] Alexander Mordvintsev & Abid K, "OpenCV-Python Tutorials Documentation Release 1", 2017

[11] Adrian Rosebrock, "Image Search Engine Resource Guide", 2016

Πτυχιακές εργασίες:

[12] Σταμάτιος Αρμένης, "Οπτική αναγνώριση χαρακτήρων με συσκευή Android", Πτυχιακή εργασία, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων, ΑΕΙ Πειραιά ΤΤ, 2014

[13] Ιωάννης Πρωτονοτάριος, "Ανάπτυξη συστήματος ενσωματωμένων αισθητήρων για ασύρματη μετάδοση εικόνας και δεδομένων", Πτυχιακή εργασία, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών, 2011

[14] Παπαβασιλείου Κατερίνα, Φραγγεδάκη Γεωργία, "Στατιστική ανάλυση των δεδομένων της Γραμμικής Α σε σχέση με τη Γραμμική Β με χρήση της αναγνώρισης προτύπων", Πτυχιακή εργασία, Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων, ΤΕΙ Κρήτης, 2010

[15] Επαμεινώνδας Π. Αντωνάκος, "Οπτική Μοντελοποίηση Ανθρώπινου Προσώπου με Εφαρμογές σε Αναγνώριση", Πτυχιακή εργασία, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστικών, Εθνικό Μετσόβιο Πολυτεχνείο, 2011

Ιστοσελίδες:

[16] Wikipedia Επεξεργασία Εικόνας: https://en.wikipedia.org/wiki/Digital_image_processing

[17] Επεξεργασία εικόνας e-courses: http://eclass.teipir.gr/openeclass/courses/HYS107/

[18] Wikipedia Raspberry Pi: https://en.wikipedia.org/wiki/Raspberry_Pi

[19] Raspberry Pi web page: https://www.raspberrypi.org

[20] What is a Raspberry Pi: https://www.raspberrypi.org/help/faqs/#introWhatIs

[21] Raspberry Pi Uses: http://www.pcworld.com/article/3043022/computers/10-surprisingly-practical-raspberry-pi-projects-anybody-can-do.html

[22] Wikipedia OpenCV: https://en.wikipedia.org/wiki/OpenCV#cite_ref-6

[23] OpenCV.org web page: http://opencv.org

[24] About OpenCV: http://opencv.org/about.html

[25] Introduction to OpenCV – Python Tutorials: http://docs.opencv.org/3.1.0/d0/de3/tutorial_py_intro.html

[26] Learning OpenCV: https://www.safaribooksonline.com/library/view/learning-opencv-3/9781491937983/titlepage01.html

[27] Wikipedia Αναγνώριση Προτύπων: https://el.wikipedia.org/wiki/Αναγνώριση_προτύπων

[28] Σύστημα Αναγνώρισης Προτύπων: http://www.icsd.aegean.gr/lecturers/kavallieratou/PattRec_files/2%20Basics.pdf

[29] Τι είναι η Αναγνώριση Προτύπων: http://www.cs.uoi.gr/~kblekas/courses/PR/

[30] Wikipedia Python: https://en.wikipedia.org/wiki/Python_(programming_language)

[31] Wikipedia Dlib: https://en.wikipedia.org/wiki/Dlib

[32] Dlib: http://dlib.net/intro.html

[33] Wikipedia SciPy: https://en.wikipedia.org/wiki/SciPy

[34] Wikipedia Matplotlib: https://en.wikipedia.org/wiki/Matplotlib

[35] Wikipedia scikit-learn: https://en.wikipedia.org/wiki/Scikit-learn

[36] Wikipedia scikit-image: https://en.wikipedia.org/wiki/Scikit-image

[37] Wikipedia Keras: https://en.wikipedia.org/wiki/Keras

[38] OpenCV Installation: http://www.pyimagesearch.com/2015/10/26/how-to-install-opencv-3-on-raspbian-jessie/

[39] Keras Installation: http://www.pyimagesearch.com/2016/07/18/installing-keras-for-deep-learning/

[40] Dlib Installation: http://www.pyimagesearch.com/2017/05/01/install-dlib-raspberry-pi/

[41] Installing operating system images: https://www.raspberrypi.org/documentation/installation/installing-images/README.md

[42] imutils: http://www.pyimagesearch.com/2015/02/02/just-open-sourced-personal-imutils-package-series-opencv-convenience-functions/

[43] Wikipedia KNN: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[44] Wikipedia HAAR: https://en.wikipedia.org/wiki/Viola–Jones_object_detection_framework

[45] HAAR: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[46] Wikipedia HOG: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

[47] Wikipedia LBP: https://en.wikipedia.org/wiki/Local_binary_patterns

[48] Wikipedia Lenet: https://en.wikipedia.org/wiki/Convolutional_neural_network#LeNet-5

[49] LeNet: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

[50] Facial Landmarks: http://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

[51] Python Picamera: https://www.raspberrypi.org/documentation/usage/camera/python/README.md

[52] Accessing the Raspberry Pi Camera with OpenCV and Python: http://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/

[53] Using a standard usb webcam: https://www.raspberrypi.org/documentation/usage/webcams/