
**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ**

**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

# Μελέτη μεθόδων ανίχνευσης ανθρώπινης μορφής

Διπλωματική Εργασία

Κωνσταντίνος Μιχαλακάκης

Επιβλέπουσα: Παναγιώτα Τσομπανοπούλου

Βόλος 2021


**ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ**

**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

**ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

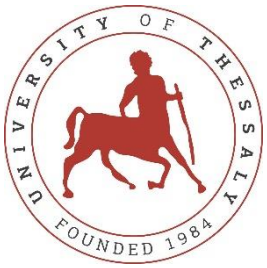
# Μελέτη μεθόδων ανίχνευσης ανθρώπινης μορφής

Διπλωματική Εργασία

Κωνσταντίνος Μιχαλακάκης

Επιβλέπουσα: Παναγιώτα Τσομπανοπούλου

Βόλος 2021




**UNIVERSITY OF THESSALY**

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING**

# Study of methods for human figure detection

Diploma Thesis

Konstantinos Michalakakis

Supervisors: Panagiota Tsompanopoulou

Volos 2021

Εγκρίνεται από την Επιτροπή Εξέτασης:

| | |
|---|---|
| Επιβλέπων | **Παναγιώτα Τσομπανοπούλου**<br>Αναπληρώτρια Καθηγήτρια, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας |
| Μέλος | **Ελευθέριος Τσουκαλάς**<br>Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας |
| Μέλος | **Δημήτριος Μπαργιώτας**<br>Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας |

Ημερομηνία έγκρισης: 12-02-2021

## ΕΥΧΑΡΙΣΤΙΕΣ

*Θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου για την στήριξη τους καθώς και την κ.Τσομπανοπούλου για την πολύτιμη βοήθεια και καθοδήγηση που μου παρείχε.*

**ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ**

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών

(Υπογραφή)
Κωνσταντίνος Μιχαλακάκης
12/02/2021

# ΠΕΡΙΛΗΨΗ

Η Τεχνητή Νοημοσύνη είναι ένα επιστημονικό πεδίο που γνωρίζει μεγάλη ανάπτυξη τις τελευταίες δεκαετίες. Ένας τομέας της Τεχνητής Νοημοσύνης είναι η Υπολογιστική Όραση, οι εφαρμογές της οποίας απαντούν ένα μεγάλο σύνολο προβλημάτων στην σημερινή εποχή. Πιο συγκεκριμένα, η αναγνώριση εικόνας χρησιμοποιείται ευρέως σε πολλούς τομείς της τεχνολογίας και της επιστήμης, για αυτό το λόγο, οι αλγόριθμοι που σχετίζονται με την ανίχνευση αντικειμένων σε εικόνες ή βίντεο αποτελούν ένα μεγάλο πεδίο μελέτης από πολλούς επιστήμονες και μηχανικούς παγκοσμίως.

Στην παρούσα διπλωματική εργασία μελετήθηκαν δύο από τους πιο γνωστούς αλγόριθμους αναγνώρισης αντικειμένων. Αυτοί ήταν ο αλγόριθμος των Viola και Jones που χρησιμοποιεί τα χαρακτηριστικά τύπου Haar και έναν αλγόριθμο AdaBoost για ταξινόμηση και ο αλγόριθμος των Dalal και Triggs που χρησιμοποιεί ένα ιστόγραμμα κατευθυνόμενων κλίσεων σε συνδυασμό με ένα Support Vector Machine για την ταξινόμηση των εικόνων. Η υλοποίηση έγινε σε Python με χρήση της βιβλιοθήκης OpenCV. Οι δύο αυτοί αλγόριθμοι δοκιμάστηκαν σε ένα σύνολο δεδομένων, το INRIA Pedestrian Dataset, και μελετήθηκε η συμπεριφορά και η απόδοση τους για διάφορες τιμές των παραμέτρων τους.

Τέλος παρουσιάστηκαν τα αποτελέσματα της σύγκρισης αυτής καθώς και τα διάφορα συμπεράσματα και οι παρατηρήσεις που προέκυψαν κατά τη διάρκεια αυτής της διαδικασίας.

# ABSTRACT

Artificial Intelligence is a scientific field that is growing rapidly in the last few decades. Computer Vision is part of Artificial Intelligence and is used to provide solutions to a variety of problems. More specifically, Image Recognition is frequently used for scientific or commercial purposes thus a lot of scientists and engineers study and develop algorithms related to object detection.

In this diploma thesis we compare two of the most popular object detection algorithms. These are the one proposed by Viola and Jones, which uses Haar futures combined with an AdaBoost algorithm for classification and the one proposed by Dalal and Triggs, which uses a Histogram of Oriented Gradients and a Support Vector Machine for classification. We implemented these algorithms using the OpenCV library in Python. These two methods were tested on the INRIA Pedestrian Dataset. We observed how these algorithms behave on various occasions and how efficient they were in detecting humans.

Finally, we summarize the conclusions of this comparison and we present the results of our tests.

# ΠΕΡΙΕΧΟΜΕΝΑ

# ΚΕΦΑΛΑΙΟ 1: Εισαγωγή

## 1.1 Γενικά

Ο άνθρωπος είναι μια από τις πιο σημαντικές συνιστώσες στο περιβάλλον μιας μηχανής, συνεπώς η αλληλεπίδραση του με αυτή αποτελεί ένα ευρύ πεδίο μελέτης στην σύγχρονη μηχανική. Υπάρχει ένας τομέας της πληροφορικής που ως κύριο αντικείμενο του έχει την δημιουργία υπολογιστικών συστημάτων που μιμούνται ανθρώπινες συμπεριφορές και διαδικασίες [1]. Αυτός είναι ο κλάδος της Τεχνητής Νοημοσύνης. Η ανίχνευση αντικειμένων είναι μια φυσική διαδικασία που γίνεται πολλές φορές από τον άνθρωπο και η αυτοματοποίηση της από τον υπολογιστή είναι ένα πρόβλημα της Υπολογιστικής Όρασης.

Η Υπολογιστική Όραση (Computer Vision) είναι ένας τομέας της Τεχνητής Νοημοσύνης (Artificial Intelligence) που ασχολείται με την ανάλυση εικόνων και την εξαγωγή πληροφορίας από αυτές. Είναι ένας τρόπος να κατανοήσει ένας υπολογιστής τον κόσμο μέσα από ψηφιακά πολυμέσα, όπως είναι εικόνες ή βίντεο. Ένα πρόβλημα που καλείται συχνά να λύσει η υπολογιστική όραση είναι το αν υπάρχει κάποιο συγκεκριμένο αντικείμενο σε μία εικόνα. Ένα σύστημα αναγνώρισης εικόνας συνήθως δέχεται ως είσοδο μία εικόνα και επιστρέφει την θέση του αντικειμένου προς αναζήτηση μέσα στην εικόνα αυτή. Ένα από τα αντικείμενα, η αναγνώριση του οποίου αποτελεί πολύ συχνά αντικείμενο μελέτης στον τομέα αυτό, είναι ο άνθρωπος.

## 1.2 Βιβλιογραφική Ανασκόπηση

Η ανίχνευση αντικειμένων προσέλκυσε το ενδιαφέρον πολλών ερευνητών στο πέρασμα των χρόνων. Υπάρχει λοιπόν μια ποικιλία στις κατηγορίες και στα είδη των ανιχνευτών. Το 1987 ο Förstner και ο Gülch ασχολήθηκαν με την ανίχνευση διάσπαρτων διακριτών σημείων και παρουσίασαν έναν ανιχνευτή που εντόπιζε τις ακμές και τις γωνίες των αντικειμένων που εμφανίζονταν σε μία εικόνα [2]. Επίσης το 1987, οι Sirovich και Kirby χρησιμοποιούν τις τιμές των εντάσεων των εικονοστοιχείων (pixels) για να εξάγουν πληροφορίες σχετικά με το περιεχόμενο μιας εικόνας [3]. Το 1997 οι Forsyth και Fleck προτείνουν έναν ανιχνευτή για ανθρώπους, που αρχικά ανιχνεύει τα μέλη του σώματος

όπως είναι τα χέρια, τα πόδια, ο κορμός κλπ., κι έπειτα χρησιμοποιεί τις γεωμετρικές σχέσεις των μελών αυτών καθώς επίσης και πληροφορίες που σχετίζονται με το χρώμα και την υφή για την ανίχνευση ολόκληρου του ανθρώπου [4]. Το 1999 ο Gavrila και ο Philomin βασίζονται στο σχήμα του ανθρώπινου σώματος και δημιουργούν έναν ανιχνευτή ο οποίος συγκρίνει το σχήμα ενός αντικειμένου που εντόπισε σε μια εικόνα με ένα σύνολο από πρότυπα, για να αποφασίσει αν το εν λόγω σχήμα αντιστοιχεί σε ανθρώπινη φιγούρα ή όχι [5].

## 1.3 Εφαρμογές και Προβλήματα

Τα τελευταία χρόνια υπήρξε μεγάλη πρόοδος στις συσκευές λήψης εικόνας και βίντεο καθώς μειώθηκε το μέγεθος των συσκευών αυτών, ενώ ταυτόχρονα βελτιώθηκε η ποιότητα του υλικού που αυτές οι συσκευές μπορούν να καταγράψουν. Αυτό οδήγησε στην όλο και πιο συχνή χρήση τους με αποτέλεσμα να αυξηθεί το ενδιαφέρον και για τα συστήματα αναγνώρισης εικόνας. Δύο μεγάλοι τομείς στους οποίους τα συστήματα αυτά και συγκεκριμένα τα συστήματα αναγνώρισης του ανθρώπου έχουν μεγάλη χρησιμότητα, είναι τα συστήματα παρακολούθησης και η αυτοκινητοβιομηχανία.

Τα κλειστά κυκλώματα παρακολούθησης (CCTV) είναι ένα από τα πιο δημοφιλή συστήματα ασφαλείας τις τελευταίες δεκαετίες και χρησιμοποιείται για την καταγραφή συμβάντων σε δημόσιους ή ιδιωτικούς χώρους. Η ουσία της χρήσης τέτοιων συστημάτων ασφαλείας είναι η πρόληψη και η αντιμετώπιση απειλών. Το γεγονός όμως ότι η παρακολούθηση της πληροφορίας που αυτές οι συσκευές καταγράφουν γίνεται από ανθρώπους, πολλές φορές τα καθιστά ευάλωτα. Η αυτοματοποίηση της διαδικασίας αυτής με κάποιο σύστημα αναγνώρισης εικόνας διευκολύνει το έργο της παρακολούθησης, πόσο μάλλον σε μεγάλους χώρους, όπως είναι ένα αεροδρόμιο, όπου υπάρχει μεγάλη ροή εισερχόμενης πληροφορίας από τις κάμερες [6].

Στο τομέα της αυτοκινητοβιομηχανίας, ένα σύστημα αναγνώρισης εικόνας θα μπορούσε να μειώσει τον αριθμό των τροχαίων ατυχημάτων. Στις Ηνωμένες Πολιτείες της Αμερικής στο 15% των θανατηφόρων ατυχημάτων εμπλέκονται πεζοί [7]. Τα αντίστοιχα ποσοστά για την Ευρωπαϊκή Ένωση και την Ιαπωνία είναι 18% και 32% [8]. Η αιτία αυτών των ατυχημάτων πολλές φορές οφείλεται στην έλλειψη προσοχής του οδηγού. Ένα

σύστημα αναγνώρισης πεζών θα μπορούσε να προειδοποιεί τον οδηγό ή να αναλαμβάνει αυτόματα τον έλεγχο του οχήματος για την αποφυγή μιας σύγκρουσης [8].

Επιπλέον εφαρμογές είναι η αυτόματη παρακολούθηση ηλικιωμένων ή παιδιών καθώς και η διαιτησία αθλημάτων όπου η ανθρώπινη κρίση δεν αρκεί πολλές φορές για ένα δίκαιο αποτέλεσμα. Τέλος, η αυτόματη ανάλυση περιεχομένου στον τεράστιο όγκο βίντεο που υπάρχει στο διαδίκτυο, βοηθάει στην καλύτερη ταξινόμησή τους και στην ευκολότερη πλοήγηση από τους χρήστες [9].

Παρόλα αυτά, για να χρησιμοποιηθεί ένα τέτοιο σύστημα, υπάρχουν κάποια ζητήματα που πρέπει να ληφθούν υπόψη. Πρέπει καταρχήν να είναι αξιόπιστο και να έχει μικρά ποσοστά σφάλματος για να μπορεί να χρησιμοποιηθεί αυτόνομα χωρίς να εμπλακεί ο ανθρώπινος παράγοντας. Η ταχύτητα έχει επίσης ιδιαίτερη σημασία αν πρόκειται για εφαρμογές πραγματικού χρόνου.

Από την άλλη, υπάρχουν κάποιες παράμετροι που σχετίζονται με τους ανθρώπους, το περιβάλλον και τα τεχνικά χαρακτηριστικά στις κάμερες, οι οποίες μπορούν να δυσκολέψουν το έργο της ανίχνευσης. Κάθε άνθρωπος έχει διαφορετικό μέγεθος, χρώμα και σχήμα. Επίσης ένας άνθρωπος μπορεί να βρίσκεται σε πολλά διαφορετικά περιβάλλοντα όπου επικρατούν διαφορετικές συνθήκες φωτεινότητας. Τέλος η κάμερα μπορεί να είναι σταθερή ή σε κίνηση και η ποιότητα του αισθητήρα ή του φακού να προκαλεί θόρυβο. Ένας αλγόριθμος αναγνώρισης λοιπόν, για να είναι αποδοτικός, θα πρέπει να λάβει υπόψη του όλες τις παραπάνω συνθήκες.

## 1.4 Διάρθρωση της εργασίας

Στην παρούσα διπλωματική εργασία έγινε μια σύγκριση μεταξύ δύο από τους πιο δημοφιλείς αλγορίθμους στην αναγνώριση ανθρώπων. Αυτοί είναι ο αλγόριθμος των Dalal και Triggs [10] και ο αλγόριθμος των Viola και Jones [11]. Ο κύριος λόγος που επιλέχθηκαν αυτοί οι δύο αλγόριθμοι είναι η ομοιότητα που παρουσιάζουν στην δομή τους [7]. Αρχικά και οι δύο βασίζονται στην μέθοδο των sliding-windows, δηλαδή και στους δύο αλγορίθμους ένα παράθυρο ολισθαίνει κατά μήκος της εικόνας, σκανάροντας την, αναζητώντας το εκάστοτε αντικείμενο προς ανίχνευση. Έπειτα η ομοιότητα τους είναι

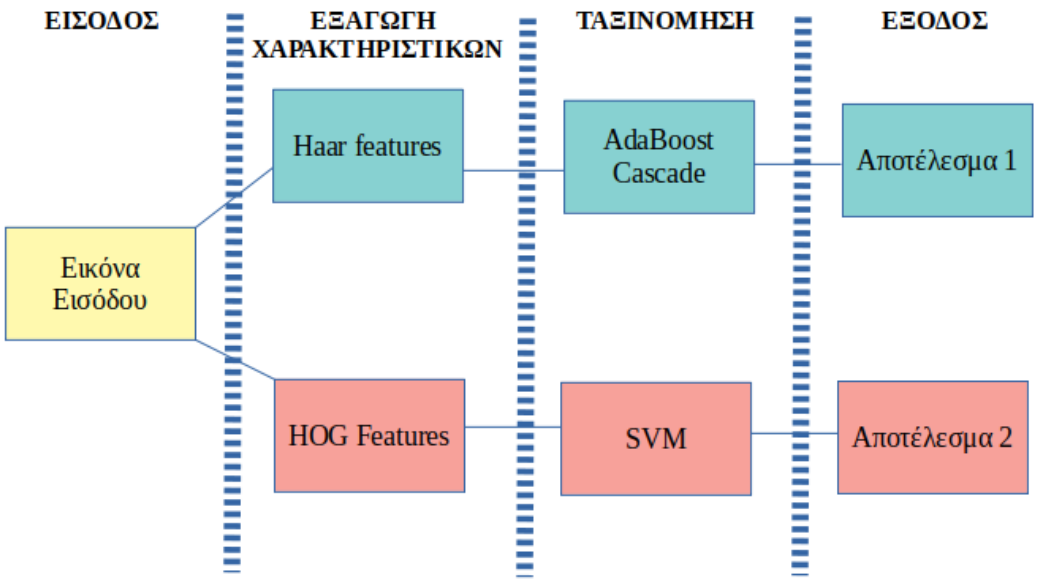
εμφανής και στην ανατομία τους καθώς και οι δυο μέθοδοι αποτελούνται από δύο κύρια στάδια, ένα στάδιο εξαγωγής χαρακτηριστικών κι ένα στάδιο ταξινόμησης.

Η δομή της εργασίας είναι η εξής: Στο Κεφάλαιο 2 παρουσιάζονται βασικές έννοιες και περιγράφεται ο τρόπος με τον οποίο οι αλγόριθμοι λειτουργούν σε θεωρητικό επίπεδο. Στο Κεφάλαιο 3 γίνεται αναφορά στα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση των αλγορίθμων. Στο Κεφάλαιο 4 περιγράφεται η πειραματική διαδικασία και παρουσιάζονται τα αποτελέσματα της σύγκρισης των δύο αλγορίθμων, καθώς και οι παρατηρήσεις που προέκυψαν από την διαδικασία αυτή. Στο Κεφάλαιο 5 γίνεται η σύνοψη των συμπερασμάτων και μία αναφορά σε μελλοντικούς στόχους. Τέλος, στο Παράρτημα Α παρατίθεται ο κώδικας της εργασίας.

# ΚΕΦΑΛΑΙΟ 2: Περιγραφή μεθόδων

## 2.1 Αρχιτεκτονική

Τα περισσότερα συστήματα αναγνώρισης αντικειμένων έχουν κοινή δομή. Πρώτα υπάρχει ένα στάδιο εξαγωγής χαρακτηριστικών και ακολουθεί ένα στάδιο ταξινόμησης (classification) [12]. Τα χαρακτηριστικά που εξάγονται στο πρώτο στάδιο χρησιμοποιούνται ως είσοδος σε έναν ταξινομητή (classifier) που βρίσκεται στο δεύτερο στάδιο. Η έξοδος του ταξινομητή είναι η απόφαση για το αν υπάρχει ή όχι το αντικείμενο ενδιαφέροντος στην εικόνα. Μια τέτοια αρχιτεκτονική παρουσιάζεται στην Εικόνα 2.1.




***Εικόνα 2.1:*** *Η βασική δομή ενός συστήματος αναγνώρισης αντικειμένων.*

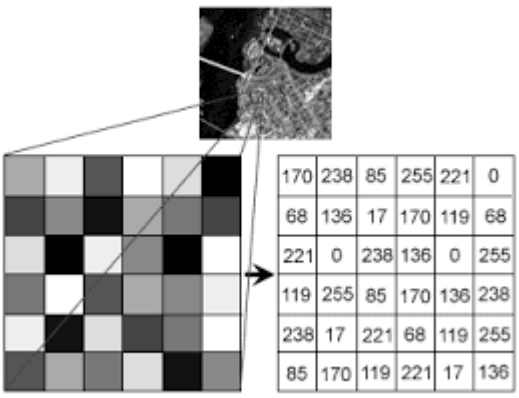
## 2.2 Εξαγωγή Χαρακτηριστικών

Κάθε εικόνα στον υπολογιστή έχει την μορφή ενός ή περισσότερων πινάκων. Μία ασπρόμαυρη εικόνα, όπως φαίνεται στην Εικόνα 2.2, αναπαρίσταται σαν ένας πίνακας, όπου κάθε κελί περιέχει μια τιμή που αντιπροσωπεύει την τιμή του εικονοστοιχείου (pixel)

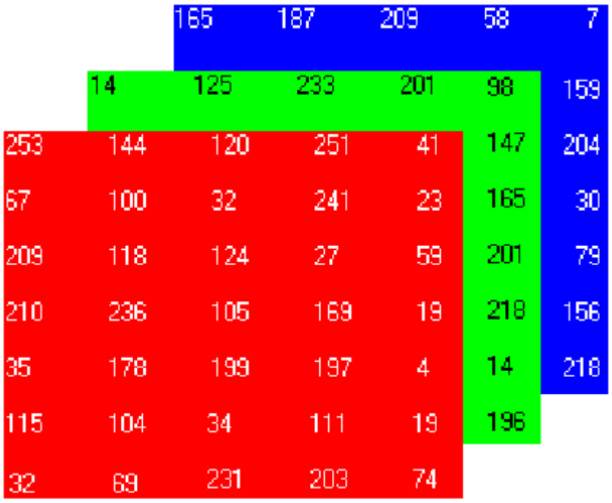
στην αντίστοιχη θέση της εικόνας [13]. Για την αναπαράσταση μιας έγχρωμης RGB εικόνας, χρησιμοποιούνται τρεις τέτοιοι πίνακες, ένας πίνακας για κάθε κανάλι χρώματος, όπως φαίνεται στην Εικόνα 2.3. Σε κάθε εικονοστοιχείο δεν αντιστοιχεί μια τιμή, αλλά ένα διάνυσμα τριών θέσεων, το οποίο παίρνει μία τιμή από κάθε πίνακα όπως δείχνει η Εικόνα 2.4. Είναι εμφανές ότι η πληροφορία που περιέχει μια εικόνα, όταν βρίσκεται σε αυτή την μορφή, είναι δύσκολο να αξιοποιηθεί.

Το στάδιο της εξαγωγής χαρακτηριστικών έχει αυτόν ακριβώς τον σκοπό, να εξάγει δηλαδή την χρήσιμη πληροφορία από μια εικόνα. Το αποτέλεσμα του σταδίου αυτού είναι ένα σύνολο χαρακτηριστικών. Όσο περισσότερο σχετίζονται τα χαρακτηριστικά αυτά με το αντικείμενο προς αναζήτηση τόσο πιο αποδοτικός είναι ο ανιχνευτής, για αυτό συχνά το στάδιο αυτό έχει μεγάλη σημασία [13].

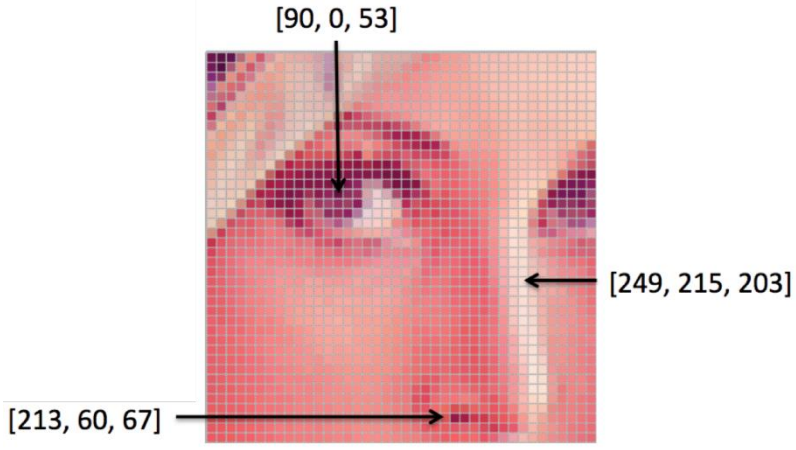
Υπάρχουν δύο βασικές κατηγορίες χαρακτηριστικών [14]. Η μία κατηγορία αφορά χαρακτηριστικά που παράγονται από κάποια σημεία ή τμήματα της εικόνας, παρέχοντας έτσι μια διάσπαρτη αναπαράσταση του περιεχομένου της. Η άλλη κατηγορία προσφέρει μια πιο πυκνή αναπαράσταση, βασιζόμενη στην εξαγωγή χαρακτηριστικών από όλη την εικόνα. Τα δύο είδη χαρακτηριστικών που εξετάζουμε ανήκουν στην δεύτερη κατηγορία.




***Εικόνα 2.2:*** *Μια ασπρόμαυρη εικόνα όπως αυτή αναπαρίσταται σε έναν υπολογιστή. Η εικόνα έχει την μορφή ενός δισδιάστατου πίνακα ο οποίος περιέχει τις τιμές των εντάσεων των αντίστοιχων εικονοστοιχείων (pixels) [15].*




***Εικόνα 2.3:*** *Κάθε έγχρωμη RGB εικόνα έχει την μορφή τριών πινάκων. Υπάρχει ένας πίνακας για κάθε κανάλι χρώματος, δηλαδή ένας πίνακας για το κόκκινο (Red), ένας για το πράσινο (Green) κι ένας για το μπλε (Blue) [16].*




***Εικόνα 2.4:*** *Η τελική τιμή που έχει ένα εικονοστοιχείο μιας έγχρωμης εικόνας αποτελείται από τον συνδυασμό των αντίστοιχων τιμών που έχει το εικονοστοιχείο αυτό στους τρεις πίνακες [15].*

### 2.2.1 HOG Descriptor

Ο περιγραφέας HOG παρουσιάστηκε το 2005 από τους Dalal και Triggs [10] και ουσιαστικά αυτό που κάνει είναι να περιγράφει τα χαρακτηριστικά μιας εικόνας. Εφαρμόζεται στο στάδιο της εξαγωγής χαρακτηριστικών και χρησιμοποιείται αρκετά για

την ανίχνευση πεζών. Η βασική ιδέα πίσω από αυτόν είναι ότι η εμφάνιση και το σχήμα των αντικειμένων μπορούν να χαρακτηριστούν αρκετά ικανοποιητικά από τις κατευθύνσεις των ακμών τους. Ένα βασικό πλεονέκτημα του είναι ότι η αναπαράσταση που παρέχει είναι αρκετά ανθεκτική ως προς τις γεωμετρικές και φωτομετρικές αλλαγές [10]. Οι γεωμετρικές αλλαγές είναι οι αλλαγές που αφορούν τη θέση του αντικειμένου, όπως για παράδειγμα η περιστροφή του, ενώ οι φωτομετρικές αλλαγές έχουν να κάνουν με μεταβολές στη φωτεινότητα.

Σαν πρώτο βήμα, ο περιγραφέας δέχεται ως είσοδο μια εικόνα μεγέθους 128 x 64 pixels. Το μέγεθος της εικόνας εισόδου ποικίλλει ανάλογα με την εφαρμογή. Στην περίπτωση της αναγνώρισης πεζών συνιστάται η χρήση του παραπάνω μεγέθους [17]. Παραδείγματα τέτοιων εικόνων παρουσιάζονται στην Εικόνα 2.5.

***Εικόνα 2.5****. Παραδείγματα εικόνων που εμπεριέχονται στο INRIA dataset το οποίο χρησιμοποιήθηκε για την εκπαίδευση του αλγορίθμου [10].*

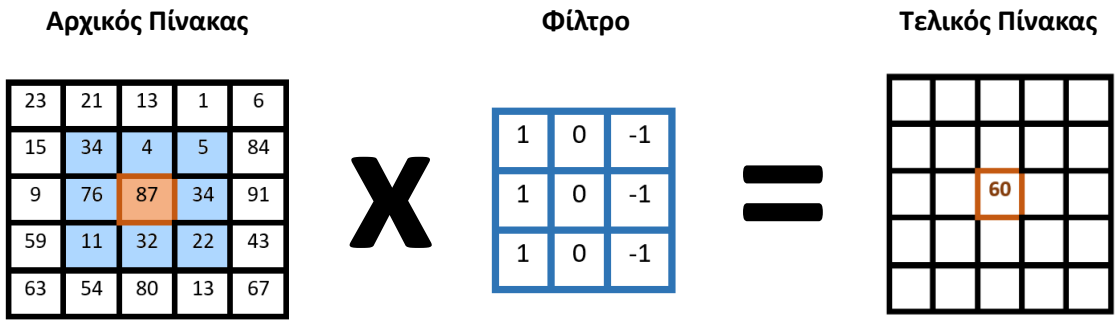
Έπειτα υπολογίζονται οι κάθετες και οι οριζόντιες κλίσεις της εικόνας εισόδου. Αυτό γίνεται χρησιμοποιώντας δύο φίλτρα, ένα για τις κάθετες και ένα για τις οριζόντιες κλίσεις, τα οποία έχουν την μορφή πινάκων όπως φαίνεται και στην Εικόνα 2.6.

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

***Εικόνα 2.6:*** *Τα φίλτρα τα οποία χρησιμοποιούνται για τον υπολογισμό των κλίσεων. Με το αριστερό φίλτρο υπολογίζονται οι οριζόντιες κλίσεις και με το δεξί οι κάθετες [18].*

Τα φίλτρα αυτά σαρώνουν την εικόνα και παράγονται δύο πίνακες, οι τιμές των οποίων υπολογίζονται με την διαδικασία της συνέλιξης. Στην Εικόνα 2.7 παρατίθεται ένα τέτοιο παράδειγμα, όπου φαίνεται πως προκύπτει μια νέα τιμή για ένα παλιό στοιχείο ενός πίνακα, αφού έχει εφαρμοστεί σε αυτόν ένα φίλτρο, με την διαδικασία της συνέλιξης.




**1x34 + 1x76 + 1x11 + 0x4 + 0x87 + 0x32 + (-1)x5 + (-1)x34 + (-1)x22 = 60**

***Εικόνα 2.7:*** *Οι τιμές που έχει ο αρχικός πίνακας αντιστοιχούν στις τιμές των εικονοστοιχείων (pixels) της εικόνας. Μία νέα τιμή προκύπτει από τον πολλαπλασιασμό κάθε παλιάς τιμής με την αντίστοιχη τιμή του φίλτρου. Αξίζει να σημειωθεί ότι η πράξη της συνέλιξης διαφέρει από αυτή του πολλαπλασιασμού πινάκων.*

Οι δύο νέοι αυτοί πίνακες αντιστοιχούν σε δύο νέες εικόνες, όπως φαίνεται στην Εικόνα 2.8, όπου η μία αναπαριστά τις κάθετες και η άλλη τις οριζόντιες κλίσεις. Μπορεί να παρατηρήσει κανείς ότι στην Εικόνα 2.8b, όπου παρουσιάζεται το αποτέλεσμα του υπολογισμού των οριζόντιων κλίσεων, οι οριζόντιες ακμές είναι πιο έντονες ενώ στην Εικόνα 2.8c, όπου φαίνεται το αποτέλεσμα του υπολογισμού των κάθετων κλίσεων, πιο έντονες είναι οι κάθετες ακμές.

Αφού έχουν υπολογιστεί οι κάθετες και οι οριζόντιες κλίσεις, υπολογίζονται τα διανύσματα κλίσεων. Για να γίνει αυτό, πρέπει να υπολογιστούν το μέτρο και η κατεύθυνσή τους και ο υπολογισμός, για κάθε pixel, γίνεται ως εξής:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = tan^{-1}\frac{g_y}{g_x}$$

Το $g_x$αποτελεί την οριζόντια συνιστώσα της κλίσης του αντίστοιχου pixel και την τιμή της την παίρνουμε από την εικόνα οριζόντιων κλίσεων, ενώ το $g_y$ αποτελεί την κάθετη συνιστώσα της κλίσης για το ίδιο pixel, την τιμή της οποίας παίρνουμε από την εικόνα κάθετων κλίσεων. Οι παραπάνω υπολογισμοί γίνονται για όλα τα pixel της εικόνας και το αποτέλεσμα παρουσιάζεται στην Εικόνα 2.8d.

(a) (b) (c) (d)

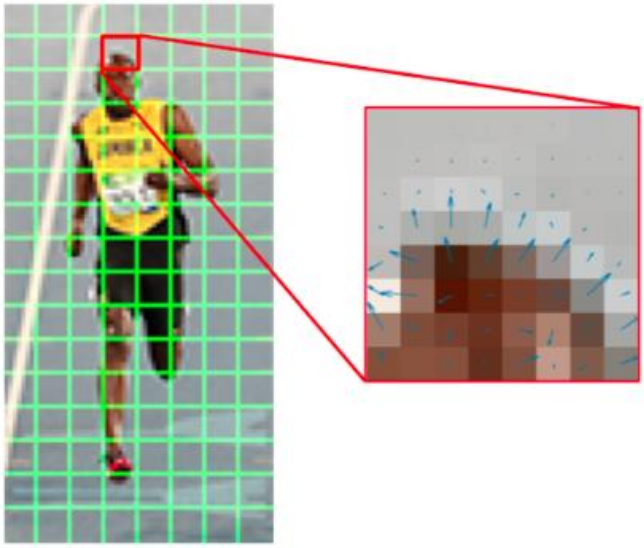
***Εικόνα 2.8:*** *(a) Αρχική εικόνα. (b) Εικόνα οριζόντιων κλίσεων. (c) Εικόνα κάθετων κλίσεων. (d) Τελικό αποτέλεσμα [19].*

Στην περίπτωση που έχουμε έγχρωμες εικόνες υπολογίζονται οι οριζόντιες και οι κάθετες κλίσεις ξεχωριστά για κάθε κανάλι χρώματος όπως επίσης και τα μέτρα και οι γωνίες των αντίστοιχων διανυσμάτων κλίσεων. Για κάθε εικονοστοιχείο, από τα τρία διανύσματα κλίσεων που προκύπτουν (ένα για κάθε κανάλι χρώματος) από την παραπάνω διαδικασία, κρατάμε αυτό το διάνυσμα που έχει την μεγαλύτερη τιμή ως προς το μέτρο σε σχέση με τα άλλα δύο.

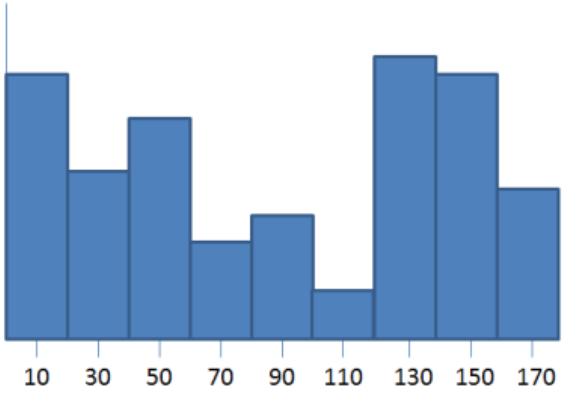
Το επόμενο βήμα είναι ο υπολογισμός του ιστογράμματος των κλίσεων. Για να γίνει αυτό, η εικόνα διαιρείται σε κελιά όπως φαίνεται στην Εικόνα 2.9. Το κάθε κελί έχει μέγεθος 8 x 8 pixels. Το μέγεθος αυτό όπως και το μέγεθος της εικόνας εισαγωγής εξαρτάται από το είδος της εφαρμογής. Τα κελιά μεγέθους 8 x 8 pixels είναι ιδανικά για την αναγνώριση ανθρώπων [17].

Μέσα σε κάθε κελί υπάρχουν 8 x 8 = 64 pixels και σε κάθε pixel αντιστοιχεί ένα διάνυσμα κλίσης. Στην Εικόνα 2.9 απεικονίζονται τα 64 αυτά διανύσματα τα οποία έπειτα κατανέμονται σε ένα ιστόγραμμα 9 θέσεων. Αυτές οι 9 θέσεις του ιστογράμματος αντιπροσωπεύουν 9 διαστήματα γωνιών εύρους $20^o$ το καθένα. Η κατανομή των διανυσμάτων σε αυτές τις 9 θέσεις του ιστογράμματος, γίνεται με βάση την γωνία κλίσης

του διανύσματος ενώ η συνεισφορά ή ψήφος του διανύσματος στην θέση στην οποία κατανέμεται εξαρτάται από το μέτρο του. Αξίζει να σημειωθεί ότι συνήθως οι γωνίες των κλίσεων των διανυσμάτων παίρνουν τιμές από $0^o$ έως $180^o$ κι όχι από $0^o$ έως $360^o$ [10]. Ένα παράδειγμα τέτοιου ιστογράμματος υπάρχει στην Εικόνα 2.10. Αυτή η διαδικασία γίνεται για όλα τα κελιά της εικόνας. Αν το μέγεθος της εικόνας είναι 128 x 64 pixels προκύπτουν 128 κελία δηλαδή 128 ιστογράμματα.



***Εικόνα 2.9:*** *Αριστερά απεικονίζεται η διαίρεση της εικόνας σε κελιά μεγέθους 8 x 8 pixels το καθένα. Δεξιά απεικονίζονται τα διανύσματα κλίσεων των pixels για ένα κελί της εικόνας [20].*




***Εικόνα 2.10:*** *Ιστόγραμμα κλίσεων 9 θέσεων [17].*

Μετά τον υπολογισμό του ιστογράμματος για κάθε κελί της εικόνας, ακολουθεί η κανονικοποίηση αυτών των ιστογραμμάτων που προέκυψαν. Ο λόγος που γίνεται αυτό

είναι γιατί η κανονικοποίηση παρέχει ανεξαρτησία ως προς τις αλλαγές στην φωτεινότητα, αλλαγές που μπορεί να επηρεάσουν τα μέτρα των διανυσμάτων [10].

Αρχικά τα κελιά ομαδοποιούνται σε blocks των τεσσάρων κελιών το καθένα. Κάθε ιστόγραμμα σε κάθε κελί αναπαρίσταται από ένα διάνυσμα 9 θέσεων, συνεπώς ένα block αναπαρίσταται από ένα διάνυσμα 36 θέσεων αφού σε κάθε block υπάρχουν 4 κελιά. Η κανονικοποίηση γίνεται σε αυτό το διάνυσμα 36 θέσεων διαιρώντας κάθε στοιχείο του με την ευκλείδεια νόρμα του διανύσματος. Η παραπάνω διαδικασία γίνεται σε όλη την εικόνα με επικάλυψη των blocks κατά 50% όπως φαίνεται και στην Εικόνα 2.11.


***Εικόνα 2.11:*** *Η επικάλυψη των blocks κατά της διαδικασία της κανονικοποίησης των ιστογραμμάτων τους [17].*

Τέλος, αυτή η διαδικασία έχει ως αποτέλεσμα την παραγωγή ενός διανύσματος χαρακτηριστικών που θα χρησιμοποιηθεί ως είσοδος στον ταξινομητή SVM του επόμενου σταδίου της αρχιτεκτονικής μας. Αυτό το τελικό διάνυσμα περιέχει όλα τα κανονικοποιημένα διανύσματα όλων των blocks. Για μία εικόνα 128 x 64 pixels, έχουμε 15 x 7 = 105 blocks σε καθένα από τα οποία αντιστοιχεί ένα διάνυσμα 36 θέσεων, συνεπώς το τελικό διάνυσμα χαρακτηριστικών έχει 105 x 36 = 3780 θέσεις.

### 2.2.2 Χαρακτηριστικά τύπου Haar

Τα χαρακτηριστικά τύπου Haar παρουσιάστηκαν από τους Viola και Jones το 2001 [11] και με βάση τις τιμές των χαρακτηριστικών αυτών γίνεται η διαδικασία της ταξινόμησης. Οι Viola και Jones εμπνεύστηκαν από την δουλεία των Papageorgiou κ.α. [21], οι οποίοι χρησιμοποίησαν κυματίδια τύπου Haar για να κωδικοποιήσουν οπτικά μοτίβα [14]. Γενικά η χρήση χαρακτηριστικών, για την αναπαράσταση του περιεχομένου μιας εικόνας, προτιμάται σε αντίθεση με τις απλές τιμές των εικονοστοιχείων και δύο βασικοί λόγοι είναι

η ακρίβεια στην ανίχνευση και η ταχύτητα που παρέχουν. Παρόλο που στην έρευνα τους εστίασαν κυρίως στο θέμα της ανίχνευσης προσώπων, υποστήριξαν ότι η μέθοδος τους μπορεί να εφαρμοστεί και να λειτουργήσει για την ανίχνευση διαφόρων άλλων αντικειμένων όπως είναι αμάξια, άνθρωποι κ.α. [22]. Τέλος, αξίζει να σημειωθεί ότι η μέθοδος αυτή εφαρμόζεται σε ασπρόμαυρες εικόνες.

Στην Εικόνα 2.12 παρουσιάζονται τρεις τύποι τέτοιων χαρακτηριστικών. Ο υπολογισμός των τιμών αυτών των χαρακτηριστικών γίνεται αφαιρώντας το άθροισμα των τιμών των εικονοστοιχείων που βρίσκονται στις μαύρες περιοχές, από το άθροισμα των τιμών των εικονοστοιχείων που βρίσκονται στις λευκές περιοχές. Το ελάχιστο μέγεθος που μπορεί να έχει αυτή η περιοχή της εικόνας, για την οποία γίνονται οι παραπάνω υπολογισμοί, είναι ένα παράθυρο μεγέθους 24 x 24 pixels το οποίο συχνά λέγεται και παράθυρο εντοπισμού (detection window). Μόνο από μια τόσο μικρή περιοχή της εικόνας, μπορεί να προκύψουν πάνω από 180.000 τιμές χαρακτηριστικών. Αν σκεφτεί κανείς ότι αυτοί οι υπολογισμοί γίνονται για κάθε τέτοιο παράθυρο της εικόνας, η διαδικασία έχει ως αποτέλεσμα την δημιουργία ενός αρκετά μεγάλου συνόλου χαρακτηριστικών.


***Εικόνα 2.12:*** *Παραδείγματα τετράγωνων χαρακτηριστικών σε σχέση με το παράθυρο εντοπισμού. Τα σχήματα A και B απεικονίζουν χαρακτηριστικά δύο τετραγώνων, το σχήμα C χαρακτηριστικά τριών τετραγώνων και το σχήμα D χαρακτηριστικά τεσσάρων τετραγώνων [22].*

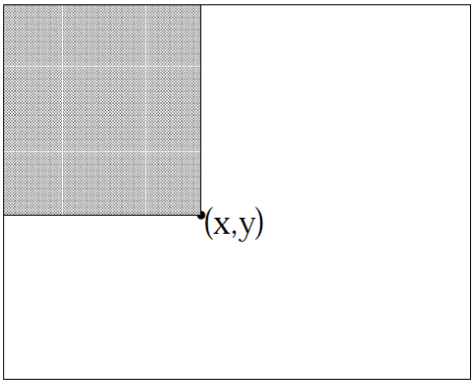
Οι Viola και Jones εισήγαγαν έναν πρωτοποριακό τρόπο για τον γρήγορο υπολογισμό των τετράγωνων χαρακτηριστικών που αναφέρθηκαν παραπάνω και αυτός ήταν η εικόνα ολοκλήρωσης. Η θέση κάθε εικονοστοιχείου μπορεί να εκφραστεί σαν ένα ζεύγος τιμών $(x, y)$, όπου το $x$ είναι η οριζόντια και το $y$ η κάθετη απόσταση αυτού του pixel από την αρχή της εικόνας. Ο υπολογισμός της εικόνας ολοκλήρωσης σε μία θέση $(x, y)$ της αρχικής εικόνας, γίνεται αθροίζοντας τις τιμές των εικονοστοιχείων της περιοχής που βρίσκεται

πάνω και αριστερά από τη θέση αυτή, όπως φαίνεται και στην Εικόνα 2.13. Ο μαθηματικός τύπος είναι ο:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y')$$

όπου $ii(x,y)$ είναι η εικόνα ολοκλήρωσης και $i(x,y)$ η αρχική εικόνα.

Η εικόνα ολοκλήρωσης ονομάζεται έτσι, γιατί είναι ουσιαστικά το διπλό ολοκλήρωμα της εικόνας, πρώτα ως προς τις γραμμές κι έπειτα ως προς τις στήλες [11].




***Εικόνα 2.13:*** *Παράδειγμα υπολογισμού εικόνας ολοκλήρωσης σαν άθροισμα των στοιχείων που βρίσκονται πάνω και αριστερά από το* $(x,y)$ *[11].*

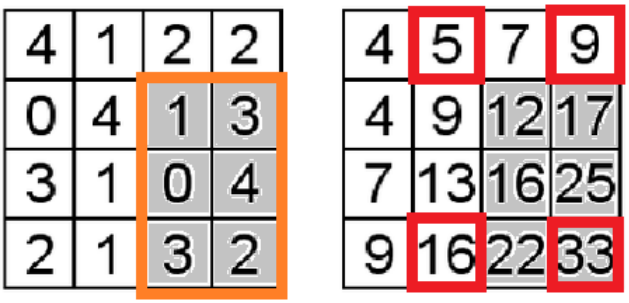
Χρησιμοποιώντας τις δύο παρακάτω σχέσεις η εικόνα ολοκλήρωσης μπορεί να υπολογιστεί με ένα πέρασμα πάνω από την αρχική εικόνα [22].

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

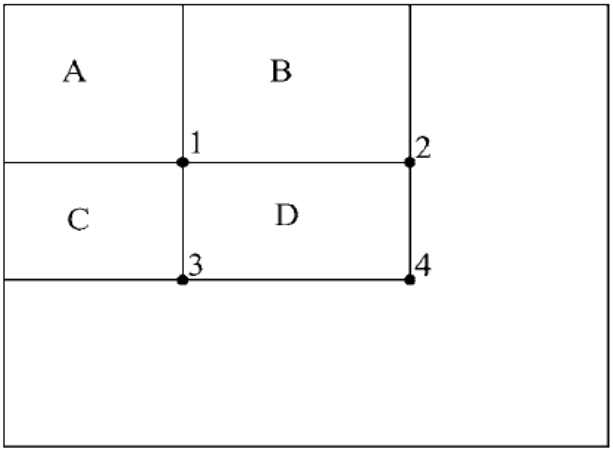
Με $s(x,y)$ συμβολίζεται το σωρευτικό άθροισμα μέχρι το εικονοστοιχείο $(x,y)$ και ισχύει ότι $s(x,-1) = 0$ και $s(-1,y) = 0$.

Το πλεονέκτημα αυτής της μεθόδου είναι ότι το άθροισμα μιας ορθογώνιας περιοχής μπορεί να υπολογιστεί χρησιμοποιώντας μόνο τέσσερις τιμές. Στο παράδειγμα της Εικόνας 2.14 παρουσιάζονται αναλυτικά οι υπολογισμοί για μία τέτοια ορθογώνια περιοχή. Παρατηρούμε, ότι κάνοντας χρήση της εικόνας ολοκλήρωσης, οδηγούμαστε σε λιγότερες

αριθμητικές πράξεις. Γενικά, το άθροισμα των τιμών των εικονοστοιχείων μίας περιοχής D, όπως δείχνει η Εικόνα 2.15, μπορεί να υπολογιστεί ως D = 4 + 1 – (2 + 3), όπου το 1 αντιστοιχεί στο άθροισμα των τιμών των εικονοστοιχείων της περιοχής A, το 2 στο άθροισμα των περιοχών A και B, το 3 στο άθροισμα των περιοχών A και C και το 4 στο άθροισμα όλων των περιοχών A, B, C και D.




***Εικόνα 2.14:*** *Αριστερά βρίσκεται η αρχική εικόνα και δεξιά η εικόνα ολοκλήρωσης. Το άθροισμα των τιμών στην γκρίζα περιοχή αν χρησιμοποιηθεί η αριστερή εικόνα είναι 1+3+0+4+3+2=13, ενώ αν χρησιμοποιηθεί η δεξιά εικόνα είναι 33+5-16-9=13. Παράγεται συνεπώς το ίδιο αποτέλεσμα αλλά με λιγότερους υπολογισμούς [23].*




***Εικόνα 2.15:*** *Παράδειγμα χρήσης των περιοχών A, B, C, και D της εικόνας ολοκλήρωσης για τον υπολογισμό του αθροίσματος των εικονοστοιχείων στα σημεία 1, 2, 3 και 4 [22].*

## 2.3 Ταξινόμηση

Στο δεύτερο στάδιο ενός συστήματος αναγνώρισης εικόνων, υλοποιείται ο αλγόριθμος της ταξινόμησης. Στην μηχανική μάθηση και στην στατιστική η ταξινόμηση είναι η διαδικασία στην οποία ένα νέο δείγμα τοποθετείται σε μία κατηγορία (κλάση), με βάση ένα σύνολο δειγμάτων, για τα οποία έχει ήδη καθοριστεί η κλάση στην οποία ανήκουν [24].

Ο αλγόριθμος λοιπόν ταξινόμησης δέχεται ως είσοδο τα χαρακτηριστικά του προηγούμενου σταδίου και τα ταξινομεί σε μια θετική κλάση αν σχετίζονται με το αντικείμενο ενδιαφέροντος ή σε μια αρνητική κλάση στην αντίθετη περίπτωση. Οι ταξινομητές που χρησιμοποιούνται για την αναγνώριση ανθρώπων είναι ταξινομητές δύο κλάσεων (binary classifiers).

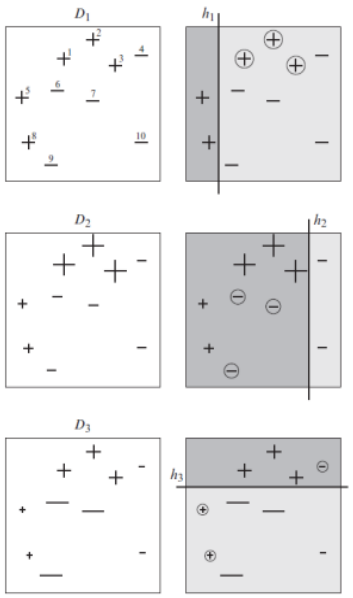
Τα Support Vector Machines (SVM) και ο AdaBoost σε συνδυασμό με ένα καταρράχτη ταξινομητών (cascade classifier) είναι δύο από τους πιο γνωστούς αλγόριθμους ταξινόμησης και είναι αυτοί που αναλύονται παρακάτω.

### 2.3.1 AdaBoost

Από την διαδικασία εξαγωγής χαρακτηριστικών, για κάθε ένα παράθυρο εντοπισμού μπορεί να προκύψει ένας πολύ μεγάλος αριθμός χαρακτηριστικών. Κάθε ένα από αυτά τα χαρακτηριστικά μπορεί να υπολογιστεί αρκετά γρήγορα κάνοντας χρήση της εικόνας ολοκλήρωσης, αλλά ο υπολογισμός όλου του συνόλου των χαρακτηριστικών μιας εικόνας είναι μια διαδικασία με υψηλό υπολογιστικό κόστος. Στην προκειμένη περίπτωση, οι Viola και Jones [11] υποστηρίζουν ότι ο συνδυασμός ενός μικρού αριθμού χαρακτηριστικών αρκεί για να δημιουργηθεί ένας αποτελεσματικός ταξινομητής. Για να το πετύχουν αυτό, χρησιμοποιούν μια παραλλαγή του αλγορίθμου AdaBoost για την επιλογή των πιο σημαντικών χαρακτηριστικών και την εκπαίδευση του ταξινομητή.

Ο AdaBoost είναι ένας αλγόριθμος ενίσχυσης (boosting), συγκεκριμένα προσαρμοστικής ενίσχυσης (adaptive boosting), που παρουσιάστηκε από τους Freund και Schapire [25] το 2003 και συνδυάζει πολλούς αδύναμους ταξινομητές για να σχηματίσει έναν ισχυρό ταξινομητή.

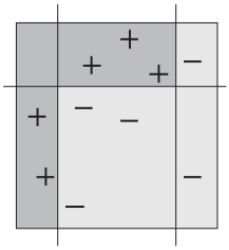
Ακολουθεί ένα παράδειγμα για το πως λειτουργεί ένας τέτοιος αλγόριθμός ενίσχυσης και ποια είναι η λογική του. Αρχικά, όπως φαίνεται και στην Εικόνα 2.16, υπάρχουν 10 δείγματα που ανήκουν σε δύο κατηγορίες (κλάσεις) [26]. Με το σύμβολο συν (+) συμβολίζουμε τα πέντε δείγματα που ανήκουν στην μια κλάση και με το σύμβολο πλην (-) τα πέντε δείγματα που ανήκουν στην άλλη κλάση. Σκοπός του ταξινομητή είναι να βρει μια συνάρτηση η οποία να διαχωρίζει τα δείγματα όσο το δυνατόν καλύτερα. Όπως αναφέρθηκε, ο αλγόριθμος AdaBoost χρησιμοποιεί έναν συνδυασμό από αδύναμους ταξινομητές (weak classifiers). Οι ταξινομητές αυτοί λέγονται αδύναμοι διότι αδυνατούν να ταξινομήσουν σωστά όλα τα δεδομένα [11].




***Εικόνα 2.16:*** *Παράδειγμα ενός αλγορίθμου ενίσχυσης (boosting). Στην αριστερή στήλη παρουσιάζεται η κατανομή των δειγμάτων και στην δεξιά στήλη η ταξινόμηση των δειγμάτων αυτών από τον εκάστοτε αδύναμο ταξινομητή $h_t$. Η αυξομείωση του μεγέθους των συμβόλων αναλογεί στην αυξομείωση των βαρών των αντίστοιχων δειγμάτων [26].*

Όπως φαίνεται και στο πρώτο γράφημα της δεύτερης στήλης στην Εικόνα 2.16, ο πρώτος ταξινομητής ταξινομεί λάθος τρία από τα δέκα δείγματα, δηλαδή το ποσοστό λάθους ταξινόμησης είναι 30%. Στην επόμενη επανάληψη, ο αλγόριθμος ενισχύει τα βάρη των δειγμάτων που δεν ταξινομήθηκαν σωστά, μειώνει τα βάρη αυτών που ταξινομήθηκαν και προσπαθεί ξανά να διαχωρίσει τα δείγματα. Αυτή η επαναληπτική διαδικασία έχει ως αποτέλεσμα την δημιουργία ενός τελικού ισχυρού ταξινομητή ο οποίος παρουσιάζεται στην Εικόνα 2.17. Παρατηρούμε ότι παρόλο που κανένας αδύναμος ταξινομητής δεν

μπόρεσε να ταξινομήσει σωστά όλα τα δείγματα, ο τελικός ισχυρός ταξινομητής κατάφερε να το κάνει.



***Εικόνα 2.17:*** *Ο τελικός ισχυρός ταξινομητής ο οποίος κατάφερε να διαχωρίσει σωστά όλα τα δείγματα [26].*

Στον αλγόριθμο AdaBoost που χρησιμοποιείται από τους Viola και Jones με σκοπό να βρεθεί ένα μικρό σύνολο καλών χαρακτηριστικών, ο υπολογισμός της συνάρτησης ταξινόμησης βασίζεται σε ένα μόνο χαρακτηριστικό, για αυτό και κάθε χαρακτηριστικό μπορεί να θεωρηθεί σαν ένας αδύναμος ταξινομητής. Ο αλγόριθμος προσπαθεί να βρει αυτό το χαρακτηριστικό με βάση το οποίο διαχωρίζονται καλύτερα τα θετικά από τα αρνητικά δείγματα, όπου θετικά δείγματα είναι οι περιοχές της εικόνας που περιέχουν το αντικείμενο προς αναγνώριση ενώ αρνητικά δείγματα οι περιοχές της εικόνας που δεν το περιέχουν. Παρακάτω δίνεται η σχέση που περιγράφει έναν τέτοιο ταξινομητή:

$$h_j(x) = \begin{cases} 1, & p_j f_j(x) < p_j \theta_j \\ 0, & \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Το $h_j$ συμβολίζει την συνάρτηση ταξινόμησης που αντιστοιχεί στο χαρακτηριστικό $f_j$, το $\theta_j$ είναι μία τιμή κατωφλίου η οποία ορίζεται ώστε ο αριθμός των δειγμάτων που θα ταξινομηθούν λάθος να είναι ο ελάχιστος, το $p_j$ ορίζει την φορά της ανισότητας και το $x$ είναι ένα παράθυρο 24x24 pixel.

Παρακάτω παρουσιάζεται η υλοποίηση του αλγορίθμου AdaBoost.

- Έστω n παραδείγματα εικόνων $(x_1, y_1), \dots, (x_n, y_n)$όπου $y_i = 0$ για αρνητικά δείγματα και $y_i = 1$ για θετικά δείγματα.
- Γίνεται αρχικοποίηση στα βάρη των δειγμάτων $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ για $y_i = 0,1$ για τα αρνητικά και τα θετικά δείγματα αντίστοιχα.
- Για t = 1, …, T:

1. Γίνεται κανονικοποίηση στα βάρη έτσι ώστε το $w_t$ να αποτελεί μια κατανομή πιθανότητας:

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

2. Για κάθε χαρακτηριστικό j γίνεται εκπαίδευση ενός ταξινομητή $h_j$ με βάση μόνο αυτό το χαρακτηριστικό j και υπολογίζεται το σφάλμα ταξινόμησης για όλα τα δείγματα εικόνων ως εξής:

$$e_j = \sum_i w_i * |h_j * (x_i - y_i)|$$

3. Επιλέγεται ο ταξινομητής με το μικρότερο σφάλμα. Από την στιγμή που οι ταξινομητές εκπαιδεύτηκαν χρησιμοποιώντας μόνο ένα χαρακτηριστικό ο καθένας, ο ταξινομητής με το μικρότερο σφάλμα αντιστοιχεί στο χαρακτηριστικό με το μικρότερο σφάλμα.

4. Τα βάρη ενημερώνονται παίρνοντας τις νέες τους τιμές ως εξής:

$$w_{t+1,i} = w_{t,i} * \beta_t^{1-e_i}$$

όπου $e_i = 0$ αν η εικόνα $x_i$ ταξινομήθηκε σωστά, $e_i = 1$ αν η εικόνα $x_i$ ταξινομήθηκε λάθος και $\beta_t = \frac{e_t}{1-e_t}$

- Ο τελικός ταξινομητής προκύπτει ως εξής:

$$h(x) = \begin{cases} 1, \sum_{t=1}^{T} a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

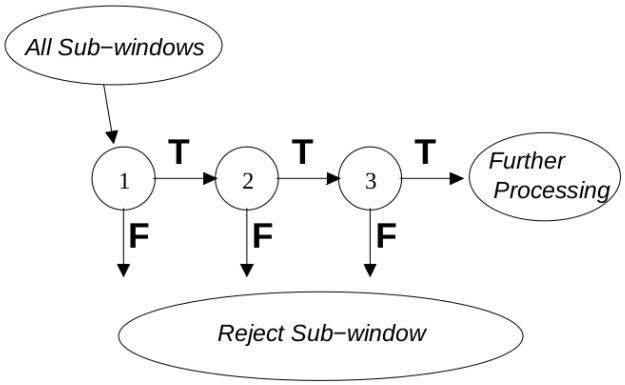
όπου $\alpha_t = log \frac{1}{\beta_t}$

### 2.3.2 Καταρράχτης Ταξινομητών

Προκειμένου να γίνει η αναγνώριση κάποιου αντικειμένου, κάθε εικόνα σαρώνεται από ένα παράθυρο εντοπισμού. Συνήθως το αντικείμενο ενδιαφέροντος καταλαμβάνει ένα μικρό μέρος της εικόνας, με αποτέλεσμα τα περισσότερα παράθυρα εντοπισμού να ταξινομούνται ως αρνητικά.

Οι Viola και Jones [11] προτείνουν ένα σχήμα ταξινόμησης, με σκοπό να αυξήσουν την απόδοση των ανιχνεύσεων και να μειώσουν το υπολογιστικό κόστος. Η βασική ιδέα είναι να χρησιμοποιηθούν αρχικά απλοί ταξινομητές, για να απορριφθεί η πλειοψηφία των παραθύρων που δεν περιέχουν το αντικείμενο ενδιαφέροντος, κι έπειτα, για την διαδικασία της ανίχνευσης, να εφαρμοστούν πιο σύνθετοι ταξινομητές στα παράθυρα που πιθανόν το περιέχουν.

Η ταξινόμηση γίνεται από έναν καταρράχτη ταξινομητών (cascade classifier) που έχει την μορφή ενός δέντρου απόφασης (decision tree) όπως φαίνεται στην Εικόνα 2.18. Ο ταξινομητής αυτός αποτελείται από στάδια, σε καθένα από τα οποία υπάρχει ένας ισχυρός ταξινομητής, η εκπαίδευση του οποίου γίνεται με τον αλγόριθμο AdaBoost. Κάθε υποπαράθυρο εισέρχεται στο πρώτο στάδιο και προχωράει στο επόμενο μόνο αν δεν απορριφθεί από το τρέχων στάδιο. Αν κάποιο παράθυρο απορριφθεί, τότε δεν εξετάζεται ξανά.




***Εικόνα 2.18:*** *Καταρράχτης ταξινομητών σε μορφή δέντρου απόφασης [11].*

Οι ταξινομητές που βρίσκονται στα πρώτα στάδια ενός τέτοιου καταρράχτη είναι απλοί, δηλαδή κατασκευάζονται χρησιμοποιώντας έναν μικρό αριθμό χαρακτηριστικών. Αυτό έχει ως συνέπεια έναν μεγάλο αριθμό ανιχνεύσεων οι οποίες όμως στην πλειοψηφία τους είναι λανθασμένες. Αυτό δεν αποτελεί πρόβλημα καθώς τα παράθυρα που σχετίζονται με

λάθος ανιχνεύσεις, αναμένεται να απορριφθούν σε κάποιο επόμενο στάδιο του ταξινομητή. Για την εκπαίδευση των ταξινομητών των επόμενων σταδίων χρησιμοποιούνται περισσότερα χαρακτηριστικά και έτσι οι ταξινομητές που κατασκευάζονται είναι πιο σύνθετοι.

Για να επιτευχθεί μεγάλος αριθμός ανιχνεύσεων στα πρώτα στάδια, μειώνεται η τιμή κατωφλίου $\frac{1}{2}\sum_{t=1}^{T} a_t$ του αλγορίθμου AdaBoost. Η τιμή αυτή είναι ένας αριθμός που συγκρίνεται με το ζυγισμένο άθροισμα των τιμών των χαρακτηριστικών κάθε σταδίου και καθορίζει αν ένα δείγμα ταξινομηθεί ως θετικό ή αρνητικό.

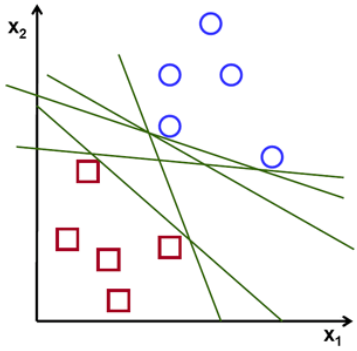
Τέλος, υπάρχει ένας επιθυμητός αριθμός ανιχνεύσεων τόσο για κάθε στάδιο όσο και για τον συνολικό ταξινομητή. Σε κάθε στάδιο η επίτευξη των επιθυμητών ανιχνεύσεων γίνεται με την πρόσθεση χαρακτηριστικών κατά την εκπαίδευση του εκάστοτε ταξινομητή, ενώ για τον συνολικό ταξινομητή η επίτευξη του στόχου αυτού γίνεται με την πρόσθεση σταδίων.

### 2.3.3 Support Vector Machines

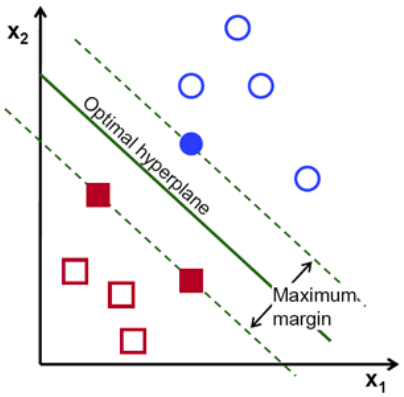
Οι Μηχανές Διανυσμάτων Υποστήριξης ή Support Vector Machines (SVM) είναι ένας αλγόριθμος επιτηρούμενης μάθησης (supervised learning) που χρησιμοποιείται για την εκπαίδευση ενός ταξινομητή. Η ιδέα των SVM ξεκίνησε το 1963 αλλά η πιο διαδεδομένη υλοποίηση είναι αυτή των Cortes και Vapnik [27]. Μια τέτοια μέθοδο ταξινόμησης επέλεξαν ο Dalal και ο Triggs [11] στο σύστημα ανίχνευσης ανθρώπων που υλοποίησαν. Συγκεκριμένα χρησιμοποίησαν ένα Linear SVM το οποίο ενδείκνυται για προβλήματα ταξινόμησης δύο κλάσεων. Ο αλγόριθμος αναπαριστά κάθε n-διάστατο διάνυσμα χαρακτηριστικών ως σημεία σε έναν n-διάστατο χώρο κι έπειτα προσπαθεί να βρει ένα υπερεπίπεδο έτσι ώστε να διαχωρίσει όσο το δυνατόν καλύτερα τα σημεία αυτά [28]. Κάθε νέο σημείο ταξινομείται με βάση την θέση του ως προς το υπερεπίπεδο στον χώρο αυτό. Η συνάρτηση απόφασης ορίζεται από ένα σύνολο δειγμάτων εκπαίδευσης που αποκαλούνται διανύσματα υποστήριξης.

Παρατίθεται ένα παράδειγμα [29] για να γίνει πιο κατανοητό το πως λειτουργεί ο αλγόριθμος. Για λόγους απλότητας, στο παράδειγμα χρησιμοποιούνται σημεία του δισδιάστατου χώρου που είναι γραμμικώς διαχωρίσιμα. Στον δισδιάστατο χώρο ο διαχωρισμός των σημείων γίνεται με ένα ευθύγραμμο τμήμα ενώ σε χώρους μεγαλύτερων

διαστάσεων χρησιμοποιούνται υπερεπίπεδα. Έστω λοιπόν ένα σύνολο δεδομένων $(x_1, y_1), \dots, (x_n, y_n)$ τα οποία ανήκουν σε δύο κλάσεις και είναι γραμμικώς διαχωρίσιμα, με $y_i = -1,1$ για τα αρνητικά και τα θετικά δείγματα αντίστοιχα, όπως φαίνεται στην Εικόνα 2.19. Παρατηρούμε ότι υπάρχουν πολλά ευθύγραμμα τμήματα που μπορούν να διαχωρίσουν τα δείγματα αυτά. Σκοπός του αλγορίθμου είναι να βρει εκείνο το υπερεπίπεδο (ευθύγραμμο τμήμα στην περίπτωση μας) το οποίο περνάει όσο πιο μακριά γίνεται από τα σημεία αυτά αφήνοντας το μεγαλύτερο δυνατό περιθώριο (margin). Ένα τέτοιο υπερεπίπεδο απεικονίζεται στην Εικόνα 2.20.




***Εικόνα 2.19:*** *Δείγματα γραμμικώς διαχωρίσιμα στο καρτεσιανό επίπεδο. Φαίνεται πως ο διαχωρισμός των δειγμάτων αυτών μπορεί να γίνει με πολλά ευθύγραμμα τμήματα [29].*




***Εικόνα 2.20:*** *Το ιδανικό ευθύγραμμο τμήμα που διαχωρίζει τα σημεία αφήνοντας την μέγιστη απόσταση μεταξύ της ευθείας και των σημείων [29].*

Δεδομένου ότι τα σημεία είναι γραμμικώς διαχωρίσιμα ισχύουν οι δύο παρακάτω σχέσεις:

$$x_i * \vec{w} + b \geq +1 \, για \, y_i = 1$$
$$x_i * \vec{w} + b \leq -1 \, για \, y_i = -1$$

όπου $\vec{w}$ είναι ένα κάθετο στο υπερεπίπεδο διάνυσμα που αποτελείται από βάρη, και $b$ μια τιμή πόλωσης. Οι παραπάνω σχέσεις συνδυάζονται σε μία:

$$y_i * (x_i * \vec{w} + b) - 1 \geq 0, \forall i$$

Αυτές οι δύο εξισώσεις ορίζουν δύο υπερεπίπεδα H1 και H2 όπως φαίνεται και στην Εικόνα 2.21 [30]. Το βέλτιστο υπερεπίπεδο H που διαχωρίζει τα δεδομένα αφήνοντας το μέγιστο περιθώριο μεταξύ τους δίνεται από την σχέση:

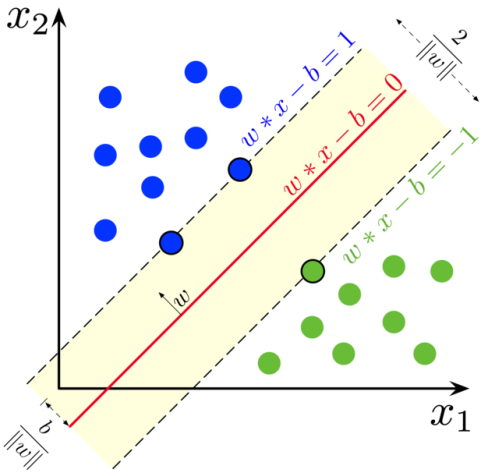
$$x_i * \vec{w} + b = 0$$

Η απόσταση των δύο υπερεπιπέδων H1 και H2 μεταξύ τους ορίζεται ως:

$$απόσταση = \frac{2}{\|\vec{w}\|}$$

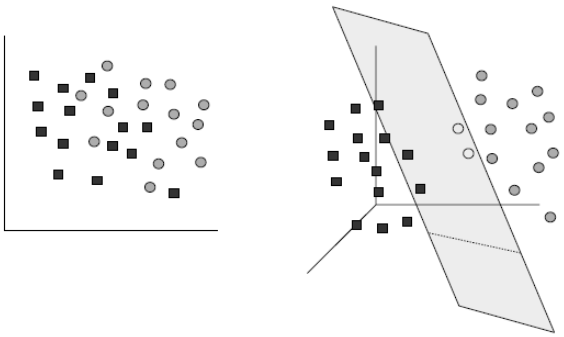
όπου $\|\vec{w}\|$ η Ευκλείδεια νόρμα του διανύσματος $\vec{w}$ [31]. Από την παραπάνω σχέση παρατηρούμε ότι για να μεγιστοποιήσουμε την περιοχή του περιθωρίου αρκεί να ελαχιστοποιήσουμε το $\|\vec{w}\|$. Για να το κάνουμε αυτό θα χρησιμοποιούμε πολλαπλασιαστές Lagrance. Πιο συγκεκριμένα θα ελαχιστοποιήσουμε την παρακάτω εξίσωση:

$$L_p = \frac{\|\vec{w}\|^2}{2} - \sum_{i=1}^{l} a_i * y_i * (x_i * \vec{w} + b) + \sum_{i=1}^{l} a_i$$




***Εικόνα 2.21:*** *Απεικονίζονται τα τρία υπερεπίπεδα H1, H2, H που ορίζονται από τις συναρτήσεις* $\vec{w} * x - b = 1, \vec{w} * x - b = -1$ *και* $\vec{w} * x - b = 0$ *αντίστοιχά καθώς και οι αποστάσεις μεταξύ τους [30].*

Στην περίπτωση που τα σημεία δεν είναι γραμμικώς διαχωρίσιμα, τα προβάλουμε σε ένα χώρο μεγαλύτερης διάστασης και ο αλγόριθμος προσπαθεί να βρει στον νέο αυτό χώρο ένα υπερεπίπεδο που να τα διαχωρίζει. Μία τέτοια περίπτωση παρουσιάζεται στην Εικόνα 2.22 [32].



(a) original data set (b) mapped feature space

***Εικόνα 2.22:*** *Αριστερά υπάρχει ένα σύνολο δειγμάτων στο καρτεσιανό επίπεδο που δεν είναι γραμμικώς διαχωρίσιμα. Δεξιά, κάθε σημείο προβάλλεται στον τρισδιάστατο χώρο όπου και γίνεται ο διαχωρισμός τους [32].*

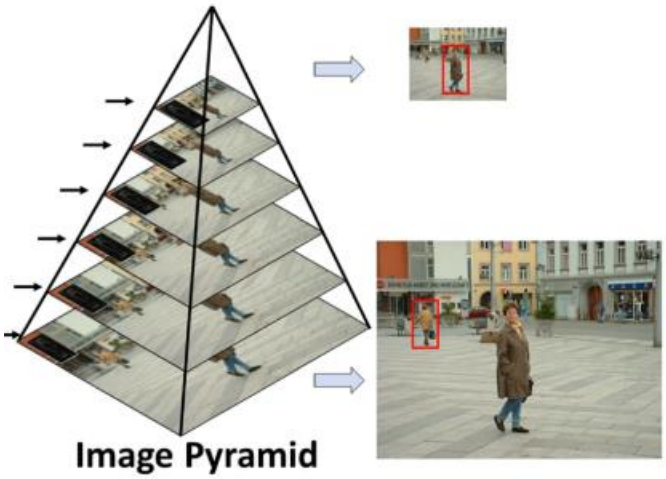
# ΚΕΦΑΛΑΙΟ 3 :Υλοποίηση αλγορίθμων

Η υλοποίηση των αλγορίθμων ανίχνευσης έγινε με τη χρήση της βιβλιοθήκης OpenCV σε περιβάλλον προγραμματισμού Python. Τα δύο συστήματα ανίχνευσης που εξετάστηκαν, λειτουργούν με παρόμοιο τρόπο καθώς και τα δύο βασίζονται στην μέθοδο των ολισθαινόντων παραθύρων (sliding windows). Στις μεθόδους αυτές, κάθε εικόνα από ένα σύνολο εικόνων σαρώνεται από ένα παράθυρο ανίχνευσης το οποίο ολισθαίνει κατά μήκος και κατά πλάτος της εικόνας όπως φαίνεται στην Εικόνα 3.1, με σκοπό να ανιχνεύσει το ζητούμενο αντικείμενο.

***Εικόνα 3.1:*** *Ο τρόπος με τον οποίο το παράθυρο ανίχνευσης ολισθαίνει, «σκανάροντας» την εικόνα, στις μεθόδους που χρησιμοποιούν ολισθαίνοντα παράθυρα (sliding windows).*

Ένα ακόμη κοινό στοιχείο των δύο αλγορίθμων, είναι ότι το μέγεθος της εικόνας εισόδου μεταβάλλεται έτσι ώστε η ανίχνευση να γίνει για πολλαπλές κλίμακες της εικόνας. Όπως φαίνεται στην Εικόνα 3.2, οι άνθρωποι σε μια εικόνα μπορεί να εμφανίζονται σε διάφορα μεγέθη και ανάλογα με το πόσο μακριά ή κοντά βρίσκονται σε σχέση με τον φακό, καταλαμβάνουν και τον αντίστοιχο αριθμό από pixels. Για να επιτευχθεί η ανίχνευση των ανθρώπων σε κάθε κλίμακα, το μέγεθος της εικόνας εισόδου αλλάζει και η διαδικασία του «σκαναρίσματος» γίνεται εκ νέου. Δημιουργείται δηλαδή μια πυραμίδα από εικόνες όπως φαίνεται και στην Εικόνα 3.3.

***Εικόνα 3.2:*** *Στην εικόνα φαίνονται δύο άνθρωποι σε διαφορετική κλίμακα. Οι άνθρωποι που βρίσκονται πιο κοντά στον φακό καταλαμβάνουν μεγαλύτερο μέρος στην εικόνα [33].*




***Εικόνα 3.3:*** *Το μέγεθος του παραθύρου ανίχνευσης παραμένει σταθερό ενώ το μέγεθος της εικόνας μεταβάλλεται δημιουργώντας έτσι μια πυραμίδα από εικόνες. Σκοπός αυτής της διαδικασίας είναι να μπορούν να ανιχνευτούν άνθρωποι κάθε κλίμακας [34].*

## 3.1 OpenCV

Η OpenCV (Open Source Computer Vision Library) είναι μια βιβλιοθήκη ανοιχτού λογισμικού που χρησιμοποιείται για εφαρμογές υπολογιστικής όρασης και μηχανικής μάθησης [35]. Η βιβλιοθήκη διαθέτει πάνω από 2500 αλγορίθμους και αναπτύχθηκε από την Intel το 1999 [36]. Κάποιες από τις χρήσεις των αλγορίθμων αυτών είναι η ανίχνευση και η αναγνώριση προσώπων ή αντικειμένων, η κατηγοριοποίηση ανθρώπινων κινήσεων σε βίντεο, η εξαγωγή τρισδιάστατων μοντέλων από αντικείμενα κ.α. Γενικότερα, η OpenCV είναι πολύ χρήσιμη σε εφαρμογές που σχετίζονται με την επεξεργασία και την ανάλυση εικόνων ή βίντεο και χρησιμοποιείται τόσο για εμπορικούς όσο και για ερευνητικούς σκοπούς [37].

Παρόλο που η βιβλιοθήκη αναπτύχθηκε αρχικά σε C++, διαθέτει interfaces και για άλλες γλώσσες προγραμματισμού όπως είναι η Java και η Python. Τέλος, υποστηρίζεται από τα πιο γνωστά λειτουργικά συστήματα όπως Windows, Linux, Mac OS, Android και διατίθεται δωρεάν.

Η έκδοση 3.2 της OpenCV είναι αυτή που χρησιμοποιήθηκε στην παρούσα εργασία.

## 3.2 Python

Η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου γενικού σκοπού. Δημιουργός της είναι ο Guido Van Rossum και κυκλοφόρησε για πρώτη φορά το 1991. Κύριος στόχος του ήταν η απλότητα τόσο στην σύνταξη όσο και στην ανάγνωση του κώδικα [38].

Η Python είναι μια αρκετά δημοφιλής γλώσσα. Σύμφωνα με το PYPL (PopularitY of Programming Language), που είναι ένας δείκτης ο οποίος αναπτύχθηκε από το GitHub και μετράει την δημοφιλία μιας γλώσσας με βάση τις αναζητήσεις μαθημάτων για αυτή τη γλώσσα στο Google, η Python βρίσκεται πρώτη σε αναζητήσεις με ποσοστό 31.6 % ανάμεσα σε 28 γλώσσες προγραμματισμού [39]. Η έκδοση της Python που χρησιμοποιήθηκε στην παρούσα εργασία είναι η 2.7.

Παρακάτω παρουσιάζονται αναλυτικά οι συναρτήσεις που χρησιμοποιήθηκαν.

## 3.3 Η Μέθοδος των Viola & Jones

Η υλοποίηση της μεθόδου που προτάθηκε από τους Viola και Jones [11] έγινε με συναρτήσεις της OpenCV. Οι δύο βασικές συναρτήσεις που χρησιμοποιήθηκαν ήταν η *CascadeClassifier* και η *detectMultiScale*. Ο κώδικας υπάρχει στο Παράρτημα Α.

### 3.3.1 Η συνάρτηση CascadeClassifier

Η OpenCV διαθέτει ένα σύνολο ήδη εκπαιδευμένων ταξινομητών για την ανίχνευση προσώπων ή σημείων σε αυτά όπως είναι τα μάτια, η μύτη, το στόμα καθώς επίσης και ένα σύνολο ταξινομητών που αφορούν την ανίχνευση του ανθρώπινου σώματος. Αυτοί οι ταξινομητές έχουν την μορφή ενός xml αρχείου. Ένα xml αρχείο έχει παρόμοια μορφή με ένα HTML κείμενο ωστόσο χρησιμοποιεί εξατομικευμένες ετικέτες για τον ορισμό αντικειμένων και δεδομένων εντός των αντικειμένων αυτών [40].

Η συνάρτηση *CascadeClassifier* χρησιμοποιείται για την φόρτωση ενός τέτοιου ταξινομητή [41] και η σύνταξη της είναι η εξής:

***<CascadeClassifier object> = cv2.CascadeClassifier(filename)***

Ως όρισμα η συνάρτηση δέχεται το αρχείο xml που περιέχει τον εκάστοτε ταξινομητή. Στην συγκεκριμένη περίπτωση χρησιμοποιήθηκε ο ταξινομητής που βρίσκεται στο αρχείο haarcascade_fullbody.xml.

### 3.3.2 Η συνάρτηση DetectMultiScale

Αφού έχει φορτωθεί ο ταξινομητής, επόμενο βήμα είναι η εφαρμογή του. Αυτό γίνεται με την χρήση της συνάρτησης detectMultiScale. Η detectMultiScale αναζητά το αντικείμενο προς ανίχνευση σε μία εικόνα και επιστρέφει την θέση του στην εικόνα, αν το εντοπίσει. Η συνάρτηση συντάσσεται ως εξής [41]:

***objects = <CascadeClassifier object>.detectMultiScale( image,***
***scaleFactor,***
***minNeihbors,***
***flags,***
***minSize,***
***maxSize)***

Παρακάτω παρουσιάζονται αναλυτικά οι παράμετροι που δέχεται η συνάρτηση.

- **objects**

Η παράμετρος *objects* είναι η λίστα ακεραίων που επιστρέφει η συνάρτηση. Οι ακέραιοι αυτοί ομαδοποιούνται νοητά σε τετράδες, όπου κάθε τετράδα αντιπροσωπεύει την θέση, το μήκος και το πλάτος ενός τετραγώνου που περιέχει το ζητούμενο αντικείμενο όπως φαίνεται και στην Εικόνα 3.4.


***Εικόνα 3.4:*** *Το πρώτο ζεύγος τιμών σε μια τετράδα ακεραίων της λίστας objects αντιστοιχεί στην θέση (x,y) του πάνω αριστερού pixel του τετραγώνου. Έπειτα ακολουθεί μια τιμή w για το πλάτος και μια τιμή h για ύψος του τετραγώνου.*

- **image**

Η εικόνα εισόδου στην οποία θα εφαρμοστεί ο αλγόριθμος ανίχνευσης η οποία έχει την μορφή ενός πίνακα δύο διαστάσεων.

- **scaleFactor**

Η παράμετρος αυτή ορίζει τον παράγοντα κατά τον οποίο μειώνεται το μέγεθος της εικόνας στην διαδικασία σχηματισμού της πυραμίδας εικόνων (image pyramid).

- **minNeihbors**

Κατά την διαδικασία αναζήτησης του αντικειμένου στην εικόνα από το ολισθαίνον παράθυρο, προκύπτουν αρκετές ανιχνεύσεις που μπορεί να σχετίζονται με τον ίδιο αντικείμενο στην εικόνα ή να είναι και λάθος. Αυτή η παράμετρος ορίζει τον ελάχιστο αριθμό γειτονικών τετραγώνων που πρέπει να έχει ένα υποψήφιο τετράγωνο για να το διατηρήσουμε. Ο λόγος που γίνεται αυτό είναι για να αυξηθεί η αξιοπιστία των ανιχνεύσεων. Στην Εικόνα 3.5 παρουσιάζεται ένα παράδειγμα για τη χρησιμότητα της παραμέτρου αυτής.

(a) (b)

***Εικόνα 3.5:*** *(a) Εφαρμογή του αλγορίθμου ανίχνευσης θέτοντας την τιμή της παραμέτρου minNeihbors ίση με το 0. Παρατηρούμε ότι προκύπτουν αρκετά τετράγωνα, μερικά από τα οποία είναι λανθασμένα. Αυτό είναι απόλυτα λογικό αφού δεν θέτουμε κανένα περιορισμό στον αλγόριθμο. (b) Εφαρμογή του αλγορίθμου ανίχνευσης θέτοντας την τιμή της παραμέτρου minNeihbors ίση με το 1. Παρατηρούμε ότι προκύπτει μόνο ένα τετράγωνο ανίχνευσης το οποίο είναι και σωστό [33].*

- **flags**

Αυτή η παράμετρος, αν ενεργοποιηθεί, χρησιμοποιεί τον Ανιχνευτή Ακμών Canny (Canny Edge Detector) για να απορρίψει περιοχές της εικόνας που δεν περιέχουν το ζητούμενο αντικείμενο. Κριτήριο για την απόρριψη μιας περιοχής αποτελεί το πλήθος των ακμών στην περιοχή αυτή. Αυτό αποσκοπεί στην επιτάχυνση της διαδικασίας ανίχνευσης.

- **minSize**

Ένα ζεύγος τιμών που ορίζει το ελάχιστο μέγεθος, μήκος και πλάτος, που μπορεί να έχει ένα αντικείμενο για να ανιχνευτεί. Αντικείμενα που έχουν μικρότερο μέγεθος αγνοούνται.

- **maxSize**

Επίσης ένα ζεύγος τιμών, το οποίο ορίζει το μέγιστο μέγεθος που μπορεί να έχει ένα αντικείμενο. Αντικείμενα μεγαλύτερου μεγέθους αγνοούνται.

Επιπλέον συναρτήσεις της OpenCV που χρησιμοποιήθηκαν ήταν η ***cv2.imread()*** για την ανάγνωση των εικόνων, η ***cv2.rectangle()*** για την σχεδίαση του τετραγώνου που επέστρεφε ο αλγόριθμος πάνω στην εικόνα και η ***cv2.imshow()*** για την εμφάνιση της εικόνας. Αξίζει να σημειωθεί ότι ο αλγόριθμος των Viola και Jones εφαρμόζεται σε ασπρόμαυρες εικόνες [22]. Για την μετατροπή των εικόνων από έγχρωμες σε ασπρόμαυρες χρησιμοποιήθηκε η συνάρτηση ***cv2.cvtColor()***.

## 3.4 Η μέθοδος των Dalal και Triggs

Η μέθοδος των Dalal και Triggs [10] ήταν η δεύτερη μέθοδος που υλοποιήθηκε. Παρακάτω παρουσιάζονται οι συναρτήσεις της OpenCV που χρησιμοποιήθηκαν.

### 3.4.1 Η συνάρτηση HOGDescriptor

Η συνάρτηση HOGDescriptor ήταν η βασική συνάρτηση που χρησιμοποιήθηκε στην υλοποίηση της παραπάνω μεθόδου. Η συνάρτηση αυτή επιστρέφει ένα περιγραφέα και ανιχνευτή τύπου HOG. Η σύνταξη της είναι η εξής [42]:

```
<descriptor> = cv2.HOGDescriptor(win_size,
                                 block_size,
                                 block_stride,
```

**cell_size,**

**nbins,**

**win_sigma,**

**threshold_L2hys,**

**gamma_correction,**

**nlevels)**

Παρακάτω παρουσιάζονται αναλυτικά οι παράμετροι της συνάρτησης HOGDescriptor. Οι τιμές που παίρνουν οι παράμετροι είναι αυτές που προτείνονται από τους Dalal και Triggs [10].

- **win_size**

Η παράμετρος αυτή ορίζει το μέγεθος του παραθύρου ανίχνευσης. Το μέγεθος αυτού του παραθύρου είναι 64x128 pixels.

- **block_size**

Ορίζεται το μέγεθος των blocks σε pixels. Το μέγεθος που υποστηρίζεται από την OpenCV είναι 16x16 pixels.

- **block_stride**

Το *block_stride* είναι ο ρυθμός ή το βήμα με το οποίο ένα block σαρώνει την εικόνα εισόδου. Η τιμή του είναι επίσης σε pixels και πρέπει να είναι πολλαπλάσιο του cell_size.

- **cell_size**

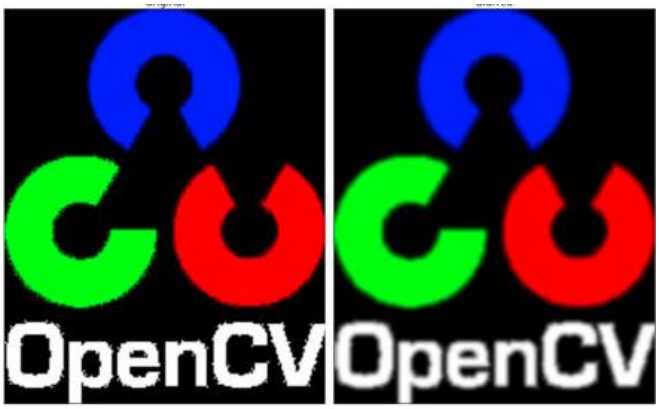
Η εικόνα εισόδου χωρίζεται σε κελιά (cells) και κάθε block περιέχει έναν αριθμό από τέτοια κελιά. Η παράμετρος αυτή ορίζει το μέγεθος των κελιών σε pixels. Η OpenCV υποστηρίζει μόνο κελιά μεγέθους 8x8 pixels.

- **nbins**

Η τιμή της παραμέτρου *nbins* ορίζει τις θέσεις του ιστογράμματος κλίσεων. Η μόνη τιμή που υποστηρίζεται είναι η τιμή 9 δηλαδή ένα ιστόγραμμα 9 θέσεων.

- **win_sigma**

Η μέθοδος χρησιμοποιεί ένα Γκαουσιανό Φίλτρο Εξομάλυνσης (Gaussian Smoothing Filter) και η παράμετρος αυτή ορίζει την τιμή *σ* της καμπύλης Gauss. Η χρήση ενός τέτοιου φίλτρου γίνεται για να μειωθεί ο θόρυβος όπως φαίνεται και στην Εικόνα 3.6. Μπορεί κανείς να παρατηρήσει ότι αφού έχει εφαρμοστεί το φίλτρο, η εικόνα γίνεται πιο θολή και οι ακμές πιο ομαλές.




***Εικόνα 3.6:*** *Στην εικόνα δεξιά απεικονίζεται το αποτέλεσμα της εφαρμογής του Gaussian Smoothing Filter στην αριστερή εικόνα. Η τελική εικόνα έχει γίνει πιο θολή και οι ακμές της δεν είναι τόσο έντονες [43].*

- **threshold_L2hys**

Η τιμή αυτής της παραμέτρου χρησιμοποιείται για την κανονικοποίηση του διανύσματος χαρακτηριστικών. Το σχήμα κανονικοποίησης που χρησιμοποιείται λέγεται L2-Hys (Lowe-style clipped L2 norm), που είναι μία παραλλαγή της ευκλείδειας νόρμας [10]. Με την παράμετρο αυτή ορίζεται ένα ανώτατο όριο το οποίο δεν μπορούν να ξεπεράσουν οι τιμές του διανύσματός χαρακτηριστικών. Συνήθως η τιμή αυτή είναι 0,2.

- **gamma_correction**

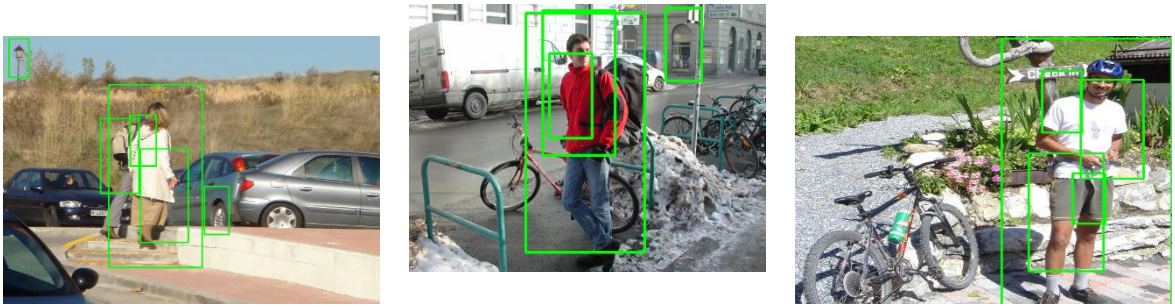
Η παράμετρος αυτή είναι τύπου Boolean και ορίζει το αν απαιτείται η προεπεξεργασία *gamma correction* ή όχι στην εικόνα εισόδου. *Το gamma correction* ή *gamma* είναι μια λειτουργία που σχετίζεται με την κωδικοποίηση και αποκωδικοποίηση της φωτεινότητας σε μία εικόνα [44].

- **nlevels**

Σε κάθε εικόνα η διαδικασία της ανίχνευσης γίνεται σε πολλαπλές κλίμακες. Για ένα δεδομένο παράγοντα προσαύξησης της κλίμακας, η παράμετρος *nlevels* ορίζει τον μέγιστο αριθμό των επαναλήψεων που θα γίνει η διαδικασίας αυτή.

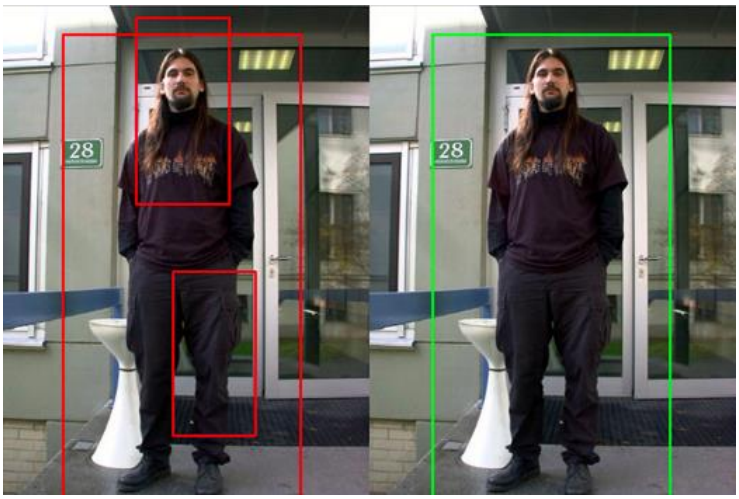
### 3.4.2 Non-Maximum Suppression

Σε συνδυασμό με την παραπάνω μέθοδο υλοποιήθηκε και ο αλγόριθμος του Non-Maximum Suppression [45] ο οποίος βασίστηκε στην υλοποίηση των Felzenszwalb κ.α. [46]. Η μέθοδος των Dalal και Triggs επιστρέφει πολλαπλές ανιχνεύσεις για την εμφάνιση του ίδιου αντικειμένου σε μια εικόνα όπως φαίνεται στην Εικόνα 3.7 [46]. Ο αλγόριθμος NMS (Non-Maximum Suppression) είναι ένας αλγόριθμος μετεπεξεργασίας σκοπός του οποίου είναι η συγχώνευση όλων των ανιχνεύσεων που αφορούν το ίδιο αντικείμενο [47].



***Εικόνα 3.7:*** *Κάποια παραδείγματα εικόνων στις οποίες γίνονται πολλαπλές ανιχνεύσεις για τον ίδιο άνθρωπο.*

Ο αλγόριθμος δέχεται ως είσοδο μία λίστα από τετράγωνα ανιχνεύσεων (bounding boxes) για μια δεδομένη εικόνα και ελέγχει το ποσοστό της επικάλυψης μεταξύ των τετραγώνων ανά ζεύγη. Αν η επικάλυψη ξεπερνάει κάποια τιμή κατωφλίου η οποία συνήθως ορίζεται στο 0.5, τότε ένα από τα τετράγωνα αγνοείται. Τέλος επιστρέφεται μια λίστα με τα τελικά τετράγωνα ανιχνεύσεων που έχει κρατήσει ο αλγόριθμος. Τα αποτελέσματα της παραπάνω διαδικασίας παρουσιάζονται στην Εικόνα 3.8.




***Εικόνα 3.8:*** *Αριστερά απεικονίζονται τα αποτελέσματα της διαδικασίας ανίχνευσης χωρίς την εφαρμογή του αλγορίθμου NMS. Δεξιά φαίνονται τα αποτέλεσμα της διαδικασίας ανίχνευσης μετά της εφαρμογή του αλγορίθμου NMS [48].*

## 3.5 Σύνολο Δεδομένων

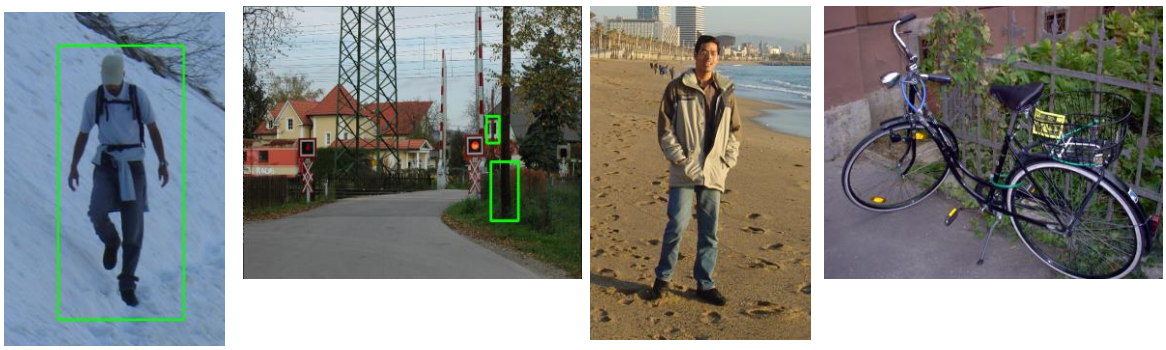
Τα τελευταία χρόνια υπάρχει ένας μεγάλος αριθμός από σύνολα δεδομένων που έχουν γίνει διαθέσιμα για την εκπαίδευση και την αξιολόγηση αλγορίθμων που σχετίζονται με την ανίχνευση ανθρώπων. Αυτά τα σύνολα έχουν δημιουργηθεί από διάφορα σενάρια και μπορούν να χρησιμοποιηθούν για διάφορες εφαρμογές [49].

Για την αξιολόγηση των παραπάνω αλγορίθμων, χρησιμοποιήθηκαν εικόνες από το INRIA Pedestrian Dataset το οποίο περιέχει την μπροστινή ή την πίσω όψη ανθρώπων σε όρθια στάση. Πιο συγκεκριμένα χρησιμοποιήθηκαν συνολικά 588 δείγματα εικόνων εκ των οποίων τα 300 ήταν αρνητικά, δηλαδή δεν περιείχαν κάποιον άνθρωπο, ενώ τα 288 ήταν θετικά, περιείχαν δηλαδή τουλάχιστον έναν άνθρωπο. Συνολικά οι άνθρωποι που εμφανίζονται στα θετικά δείγματα εικόνων είναι 598.

## 3.6. Αξιολόγηση Μεθόδων

Κάθε αλγόριθμος ανίχνευσης «σκανάρει» μία εικόνα και κάθε υποπαράθυρο της εικόνας που προκύπτει από την διαδικασία αυτή ταξινομείται ως Θετικό ή Αρνητικό

ανάλογα με το αν περιέχει ή όχι το ζητούμενο αντικείμενο. Υπάρχουν τέσσερις κατηγορίες ταξινομήσεων. Μια Αληθώς Αρνητική (True Negative) ταξινόμηση υποδηλώνει την μη εμφάνιση του ζητούμενου αντικειμένου σε μια εικόνα που αυτό δεν υπάρχει σε αντίθεση με μία Αληθώς Θετική (True Positive) ταξινόμηση, που υποδηλώνει την εμφάνιση του ζητούμενου αντικειμένου σε μια εικόνα που αυτό υπάρχει. Αυτές οι δύο κατηγορίες ταξινομήσεων είναι και οι ιδανικές, γιατί όταν μια εικόνα δεν περιέχει κάποιον άνθρωπο ο αλγόριθμος δεν ανιχνεύει τίποτα, ενώ όταν μια εικόνα περιέχει έναν άνθρωπο, ο αλγόριθμος τον εντοπίζει. Οι άλλες δύο κατηγορίες αφορούν εσφαλμένες ταξινομήσεις. Πιο συγκεκριμένα μία Ψευδώς Αρνητική (False Negative) ταξινόμηση δηλώνει την απουσία του ζητούμενου αντικειμένου σε μία εικόνα που αυτό υπάρχει και μια Ψευδώς Θετική (False Positive) ταξινόμηση δηλώνει την παρουσία ενός αντικειμένου σε μια εικόνα που αυτό δεν υπάρχει. Στην Εικόνα 3.9 παρουσιάζονται οι τέσσερις κατηγορίες ταξινομήσεων. Παρατηρούμε ότι στην Εικόνα 3.9a που αναπαριστά μια Αληθώς Θετική ταξινόμηση, ο αλγόριθμος έχει ανιχνεύσει σωστά έναν άνθρωπο, ενώ στην Εικόνα 3.9b που αναπαριστά μια Ψευδώς Θετική ταξινόμηση, ο αλγόριθμος προέβλεψε λάθος την παρουσία ανθρώπου. Από την άλλη, στην Εικόνα 3.9c ο αλγόριθμος δεν κατάφερε να ανιχνεύσει τον άνθρωπο της εικόνας, για αυτό η ταξινόμηση είναι Ψευδώς Αρνητική. Στην Εικόνα 3.9d πολύ σωστά ο ανιχνευτής δεν επέστρεψε κανένα αποτέλεσμα, καθώς δεν υπάρχει κάποιος άνθρωπος στην εικόνα αυτή. Η ταξινόμηση σε αυτή την περίπτωση θεωρείται Αληθώς Αρνητική.



(a) (b) (c) (d)

***Εικόνα 3.9:*** *a) Παράδειγμα Αληθώς Θετικής ταξινόμησης. b) Παράδειγμα Ψευδώς Θετικής ταξινόμησης. c) Παράδειγμα Ψευδώς Αρνητικής ταξινόμησης. d) Παράδειγμα Αληθώς Αρνητικής ταξινόμησης.*

Για την οπτικοποίηση των αποτελεσμάτων και την αξιολόγηση ενός ταξινομητή χρησιμοποιήθηκαν επίσης οι τιμές ακρίβειας (precision) και επανάκλησης (recall) [50].

Ως ακρίβεια (precision) ορίζεται ο λόγος των Αληθώς Θετικών ταξινομήσεων προς το σύνολο των Θετικών ταξινομήσεων. Ουσιαστικά η ακρίβεια είναι ένα μέτρο που δείχνει πόσες από τις θετικές ανιχνεύσεις ήταν σωστές. Ο τύπος της είναι:

$$precision = \frac{TruePositives}{TruePositives+FalsePositives}$$

Η επανάκληση (recall) εκφράζει των λόγο των Αληθώς Θετικών ταξινομήσεων προς το σύνολο όλων των θετικών δειγμάτων και υποδηλώνει πόσα θετικά δείγματα ανιχνευτήκαν. Ο τύπος της επανάκλησης είναι:

$$recall = \frac{TruePositives}{TruePositives+FalseNegatives}$$

Για την σύγκριση των αλγορίθμων ως προς την απόδοση τους στην ανίχνευση ανθρώπων χρησιμοποιήθηκαν επίσης οι καμπύλες του Ρυθμού των Ψευδώς Θετικών ταξινομήσεων και του Ρυθμού των Αληθώς Θετικών ταξινομήσεων. Οι τύποι τους ορίζονται αντίστοιχα [50]:

$$False\ Positive\ Rate = \frac{FalsePositives}{FalsePositives+TrueNegatives}$$

$$True\ Positive\ Rate = \frac{TruePositives}{TruePositives+FalseNegatives}$$

Αξίζει να σημειωθεί ότι ο λόγος Αληθώς Θετικών ταξινομήσεων εκφράζει το ίδιο μέγεθος με την επανάκληση.

# ΚΕΦΑΛΑΙΟ 4: Πειραματική Διαδικασία και Αποτελέσματα

## 4.1 Περιγραφή πειραματικής διαδικασίας

Για να μελετηθεί η απόδοση που είχαν οι αλγόριθμοι στην ανίχνευση ανθρώπων έγινε μία σειρά από πειράματα, στα οποία κάθε αλγόριθμος εφαρμόστηκε στο ίδιο σύνολο από εικόνες. Έπειτα έγινε καταμέτρηση του αριθμού των ανιχνεύσεων και εξετάστηκε ποιες από τις ανιχνεύσεις ήταν σωστές και ποιες όχι. Στην συνέχεια οι τιμές που προέκυψαν χρησιμοποιήθηκαν για την δημιουργία γραφημάτων για την καλύτερη αναπαράσταση των αποτελεσμάτων. Παράλληλα μετρήθηκε και ο χρόνος που χρειάστηκε ο κάθε αλγόριθμος για να τελέσει το έργο της ανίχνευσης σε κάθε περίπτωση.

## 4.2 Αποτελέσματα

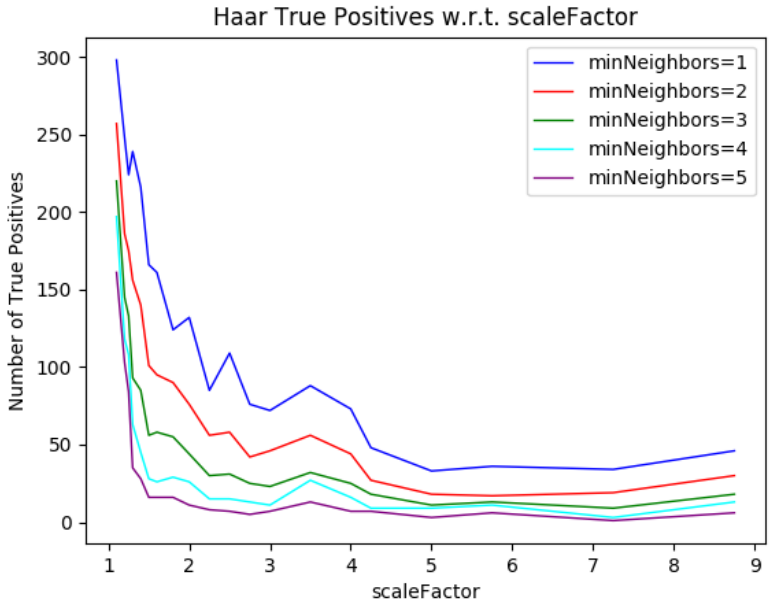
Παρακάτω παρουσιάζονται τα αποτελέσματα που προέκυψαν από τις πειραματικές δοκιμές.

### 4.2.1 Η μέθοδος των Viola και Jones

Πρώτη εξετάστηκε η μέθοδος των Viola και Jones. Μια από τις παραμέτρους που μεταβάλλονταν κατά την διαδικασία των πειραμάτων, ήταν η παράμετρος *scaleFactor*, η οποία ορίζει τον παράγοντα κατά τον οποίο μειώνεται το μέγεθος της εικόνας εισόδου, για τον σχηματισμό της πυραμίδας εικόνων (βλ. Εικόνα 3.3). Το εύρος τιμών της παραμέτρου αυτής ξεκινούσε από την ελάχιστη δυνατή τιμή, η οποία ήταν το 1.1 και έφτανε μια μέγιστη τιμή που ορίσθηκε το 8.75, ύστερα από την παρατήρηση ότι η απόδοση σταθεροποιείται από την τιμή αυτή και έπειτα. Επιπλέον, η μέθοδος αυτή εξετάστηκε και ως προς την παράμετρο *minNeighbors*, του αριθμού των ελαχίστων γειτονικών τετραγώνων, που σχετίζεται με την απαλοιφή πολλαπλών ανιχνεύσεων που αφορούν την εμφάνιση του ίδιου αντικειμένου στην εικόνα.

Στο Διάγραμμα 4.1 παρουσιάζονται οι τιμές των Αληθώς Θετικών (True Positive) ανιχνεύσεων ως προς τις τιμές της παραμέτρου *scaleFactor*, για πέντε τιμές της παραμέτρου *minNeighbors*. Σε κάθε περίπτωση, ο αριθμός των Αληθώς Θετικών

ανιχνεύσεων παίρνει την μέγιστη τιμή του όταν ο παράγοντας της κλίμακας παίρνει την ελάχιστη τιμή η οποία ισούται με 1.1. Παρατηρούμε επίσης ότι το πλήθος των σωστών ανιχνεύσεων παρουσιάζει γενικά μια φθίνουσα συμπεριφορά και ο λόγος έχει να κάνει με την πυραμίδα εικόνων (image pyramid). Κάθε νέο επίπεδο της πυραμίδας δημιουργείται μέχρι η εικόνα να γίνει μικρότερη από το παράθυρο ανίχνευσης. Ο παράγοντας της κλίμακας ορίζει το βήμα με το οποίο γίνεται η διαδικασία αυτή, συνεπώς μια μεγάλη τιμή, ισοδυναμεί με ένα γρήγορο σκανάρισμα της εικόνας, ενώ μια μικρή τιμή ισοδυναμεί με έναν πιο αργό και λεπτομερή έλεγχο της εικόνας για το αν περιέχει ή όχι το αντικείμενο ενδιαφέροντος. Καλύτερος έλεγχος της εικόνας συνεπάγεται περισσότερες ανιχνεύσεις.

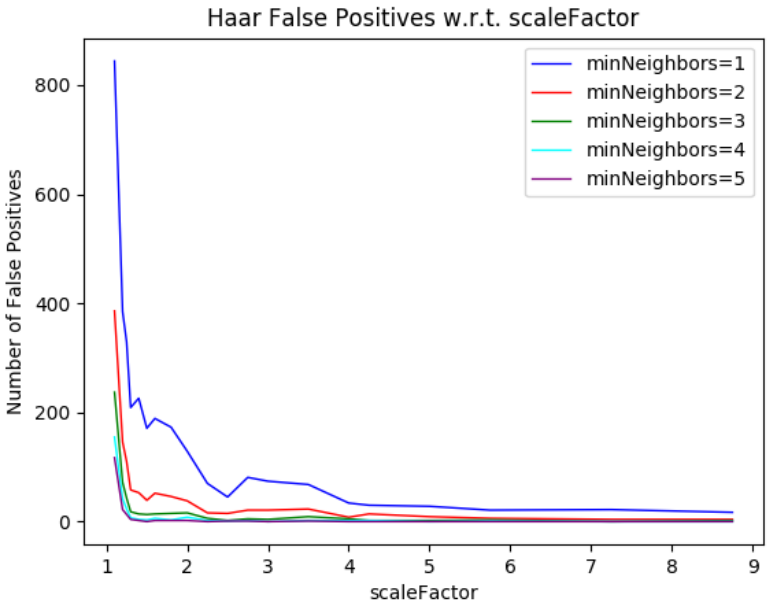



***Διάγραμμα 4.1****: Το πλήθος των Αληθώς Θετικών Ανιχνεύσεων (True Positives) ως προς την παράμετρο scaleFactor, για πέντε τιμές της παραμέτρου minNeighbors του αλγορίθμου των Viola και Jones.*

Αυτός είναι λοιπόν και ο λόγος που όσο αυξάνεται η τιμή του παράγοντα της κλίμακας, ο αριθμός των True Positives μειώνεται. Παρ ’όλα αυτά, οι καμπύλες παρουσιάζουν τοπικά μέγιστα, το οποίο σημαίνει ότι ο αλγόριθμος μπορεί να έχει καλύτερη απόδοση για μία τιμή του παράγοντα, σε σχέση με μία προηγούμενη. Παρατηρείται επίσης ότι όταν ο παράγοντας παίρνει τιμές μεγαλύτερες του 4, το πλήθος των True Positives αρχίζει να σταθεροποιείται. Τέλος, τις περισσότερες σωστές ανιχνεύσεις ο αλγόριθμος τις πετυχαίνει

όταν η παράμετρος *minNeighbors* είναι ίση με 1 ενώ τις λιγότερες όταν είναι ίση με 5. Η επίδραση της παραμέτρου αυτής αναλύεται παρακάτω, στα Διαγράμματα 4.5 και 4.6.

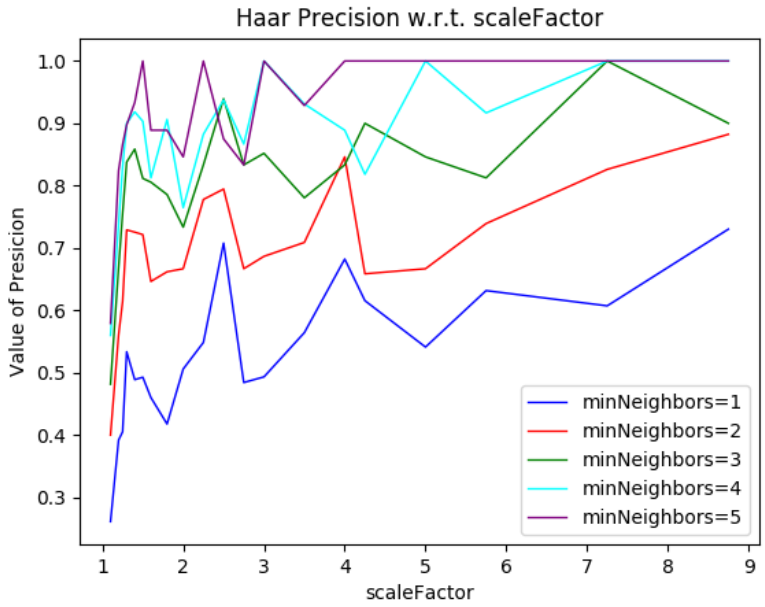
Ενδιαφέρον είχε η συμπεριφορά των False Positives. Το Διάγραμμα 4.2 απεικονίζει το πλήθος των Ψευδώς Θετικών ανιχνεύσεων ως προς την παράμετρο *scaleFactor*. Οι λάθος ανιχνεύσεις, όπως οι σωστές, μεγιστοποιούνται στην αρχική τιμή του παράγοντα της κλίμακας και έπειτα φθίνουν, παρουσιάζοντας τοπικά ακρότατα στα ίδια σημεία. Ο αριθμός τους επίσης παρουσιάζει μια σταθεροποίηση όταν η παράμετρος *scaleFactor* παίρνει τιμές μεγαλύτερες του 4. Παρόμοια επίδραση είχε και η παράμετρος *minNeighbors*, αφού τα περισσότερα False Positives προσμετρήθηκαν όταν ήταν ίση με 1 ενώ τα λιγότερα όταν ισούνταν με 5.




***Διάγραμμα 4.2:*** *Το πλήθος των Ψευδώς Θετικών Ανιχνεύσεων (False Positives) ως προς την παράμετρο scaleFactor, για πέντε τιμές της παραμέτρου minNeighbors του αλγορίθμου των Viola και Jones.*

Όπως έχει προαναφερθεί, ο παράγοντας της κλίμακας επηρεάζει τον αριθμό των συνολικών ανιχνεύσεων κι αυτός είναι ο λόγος που παρατηρείται αυτή η συμπεριφορά στον αριθμό των False Positives. Δηλαδή το γεγονός ότι όσο μεγαλώνει η τιμή της κλίμακας, μειώνονται ταυτόχρονα και οι σωστές και οι λάθος ανιχνεύσεις, σημαίνει ότι μειώνεται γενικά ο αριθμός των τετραγώνων ανίχνευσης που επιστρέφει ο αλγόριθμος.

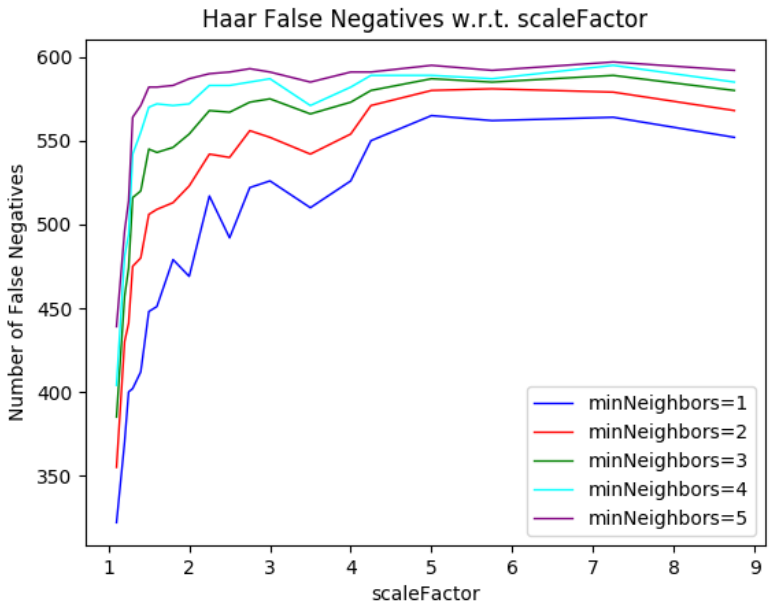
Αυτό αποδεικνύεται στα Διαγράμματα 4.3 και 4.4 παρακάτω. Στο Διάγραμμα 4.3 ο κάθετος άξονας αναπαριστά την τιμή της ακρίβειας, που εκφράζει το πόσες από τις συνολικές ανιχνεύεις ήταν σωστές, ενώ ο οριζόντιος τις τιμές της παραμέτρου *scaleFactor*. Η ακρίβεια γενικά παρουσιάζει μια ασταθή αλλά ανοδική πορεία για όλες τις τιμές της παραμέτρου *minNeighbors*. Παρατηρούνται επίσης έντονες διαφορές για γειτονικές τιμές της παραμέτρου *scaleFactor*. Σε πολλές περιπτώσεις, όπως για παράδειγμα όταν η παράμετρος *minNeighbors* είναι ίση με 5, η τιμή της ακρίβειας φτάνει ή τείνει να φτάσει την μέγιστη τιμή που είναι ίση με 1, το οποίο σημαίνει ότι το 100% των ανιχνεύσεων ήταν σωστές. Αυτό όμως δεν ερμηνεύεται απαραίτητα ως δείγμα καλής απόδοσης αλλά ως μια ακόμα ένδειξη για το πόσο μικρός είναι ο αριθμός των συνολικών ανιχνεύσεων.




***Διάγραμμα 4.3:*** *Η τιμή της Ακρίβειας (Precision) ως προς την παράμετρο scaleFactor, για πέντε τιμές της παραμέτρου minNeighbors του αλγορίθμου των Viola και Jones.*

Πράγματι, αυτό επιβεβαιώνεται και στο Διάγραμμα 4.4, στο οποίο παρουσιάζεται ο αριθμός των Ψευδώς Αρνητικών ως προς τις τιμές της *scaleFactor*, το πλήθος δηλαδή των ανθρώπων, από το σύνολο των εικόνων, που δεν ανιχνεύτηκαν. Ο αλγόριθμος που κατάφερε να ανιχνεύσει του περισσότερους ανθρώπους ήταν αυτός με την τιμή της παραμέτρου *minNeighbors* ίση με 1, ενώ τους λιγότερους αυτός με τιμή *minNeighbors* ίση με 5. Γενικά, όσο αυξάνονται οι τιμές των παραμέτρων *scaleFactor* και *minNeighbors* τόσο

λιγότεροι άνθρωποι ανιχνεύονται. Αυτό είναι άλλη μια απόδειξη για το ότι η τιμή της ακρίβειας δεν αντικατοπτρίζει την καθολική επίδοση του αλγορίθμου καθώς στις περιπτώσεις που η ακρίβεια παίρνει πολύ υψηλές τιμές, οι άνθρωποι που δεν ανιχνεύονται είναι πάρα πολλοί.

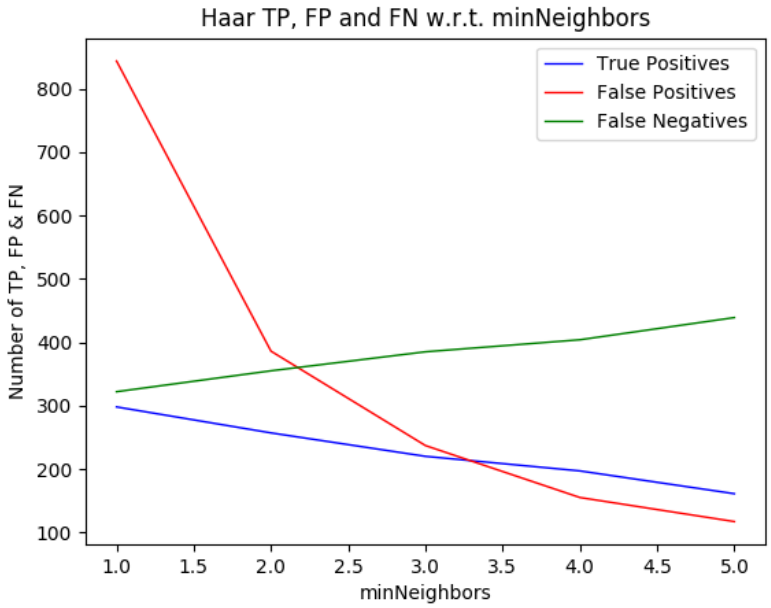



***Διάγραμμα 4.4:*** *Το πλήθος των Ψευδώς Αρνητικών Ανιχνεύσεων (False Negatives) ως προς την παράμετρο scaleFactor, για πέντε τιμές της παραμέτρου minNeighbors του αλγορίθμου των Viola και Jones.*

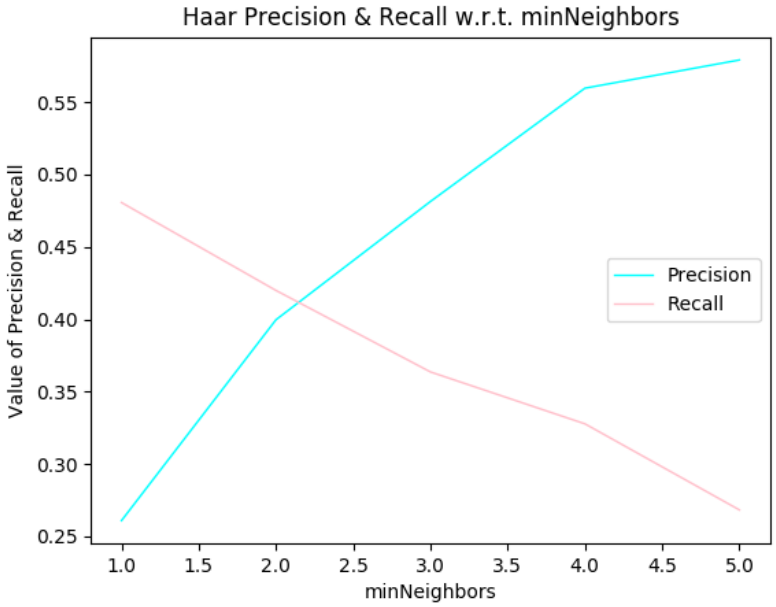
Μία ακόμα παράμετρος που επηρέασε την επίδοση του αλγορίθμου ήταν η παράμετρος *minNeighbors*. Η παράμετρος αυτή ορίζει, δεδομένου ότι ο αλγόριθμος επιστρέφει πολλαπλές ανιχνεύσεις για το ζητούμενο αντικείμενο, τον ελάχιστο αριθμό γειτονικών ανιχνεύσεων, που πρέπει να έχει μια ανίχνευση, για να θεωρηθεί σωστή. Η επίδραση αυτής της παραμέτρου απεικονίζεται στα Διαγράμματα 4.5 και 4.6 για μία δεδομένη τιμή του παράγοντα μεταβολής της κλίμακας που επιλέχθηκε τυχαία. Στο Διάγραμμα 4.5 ο κάθετος άξονας αντιστοιχεί στο πλήθος των Αληθώς Θετικών (True Positives), Ψευδώς Θετικών (False Positives) και Ψευδώς Αρνητικών (False Negatives) ανιχνεύσεων, ενώ στο Διάγραμμα 4.6 στις τιμές της Ακρίβεια (Precision) και της Επανάκλησης (Recall). Ο οριζόντιος άξονας και στα δύο διαγράμματα εκφράζει τις τιμές της παραμέτρου *minNeighbors*.

Παρατηρούμε από το Διάγραμμα 4.5 ότι καθώς αυξάνεται η τιμή του *minNeighbors*, μειώνονται τόσο οι Αληθώς Θετικές όσο και οι Ψευδώς Θετικές ανιχνεύσεις. Παράλληλα η τιμή της Ακρίβειας στο Διάγραμμα 4.6 ακολουθεί ανοδική πορεία, το οποίο όμως δεν συνεπάγεται καλύτερη επίδοση καθώς παρόμοια πορεία ακολουθεί και η τιμή των Ψευδώς Αρνητικών, των ανθρώπων δηλαδή που δεν ανιχνεύτηκαν. Την μείωση της επίδοσης επιβεβαιώνει και η καθοδική πορεία της Επανάκλησης (Recall), που εκφράζει το ποσοστό των ανθρώπων που ανιχνεύτηκαν.

Γενικότερα, για μια δεδομένη τιμή της παραμέτρου *scaleFactor*, η αύξηση της τιμής της παραμέτρου *minNeighbors* οδηγεί σε μείωση του συνολικού αριθμού ανιχνεύσεων. Αυτό είναι απόλυτα λογικό, καθώς αυξάνοντας την τιμή αυτής της παραμέτρου, θέτουμε στον αλγόριθμο πιο αυστηρά κριτήρια για να επιστρέψει ένα παράθυρο ανίχνευσης. Αυτό έχει ως αποτέλεσμα να μειωθεί ο αριθμός των παραθύρων που επιστρέφονται.




***Διάγραμμα 4.5:*** *Οι τιμές του πλήθους των Αληθώς Θετικών (True Positives), των Ψευδώς Θετικών (False Positives) και των Ψευδώς Αρνητικών (False Negatives) ως προς την παράμετρο minNeighbors για τον αλγόριθμο των Viola και Jones.*




***Διάγραμμα 4.6:*** *Οι τιμές της Ακρίβειας (Presicion) και της Επανάκλησης (Recall) ως προς την παράμετρο minNeighbors.*

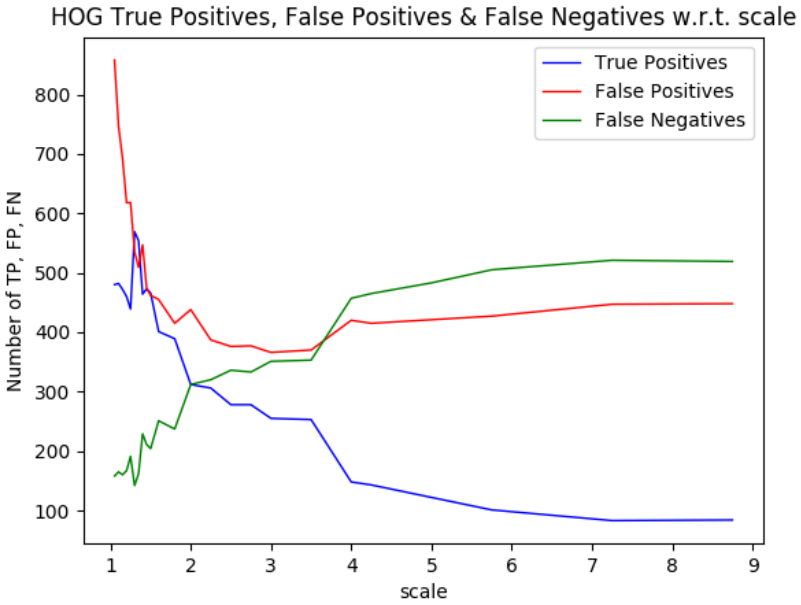
### 4.2.2 Η μέθοδος των Dalal και Triggs

Η επόμενη μέθοδος που εξετάστηκε ήταν αυτή των Dalal και Triggs. Υλοποιήθηκαν δύο εκδοχές της μεθόδου αυτής, μία με τον επιπρόσθετο αλγόριθμο του Non-Maximum Suppression και μια χωρίς. Στην δουλεία τους οι Dalal και Triggs [1] συνδυάζουν τον ανιχνευτή τους με τον παραπάνω αλγόριθμο ο οποίος ουσιαστικά συγχωνεύει πολλαπλές ανιχνεύσεις του ίδιου αντικειμένου. Οι δύο υλοποιήσεις έγιναν για μια καλύτερη μελέτη της απόδοσης του ανιχνευτή.

Η παράμετρος που μεταβάλλονταν ήταν η παράμετρος *scale,* που όπως και η *scaleFactor*, σχετίζεται με τον σχηματισμό της πυραμίδας εικόνων (βλ. Εικόνα 3.3). Πιο συγκεκριμένα, η *scale* ορίζει τον λόγο κατά τον οποίο μεγαλώνει το παράθυρο ανίχνευσης και το εύρος τιμών της ήταν από 1.05 έως 8.75.

Από τις δύο εκδοχές, πρώτη μελετήθηκε αυτή χωρίς τον αλγόριθμο Non-Maximum Suppression. Στο Διάγραμμα 4.7 απεικονίζεται το πλήθος των True Positives, False Positives και False Negatives για το εύρος τιμών της παραμέτρου *scale*. Παρατηρείται ότι καθώς αυξάνεται η τιμή της παραμέτρου *scale*, το πλήθος των Αληθώς Θετικών (True Positives) και Ψευδώς Θετικών (False Positives) μειώνεται, ενώ το πλήθος των Ψευδώς Αρνητικών (False Negatives) αυξάνεται. Αυτό σημαίνει ότι η αύξηση της παραμέτρου *scale* έχει ως

αποτέλεσμα την μείωση των σωστών ανιχνεύσεων. Ο λόγος που συμβαίνει αυτό είναι γιατί, όπως και στην προηγούμενη μέθοδο, μεγάλη τιμή στον παράγοντα της κλίμακας συνεπάγεται γρήγορος έλεγχος μιας εικόνας, οπότε είναι λογικό να οδηγηθούμε σε λιγότερες ανιχνεύσεις. Σε αντίθεση με πριν, σε αυτή την μέθοδο ο αριθμός των False Positives δεν ακολουθεί τον αριθμό των True Positives και ο λόγος έχει να κάνει με το πλήθος των παραθύρων ανίχνευσης (detection windows), όπως αναλύεται παρακάτω. Επίσης από την τιμή 4 της παραμέτρου *scale* και έπειτα η απόδοση του αλγορίθμου δεν παρουσιάζει ραγδαίες μεταβολές και αρχίζει να σταθεροποιείται.

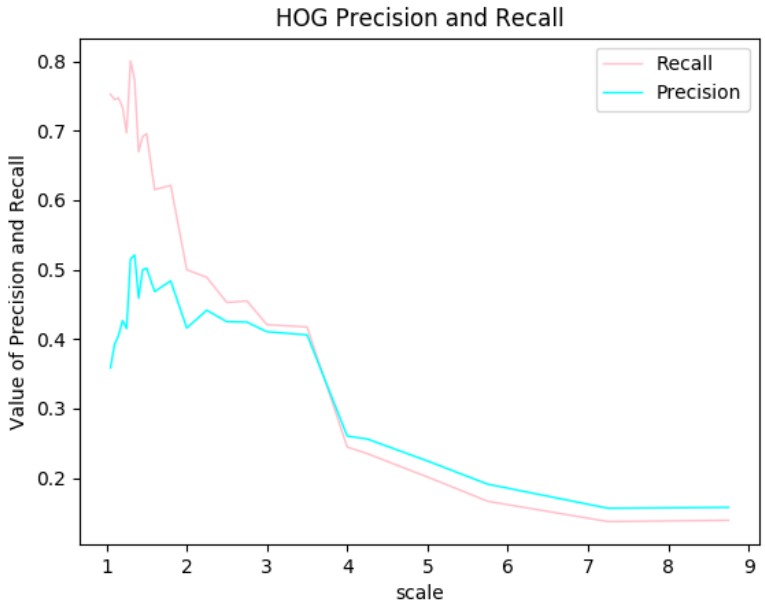



***Διάγραμμα 4.7:*** *Ο αριθμός των True Positives, False Positives και False Negatives ως προς την παράμετρο scale του αλγορίθμου HOG.*

Κάτι ακόμα που αξίζει να αναφερθεί, είναι τα ακρότατα των καμπυλών στο παραπάνω διάγραμμα. Φαίνεται πως ο αλγόριθμος έχει την καλύτερη επίδοση όταν η παράμετρος *scale* ισούται με 1.3, αφού για εκείνη την τιμή ο αριθμός των True Positives παίρνει την μέγιστη τιμή του ενώ ο αριθμός των False Negatives την ελάχιστη. Αυτό σημαίνει πως τότε πραγματοποιούνται οι περισσότερες σωστές ανιχνεύσεις.

Παρακάτω παρατίθεται το Διάγραμμα 4.8 για τις τιμές της ακρίβειας (Precision) και της επανάκλησης (Recall) ως προς της παράμετρο *scale*. Μπορεί κανείς να παρατηρήσει ότι η ακρίβεια και η επανάκληση ακολουθούν μια κοινή φθίνουσα πορεία κι αυτό συμβαίνει επειδή οι σωστές ανιχνεύσεις μειώνονται. Επιπλέον παρατηρήθηκε ότι παρόλο που η ικανότητα ανίχνευσης του αλγορίθμου μειώνεται, μειώνοντας έτσι τον αριθμό των

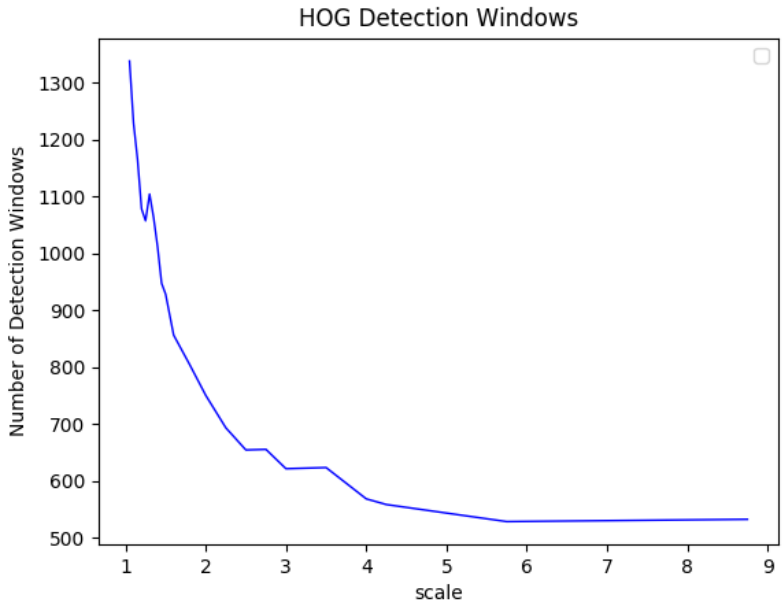
σωστών ανιχνεύσεων, ο αριθμός τετραγώνων ανίχνευσης που επιστρέφει ο αλγόριθμος δεν σημειώνει σημαντική πτώση. Πράγματι από την τιμή 6 του παράγοντα κλίμακας κι έπειτα όπως φαίνεται και στο Διάγραμμα 4.9, όπου απεικονίζεται ο αριθμός των παραθύρων ανίχνευσης (detection windows) ως προς την παράμετρο *scale*, ο συνολικός αριθμός τετραγώνων που επέστρεφε ο αλγόριθμος παραμένει σχεδόν σταθερός.




***Διάγραμμα 4.8****: Οι τιμές της ακρίβεια και της επανάκλησης ως προς την παράμετρο scale για τον αλγόριθμο HOG.*

Μια ερμηνεία που μπορεί να δοθεί για αυτό είναι η επίδραση της παραμέτρου *nLevels* της συνάρτησης HOGDescriptor. Αυτή η παράμετρος ορίζει τον αριθμό των επιπέδων της πυραμίδας που σχηματίζουν οι εικόνες (image pyramid), μια διαδικασία που, όπως έχει αναφερθεί, γίνεται για τον εντοπισμό ανθρώπων σε διάφορες κλίμακες. Ενώ στην μέθοδο των Viola και Jones η διαδικασία της δημιουργίας των επιπέδων σταματάει όταν το μέγεθος του τετραγώνου ανίχνευσης ξεπεράσει αυτό της εικόνας, στην προκειμένη περίπτωση ο αριθμός των επιπέδων είναι προκαθορισμένος. Αυτό έχει ως αποτέλεσμα οι σωστές ανιχνεύσεις να μειώνονται αλλά ο αριθμός των τετραγώνων που επιστρέφει ο αλγόριθμος να μην ακολουθεί τον ίδιο ρυθμό μείωσης επηρεάζοντας έτσι των λόγο των True Positives προς τον συνολικό αριθμό ανιχνεύσεων. Για αυτό λοιπόν η Ακρίβεια και η Επανάκληση ακολουθούν καθοδική πορεία. Αυτό δικαιολογεί επίσης την αύξηση του αριθμού των False Positives όπως αυτή παρουσιάστηκε στο Διάγραμμα 4.7 παραπάνω.

Τέλος, το γεγονός ότι η τιμή 1.3 της παραμέτρου *scale* επιφέρει την μέγιστη απόδοση αποδεικνύεται και πάλι, καθώς και οι δύο καμπύλες παρουσιάζουν μέγιστο στην τιμή αυτή.




***Διάγραμμα 4.9:*** *Το πλήθος των παραθύρων ανίχνευσης που επέστρεφε ο αλγόριθμος ως προς τον παράγοντα κλίμακας scale του αλγορίθμου HOG.*

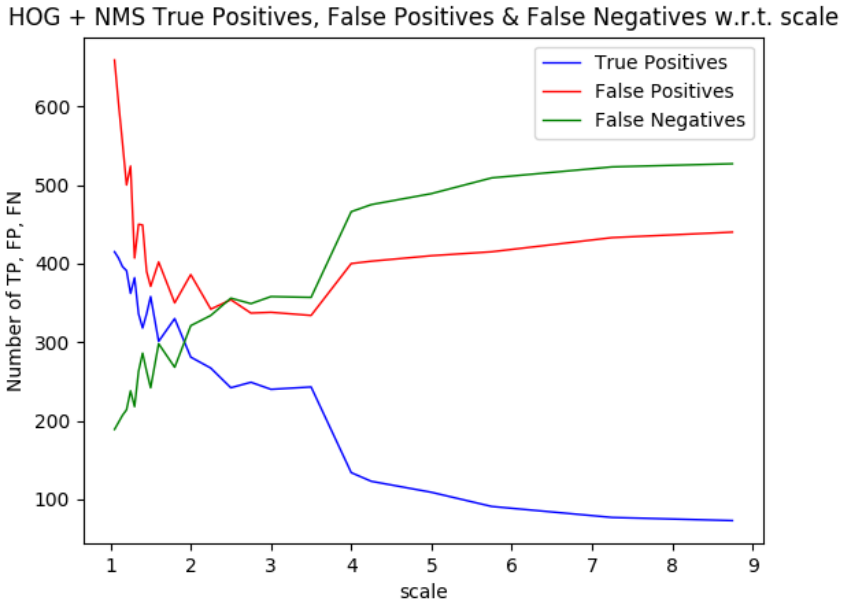
### 4.2.3 Η μέθοδος των Dalal και Triggs με εφαρμογή του αλγορίθμου NMS

Τα αποτελέσματα της εφαρμογής του επιπλέον αλγορίθμου Non Maximum Suppression στον αλγόριθμο HOG παρουσιάζονται στο Διάγραμμα 4.10. Στον οριζόντιο άξονα παρατίθενται οι τιμές της παραμέτρου *scale* και στον κάθετο το πλήθος των False Positives, True Positives και False Negatives.

Γενικά, η πορεία του αριθμού των Ψευδώς Θετικών, Αληθώς Θετικών και Ψευδώς Αρνητικών ανιχνεύσεων δεν διαφέρει από αυτή του προηγούμενο αλγορίθμου. Πιο συγκεκριμένα, η αύξηση της παραμέτρου *scale* επιφέρει μείωση των σωστών ανιχνεύσεων, κι αυτός είναι ο λόγος που η τιμή των True Positives έχει καθοδική πορεία. Η αύξηση των False Negatives ερμηνεύεται επίσης ως αδυναμία του αλγορίθμου στην ανίχνευση, καθώς όσο αυξάνεται ο παράγοντας μεταβολής της κλίμακας τόσο αυξάνεται κι ο αριθμός των ανθρώπων που δεν ανιχνεύονται. Όπως και πριν, η παράμετρος *nlevels* ορίζει τον αριθμό των επιπέδων στην πυραμίδα εικόνων, με αποτέλεσμα ο αριθμός των False Positives στην αρχή να ακολουθεί την καθοδική πορεία των True Positives, αλλά από

την τιμή 3.75 της παραμέτρου *scale* κι έπειτα να ακολουθεί ανοδική πορεία η οποία λίγο μετά την τιμή 4 αρχίζει να σταθεροποιείται.

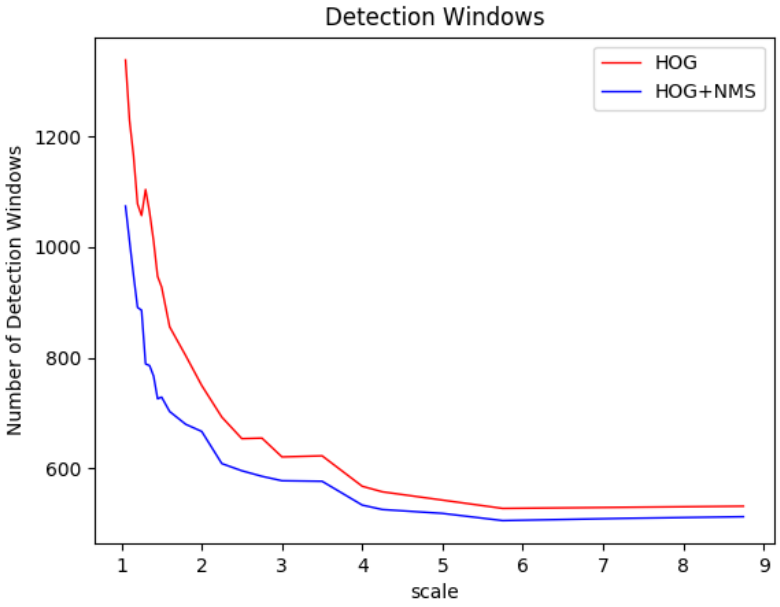
Στην συγκεκριμένη περίπτωση, η καλύτερη επίδοση του αλγορίθμου επιτυγχάνεται όταν η τιμή της scale είναι η μικρότερη δυνατή. Όπως φαίνεται κι από το Διάγραμμα 4.10, για εκείνη την τιμή το πλήθος των False Negatives και True Positives παίρνουν την ελάχιστη και μέγιστη τιμή τους αντίστοιχα, που σημαίνει ότι τότε γίνονται οι περισσότερες σωστές ανιχνεύσεις.




***Διάγραμμα 4.10:*** *Ο αριθμός των True Positives, False Positives και False Negatives ως προς τις τιμές της παραμέτρου scale με χρήση του επιπρόσθετου αλγορίθμου Non-Maximum Suppression.*

Συνολικά, η επίδραση της εφαρμογής του επιπλέον αλγορίθμου Non-Maximum Suppression, είναι εμφανής στις καμπύλες του Διαγράμματος 4.11 όπου απεικονίζεται ο αριθμός των παραθύρων ανίχνευσης ως προς τις τιμές τις παραμέτρου *scale* για τις δύο υλοποιήσεις του αλγορίθμου των Dalal και Triggs. Όπως φαίνεται κι από το διάγραμμα, το πλήθος των τετραγώνων ανίχνευσης που επέστρεψε η δεύτερη έκδοση του αλγορίθμου ήταν μικρότερο σε σχέση με την πρώτη ενώ και στις δύο περιπτώσεις, τα περισσότερα παράθυρα επιστρέφονται όταν η παράμετρος *scale* παίρνει την μικρότερη τιμή της. Μπορεί να διαπιστώσει κανείς ότι μια ανάλογη επίδραση στον συνολικό αριθμό των ανιχνεύσεων είχε και η παράμετρος *minNeighbors* της πρώτης μεθόδου. Η λογική και στις

δυο περιπτώσεις είναι η ίδια καθώς όπως και στην περίπτωση των Viola και Jones, με τον αλγόριθμο του Non-Maximum Suppression θέτουμε ένα επιπλέον κριτήριο στον αλγόριθμο για τα παράθυρα ανίχνευσης, επηρεάζοντας έτσι το πλήθος τους.

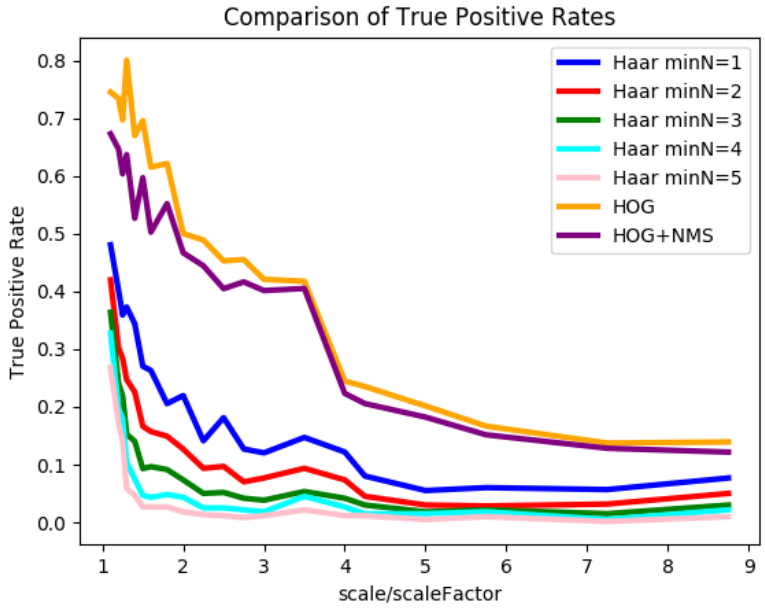



***Διάγραμμα 4.11:*** *Σύγκριση του πλήθος των παραθύρων ανίχνευσης (detection windows) που επέστρεψαν οι δυο αλγόριθμοι ως προς τις τιμές της παραμέτρου scale με χρήση του επιπρόσθετου αλγορίθμου Non-Maximum Suppression.*

## 4.3 Σύγκριση αποτελεσμάτων και παρατηρήσεις

Παρακάτω παρατίθενται συγκριτικά τα αποτελέσματα που προέκυψαν. Η σύγκριση των τριών αλγορίθμων έγινε με βάση το ποσοστό των σωστών ανιχνεύσεων επί του συνόλου των θετικών δειγμάτων και το ποσοστό των λάθος ανιχνεύσεων επί του συνόλου των αρνητικών δειγμάτων. Με άλλα λόγια, πόσοι από τους ανθρώπους που εμφανίζονταν στο σύνολο των εικόνων ανιχνεύτηκαν και πόσες ανιχνεύσεις υπήρξαν σε εικόνες που δεν περιείχαν κάποιον άνθρωπο. Αυτές οι σχέσεις εκφράζονται μέσω του Ρυθμού Αληθώς Θετικών ανιχνεύσεων ή True Positive Rate και του Ρυθμού Ψευδώς Θετικών ανιχνεύσεων ή False Positive Rate. Τέλος παρουσιάζεται ο χρόνος ανά ανίχνευση για κάθε έναν αλγόριθμο. Αυτός προέκυψε διαιρώντας τον χρόνο που χρειάστηκε κάθε αλγόριθμος να ελέγξει το σύνολο των εικόνων, με τον αριθμό των ανιχνεύσεων που επέστρεψε.

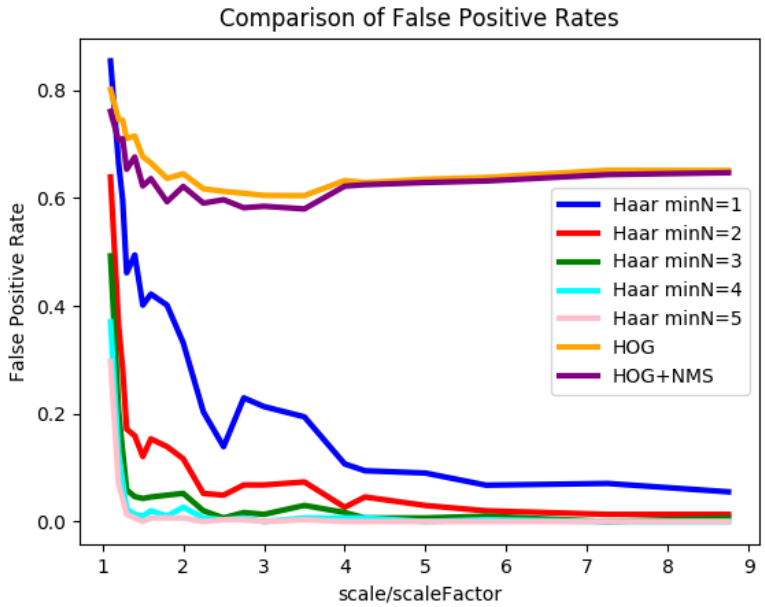
Στο Διάγραμμα 4.12 παρουσιάζονται οι ρυθμοί των Αληθώς Θετικών ανιχνεύσεων ως προς την παράμετρο μεταβολής της κλίμακας. Οι δύο εκδόσεις της μεθόδου των Dalal και Triggs σημειώνει, με σημαντική διαφορά, υψηλότερα ποσοστά σωστών ανιχνεύσεων έναντι της μεθόδου των Viola και Jones. Είναι εμφανές ότι ο αλγόριθμος HOG κατάφερε να ανιχνεύσει το μεγαλύτερο ποσοστό των ανθρώπων, με την δεύτερη υλοποίηση του αλγορίθμου να ακολουθεί με επίσης πολύ ψηλές επιδόσεις. Από την άλλη, ο αλγόριθμος Haar, με την παράμετρο *minNeighbors* ίση με 5, είχε την χειρότερη επίδοση. Μια διαφορά των ανιχνευτών, είναι η τιμή του παράγοντα μεταβολής της κλίμακας στην οποία παρουσιάζουν ολικά μέγιστα. Ο ανιχνευτής Haar όπως και η παραλλαγή του ανιχνευτή HOG, με τον επιπρόσθετο αλγόριθμο NMS, πετυχαίνουν την καλύτερη τους επίδοση για την μικρότερη δυνατή τιμή του παράγοντα *scaleFactor* και *scale* αντίστοιχα. Σε αντίθεση, ο ανιχνευτής HOG, πετυχαίνει τις περισσότερες σωστές ανιχνεύσεις για μία μεγαλύτερη τιμή του παράγοντα αυτού που είναι το 1.3.




***Διάγραμμα 4.12:*** *Οι ρυθμοί των True Positives ως προς τον παράγοντα μεταβολής της κλίμακας. Από αυτές τις καμπύλες, πέντε αντιστοιχούν στον αλγόριθμου των Viola και Jones, μία για κάθε τιμή της παραμέτρου minNeighbors και δύο στις δύο παραλλαγές του αλγορίθμου των Dalal και Triggs.*

Οι ρυθμοί των Ψευδώς Θετικών ανιχνεύσεων ως προς την παραμέτρους *scale* και *scaleFactor* παρουσιάζονται στο Διάγραμμα 4.13. Τα υψηλότερα ποσοστά λάθος

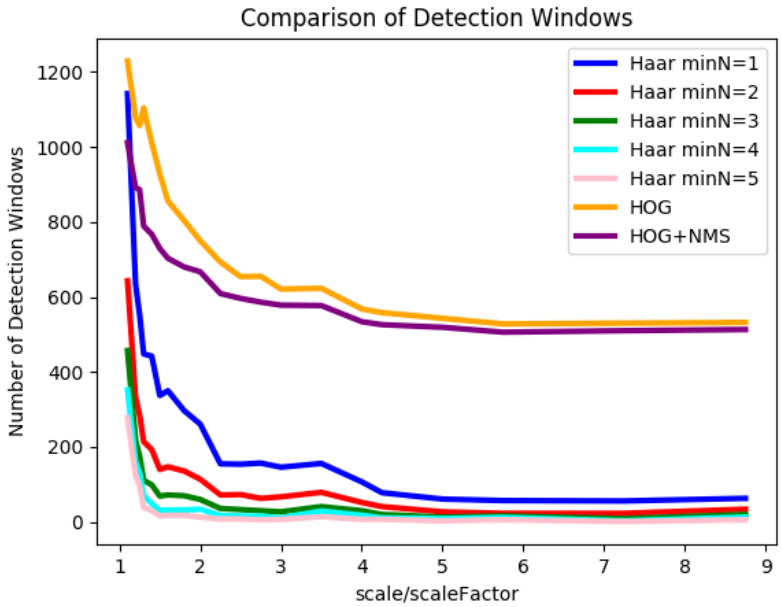
ανιχνεύσεων ανήκουν στον αλγόριθμο Haar για τιμή της παραμέτρου *minNeighbors* ίση με 1. Την καλύτερη επίδοση με τα χαμηλότερα ποσοστά έχει πάλι ο αλγόριθμος Haar όταν η παράμετρος *minNeighbors* ισούται με 5. Ωστόσο, υπάρχει μια σημαντική διαφορά. Ενώ στην αρχή, όλες οι υλοποιήσεις έχουν υψηλά ποσοστά λάθος ανιχνεύσεων, τα ποσοστά αυτά για τον αλγόριθμο Haar πολύ γρήγορα παρουσιάζουν έντονη πτώση ενώ για τους αλγορίθμους HOG παραμένουν σταθερά υψηλά. Αυτό είναι απόλυτα λογικό καθώς ο συγκεκριμένος αλγόριθμος είχε το μεγαλύτερο πλήθος ανιχνεύσεων, οπότε ήταν επόμενο να έχει και τα μεγαλύτερα ποσοστά, τόσο σωστών όσο και λανθασμένων ανιχνεύσεων. Αυτό αποδεικνύεται στο Διάγραμμα 4.14 όπου ο κάθετος άξονας αντιπροσωπεύει τον αριθμό των παραθύρων ανίχνευσης (detection windows) και ο οριζόντιος άξονας τις τιμές που έπαιρναν οι παράμετροι *scale* και *scaleFactor* αντίστοιχα. Αν παρατηρήσει κανείς, στις δύο υλοποιήσεις του αλγορίθμου των Dalal και Triggs, επιστράφηκε μεγαλύτερος αριθμός παραθύρων ανίχνευσης ο οποίος δεν παρουσίασε την ανάλογη πτώση των παραθύρων που επέστρεψε ο αλγόριθμος των Viola και Jones και παρέμεινε σταθερός από ένα σημείο κι έπειτα. Αυτό δικαιολογεί τις υψηλές τιμές του False Positive Rate.




***Διάγραμμα 4.13:*** *Οι ρυθμοί των False Positives ως προς τον παράγοντα μεταβολής της κλίμακας.*

Τέλος η σταθεροποίηση των τιμών του κάθετου άξονα από τιμή 4 της παραμέτρου *scale* ή *scaleFactor* κι έπειτα είναι ένα κοινό φαινόμενο σε όλες τις περιπτώσεις. Αυτό δικαιολογεί το εύρος των τιμών των παραμέτρων στις οποίες δοκιμάστηκαν οι αλγόριθμοι.

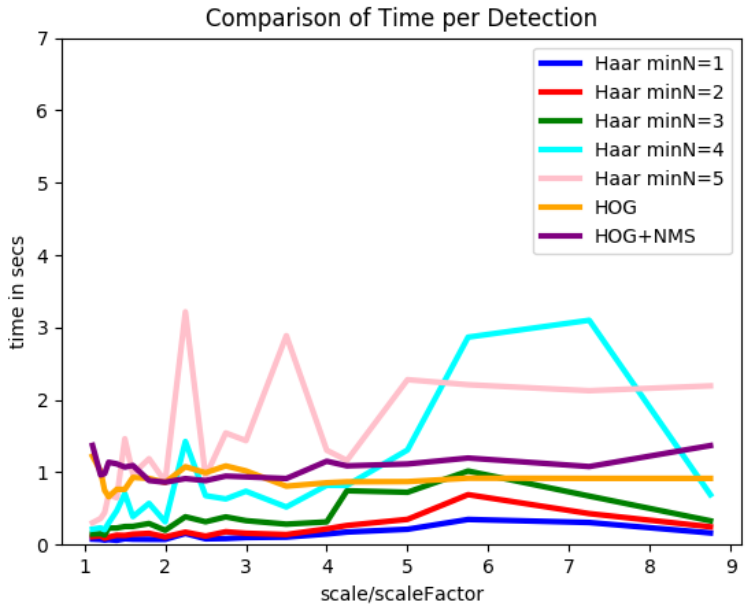
Για την σύγκριση της ταχύτητας των αλγορίθμων χρησιμοποιήθηκε η τιμή του χρόνου ανά ανίχνευση. Τα αποτελέσματα παρουσιάζονται στο Διάγραμμα 4.15. Στον κάθετο άξονα βρίσκονται οι τιμές του χρόνου σε δευτερόλεπτα ενώ στον οριζόντιο οι τιμές των παραμέτρων *scale* και *scaleFactor*. Τα αποτελέσματα αναδεικνύουν πιο γρήγορο τον




***Διάγραμμα 4.14:*** *Το πλήθος των παραθύρων ανίχνευσης (detection windows) ως προς την παράμετρο μεταβολής της κλίμακας.*

αλγόριθμο Haar με τιμή minNeighbors ίση με 1. Ενδιαφέρον έχουν δύο περιπτώσεις του ίδιου αλγορίθμου, για τις τιμές 4 και 5 της παραμέτρου *minNeighbors,* όπου παρατηρούνται έντονες διακυμάνσεις στις τιμές του χρόνου. Για τις περισσότερες τιμές του παράγοντα *scaleFactor*, ο αλγόριθμος Haar με τιμή *minNeighbors* ίση με 5 ήταν αυτός που χρειάστηκε τον περισσότερο χρόνο. Στον συγκεκριμένο αλγόριθμο παρατηρήθηκε επίσης και η μέγιστη τιμή χρόνου η οποία ήταν 3,38 δευτερόλεπτα. Παρόλα αυτά, για ένα ορισμένο εύρος τιμών της παραμέτρου *scaleFactor*, από 5.75 έως 7,25, ο αλγόριθμος Haar με τιμή minNeighbors ίση με 4, ήταν αυτός που σημείωσε τις υψηλότερες τιμές στον χρόνο ανίχνευσης. Ακολουθούν οι δύο εκδόσεις του αλγορίθμου HOG οι οποίες δεν παρουσιάζουν μεγάλη απόκλιση μεταξύ τους, με την έκδοση του αλγορίθμου που δεν περιλαμβάνει την εφαρμογή του Non-Maximum Suppression, να προηγείται στην ταχύτητα στις περισσότερες τιμές της παραμέτρου *scale*. Οι άλλες δύο περιπτώσεις του αλγορίθμου Haar, όταν η παράμετρος minNeighbors ήταν ίση με 2 και 3, έπονται των αλγορίθμων HOG όσον αφορά τον χρόνο ανίχνευσης. Συνοψίζοντας, στις περιπτώσεις που

χρησιμοποιήθηκε η μέθοδος των Viola και Jones, υπήρχε μια ποικιλία στην διάρκεια που χρειάστηκε η κάθε μέθοδος να εντοπίσει έναν άνθρωπο, ενώ γενικά σημειώθηκαν καλύτερες επιδόσεις στον χρόνο σε σχέση με τις δύο εκδοχές της μεθόδου των Dalal και Triggs.

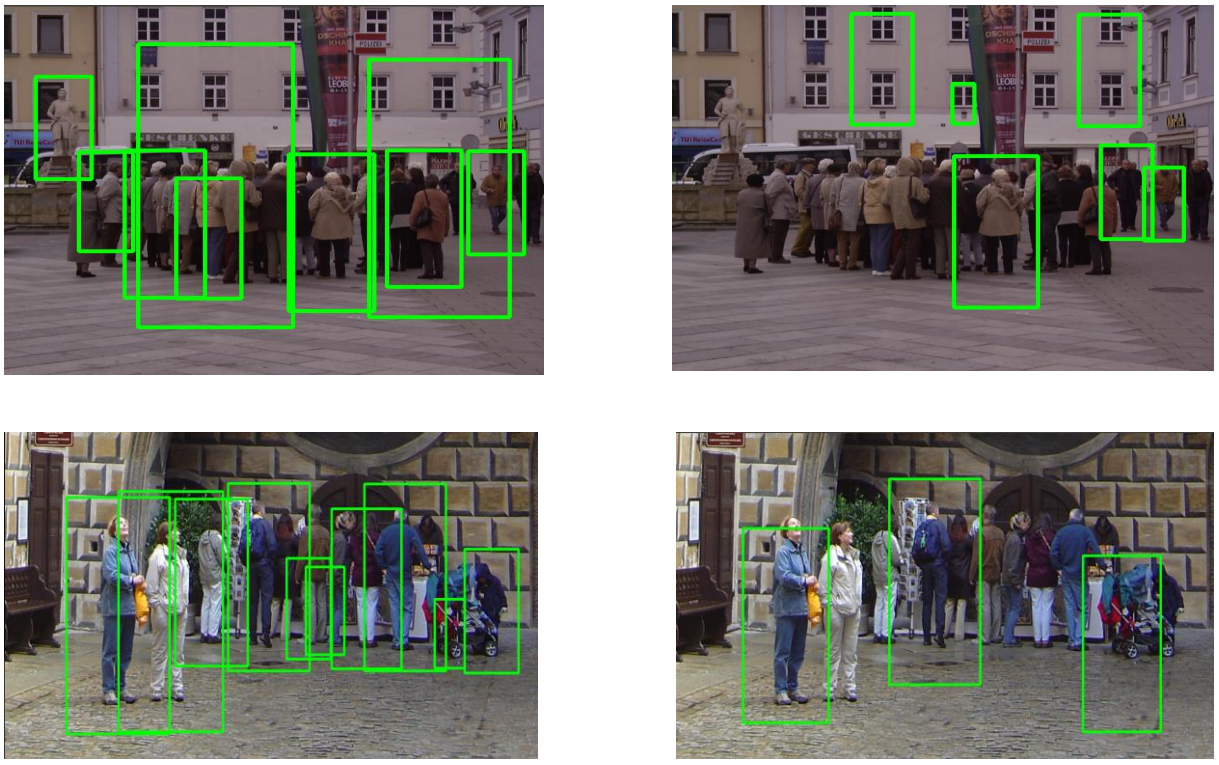



***Διάγραμμα 4.15:*** *Οι χρόνοι ανά ανίχνευση σε δευτερόλεπτα ως προς τον παράγοντα αλλαγής της κλίμακας.*

Ολοκληρώνοντας, παρατίθενται κάποιες παρατηρήσεις που έγιναν κατά την διάρκεια της διεξαγωγής των πειραμάτων και σχετίζονται με το πως απέδωσαν οι αλγόριθμοι σε ορισμένες συνθήκες. Σε κάποιες από τις εικόνες του συνόλου δεδομένων, οι άνθρωποι τύχαινε να βρίσκονται πίσω από κάποιο εμπόδιο ή να μην είναι σε όρθια στάση. Αυτές οι περιπτώσεις εικόνων δυσκόλευαν το έργο της ανίχνευσης καθώς κανένας αλγόριθμος δεν κατάφερε να εντοπίσει ανθρώπους που απεικονίζονταν υπό αυτές τις συνθήκες. Παραδείγματα τέτοιων εικόνων παρουσιάζονται στην Εικόνα 4.1. Στις περιπτώσεις που υπήρχαν πολλαπλές ανθρώπινες φιγούρες στην ίδια εικόνα, καλύτερη απόδοση είχε ο αλγόριθμος HOG καθώς, όπως φαίνεται και στην Εικόνα 4.2, μπορούσε πιο εύκολα να τις εντοπίσει σε αντίθεση με τον αλγόριθμο Haar.

Κάτι άλλο που παρατηρήθηκε συχνά ήταν οι μερικές ανιχνεύσεις ή ανιχνεύσεις μελών του ανθρώπινου σώματος. Σε αυτές τις περιπτώσεις ο αλγόριθμος δεν αναγνώριζε ολόκληρη την ανθρώπινη φιγούρα αλλά ένα ή περισσότερα μέρη αυτής όπως είναι τα ανθρώπινα άκρα. Αυτές οι ανιχνεύσεις δεν θεωρήθηκαν σωστές γιατί παρόλο που


***Εικόνα 4.1:*** *Αριστερά ένας άνθρωπος σε μία περίεργη στάση πάνω στο ποδήλατο. Δεξιά τρείς άνθρωποι βρίσκονται πίσω από ένα αμάξι. Αυτοί οι άνθρωποι δεν ανιχνεύτηκαν από κανένα αλγόριθμο [33].*



***Εικόνα 4.2:*** *Αριστερά τα αποτελέσματα της εφαρμογής του αλγορίθμου HOG και δεξιά τα αποτελέσματα εφαρμογής του αλγορίθμου Haar στην ίδια εικόνα με πολλές εμφανίσεις του αντικειμένου ενδιαφέροντος.*

απαρτίζουν το ανθρώπινο σώμα, το ζητούμενο ήταν να ανιχνευτεί ολόκληρος ο άνθρωπος. Παραδείγματα τέτοιων ανιχνεύσεων παρουσιάζονται στην Εικόνα 4.3.

(a) (b) (c)

***Εικόνα 4.3:*** *(a) Μια σωστή ανίχνευση. (b) Μια μερική ανίχνευση. (c) Ανίχνευση ενός μέλους του ανθρώπινου σώματος.*

# ΚΕΦΑΛΑΙΟ 5: Συμπεράσματα και μελλοντική εργασία

Στην παρούσα διπλωματική εργασία εξετάστηκαν δύο από τις πιο δημοφιλείς μεθόδους στον τομέα της ανίχνευσης αντικειμένων. Το αντικείμενο, με την ανίχνευσή του οποίου ασχοληθήκαμε στην παρούσα εργασία, ήταν ο άνθρωπος ή η ανθρώπινη φιγούρα. Κατασκευάστηκαν τρεις αλγόριθμοι, οι δύο εκ των οποίον αφορούσαν την ίδια μέθοδο. Διεξήχθη μια σειρά από πειράματα στα οποία οι αλγόριθμοι αυτοί εξετάστηκαν σε ένα σύνολο δεδομένων για διάφορες τιμές των παραμέτρων τους και μελετήθηκε η συμπεριφορά τους.

Την καλύτερη επίδοση αναφορικά με τα ποσοστά σωστών ανιχνεύσεων είχε η μέθοδος των Dalal και Triggs ενώ με βάση την ταχύτητα, καλύτερη επίδοση είχε μια έκδοση του αλγορίθμου των Viola και Jones με αυτή των Dalal και Triggs να ακολουθεί με μικρή διαφορά. Επιπλέον, παρατηρήθηκε ότι για μικρές τιμές των παραμέτρων οι αλγόριθμοι αποδίδουν καλύτερα. Παράλληλα υπήρχε μια αντίστροφη σχέση ποιότητας και ταχύτητας κάτι που συμβαίνει συχνά σε συστήματα ταξινόμησης. Η επιλογή των κατάλληλων τιμών των παραμέτρων καθορίζεται από την φύση της εφαρμογής, με την ταχύτητα να έχει καθοριστικό ρόλο σε συστήματα πραγματικού χρόνου.

Σαν μελλοντική εργασία θα ήταν ενδιαφέρον να κατασκευαστεί ένα υβριδικό μοντέλο των δύο αυτών μεθόδων έτσι ώστε να επιτευχθεί η μέγιστη δυνατή επίδοση τόσο στην ανίχνευση ανθρώπων όσο και στην ανίχνευση άλλων αντικειμένων.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] *“Τεχνητή νοημοσύνη,” Wikipedia, 13-Jul-2020. [Online]. Available: https://el.wikipedia.org/wiki/Τεχνητή_νοημοσύνη. [Accessed: 06-Oct-2020].*

[2] W. Förstner and E. Gülch, “A fast operator for detection and precise location of distinct points, corners and centres of circular features,” *Intercommission Conference on Fast Processing of Photogrammetric Data*, no. Interlaken, Switzerland, pp. 281–305, Jun. 1987.

[3] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces,” *Journal of the Optical Society of America A*, vol. 4, no. 3, p. 519, 1987.

[4] D. Forsyth and M. Fleck, “Body plans,” *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[5] D. Gavrila and V. Philomin, “Real-time object detection for ‘smart’ vehicles,” *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999.

[6] J. Batista, P. Peixoto, and H. Araujo, “Real-time active visual surveillance by integrating peripheral motion detection with foveated tracking,” *Proceedings 1998 IEEE Workshop on Visual Surveillance*.

[7] P. Dollar, C. Wojek, B. Schiele, and P. Perona, “Pedestrian Detection: An Evaluation of the State of the Art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[8] C. G. Keller, T. Dang, H. Fritz, A. Joos, C. Rabe, and D. M. Gavrila, “Active Pedestrian Safety by Automatic Braking and Evasive Steering,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1292–1304, 2011.

[9] Σ. Πουλαράκης, “Video analysis for event and activity detection and recognition,” thesis, University of Thessaly, Volos, 2010.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, pp. 886-893 vol. 1.

[11] P. Viola and M. Jones, “Robust Real-time Object Detection,” S*econd International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing, And Sampling*, 2001.

[12] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, “Deep Convolutional Neural Networks for pedestrian detection,” *Signal Processing: Image Communication*, vol. 47, pp. 482–489, 2016.

[13] S. Mallick, “Image Recognition and Object Detection: Part 1”, [Online]. Available: https://www.learnopencv.com/image-recognition-and-object-detection-part1/. Accessed: [10-Dec-2019]

[14] N. Dalal. Finding People in Images and Videos. PhD thesis, Institut National Polytechnique de Grenoble, 2006.

[15] *Stanford Artificial Intelligence Laboratory*. [Online]. Available: https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html. [Accessed: 12-Oct-2020]. [16] J. Courtney, “Application of Digital Image Processing to Marker-free Analysis of Human Gait,” thesis, The National University of Ireland , Dublin, 2001.

[17] C. McCormick, “HOG Person Detector Tutorial”, May 9, 2013. [Online]. Available: https://chrisjmccormick.wordpress.com/2013/05/09/hog-person-detector-tutorial/. Accessed: [10-Nov-2019].

[18] N. V. Lĩnh, “Deeplearning.ai: CNN week 1 - Convolutional Neural Network terminology,” *Medium*, 07-Feb-2018. [Online]. Available: https://medium.com/datatype/deeplearning-ai-cnn-week-1-a0bac10d509b. [Accessed: 15-Oct-2020].

[19] C. Mutia, F. Arnia, and R. Muharar, “Improving the Performance of CBIR on Islamic Women Apparels Using Normalized PHOG,” *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 3, pp. 271–280, Sep. 2017.

[20] S. Mallick, “Histogram of Oriented Gradients”, Dec. 6, 2016. [Online]. Available: https://www.learnopencv.com/histogram-of-oriented-gradients/. Accessed: [14-Nov-2019].

[21] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *International Conference on Computer Vision*, 1998.

[22] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004.

[23] D. Bradley and G. Roth, “Adaptive Thresholding using the Integral Image,” *Journal of Graphics Tools*, vol. 12, no. 2, pp. 13–21, 2007.

[24] “Statistical classification,” *Wikipedia*, 15-Oct-2019. [Online]. Available: https://en.wikipedia.org/wiki/Statistical_classification. [Accessed: 12-Dec-2019].

[25] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In Computational Learning Theory: Eurocolt ’95, pages 23–37. Springer-Verlag, 1995.

[26] R. E. Schapire and Y. Freund, *Boosting: foundations and algorithms*. Cambridge, MA: MIT Press, 2014.

[27] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[28] R. Sunil, “Understanding Support Vector Machine algorithm from examples (along with code),” 13-Sep-2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/. [Accessed: 16-Dec-2019].

[29] “Introduction to Support Vector Machines,” *Open Source Computer Vision*. [Online]. Available: https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html. [Accessed: 16-Dec-2019].

[30] “Support-vector machine,” *Wikipedia, the free encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine. [Accessed: 16-Dec-2019].

[31] C. J. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” *Data Mining and Knowledge Discovery 2*, pp. 121–167, 1998.

[32] C. P. Papageorgiou, *A trainable system for object detection in images and video sequences*. 2000.

[33] INRIA Person Dataset for Pedestrian Detection. http://pascal. inrialpes.fr/data/human/.

[34] A. Suleiman and V. Sze, “An Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD 60 fps with Multi-Scale Support,” *Journal of Signal Processing Systems*, vol. 84, no. 3, pp. 325–337, 2015.

[35] “About,” *OpenCV*. [Online]. Available: https://opencv.org/about/. [Accessed: 11-Jun-2020].

[36] “OpenCV,” *Wikipedia, the free encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV. [Accessed: 07-Jun-2020].

[37] S. A. Dave, M. S. Nagmode, and A. Jahagirdar, “Statistical Survey on Object Detection and tracking Methodologies,” *International Journal of Scientific & Engineering Research*, vol. 4, no. 3, Mar. 2013.

[38] “Python (programming language), ”*Wikipedia, the free encyclopedia*. [Online]. Available:https://en.wikipedia.org/wiki/Python_(programming_language). [Accessed: 08-Jun-2020].

[39] “PYPL PopularitY of Programming Language index,” *index*. [Online]. Available: http://pypl.github.io/PYPL.html. [Accessed: 11-Jun-2020].

[40] “.XML File Extension,” *XML File Extension - What is an .xml file and how do I open it?*, 27-Feb-2019. [Online]. Available: https://fileinfo.com/extension/xml. [Accessed: 17-Jun-2020].

[41] “cv::CascadeClassifier Class Reference,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4.1/d1/de5/classcv_1_1CascadeClassifier.html. [Accessed: 01-Nov-2020].

[42] “Object Detection,” *Object Detection - OpenCV 2.4.13.7 documentation*. [Online]. Available: https://docs.opencv.org/2.4/modules/gpu/doc/object_detection.html. [Accessed: 06-Nov-2020].

[43] “Smoothing Images,” *OpenCV*. [Online]. Available: https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html. [Accessed: 01-Sep-2020].

[44] “Gamma correction,” *Wikipedia*, 10-Sep-2020. [Online]. Available: https://en.wikipedia.org/wiki/Gamma_correction. [Accessed: 01-Sep-2020].

[45] R. Person, Real PersonReal Person 5955 bronze badges, and Abhishek MishraAbhishek Mishra 1, “Changing Non Maximum Suppression to pick smallest box for OpenCV in Python,” Stack Overflow, 01-Jul-1967. [Online]. Available: https://stackoverflow.com/questions/51599456/changing-non-maximum-suppression-to-pick-smallest-box-for-opencv-in-python?cv=1. [Accessed: 25-Jan-2021].

[46] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[47] J. Hosang, R. Benenson, and B. Schiele, “Learning Non-maximum Suppression,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] A. Rosebrock, “HOG detectMultiScale parameters explained,” *PyImageSearch*, 18-Apr-2020. [Online]. Available: https://www.pyimagesearch.com/2015/11/16/hog-detectmultiscale-parameters-explained/. [Accessed: 04-Sep-2020].

[49] D. T. Nguyen, *Human detection from images and videos*. University of Wollongong, 2012.

[50] Χ. Ι. Βαρυτιμίδης, "Ανίχνευση αντικειμένων και ημιαυτόματος χαρακτηρισμός εικόνων," thesis, Εθνικό Μετσόβειο Πολυτεχνείο, Αθήνα, 2008.

# ΠΑΡΑΡΤΗΜΑ Α: Ο κώδικας της εργασίας

Παρακάτω βρίσκεται ο κώδικας των τριών μεθόδων που υλοποιήθηκαν. Η κάθε μέθοδος υλοποιήθηκε σε ξεχωριστό αρχείο κώδικα.

## Α.1 Η υλοποίηση του αλγορίθμου των Viola και Jones

```
import cv2
from glob import glob

body_csc = cv2.CascadeClassifier('haarcascade_fullbody.xml')
numberOfpics = 0
numberOfrecs = 0

for imageF in glob('*.png'):
    numberOfpics+=1
    image = cv2.imread(imageF)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    body = body_csc.detectMultiScale(gray, 1.2, 1)
    for (x,y,w,h) in body:
        numberOfrecs+=1
        cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)


    cv2.imshow(imageF,image)
    k = cv2.waitKey(0)
    if k == 32:
        cv2.destroyAllWindows()

print("Total no. of images:", numberOfpics)
print("Total no. of rectangles:", numberOfrecs)
```

## Α.2 Η υλοποίηση του αλγορίθμου των Dalal και Triggs

```
import cv2
from glob import glob


descriptor = cv2.HOGDescriptor()
descriptor.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

thereIs= False
numberOfpics = 0
numberOfrecs = 0
n=0
for imageF in glob('*.png'):
    numberOfpics+=1
    n+=1
    if n>100:
        d=598-numberOfpics
        print("left ---->",d)
        n=0
    image = cv2.imread(imageF)
```

```
    (body, _) = descriptor.detectMultiScale(image, 0, (1,1), (4,4),
                1000, 2, False);

    for (x,y,w,h) in body:
        numberOfrecs+=1
        thereIs = True
        cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),3)

    if thereIs:
        thereIs = False
        cv2.imshow(imageF,image)
        k = cv2.waitKey(0)
        if k == 32:
                cv2.destroyAllWindows()

print("Total no. of images:", numberOfpics)
print("Total no. of rectangles:", numberOfrecs)
```

## A.3 Η υλοποίηση του αλγορίθμου των Dalal και Triggs με τον επιπρόσθετο αλγόριθμο NMS

```
import cv2
from glob import glob
import numpy as np
import matplotlib.pyplot as mt


descriptor = cv2.HOGDescriptor()
descriptor.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

#NMS FUNCTION
def nms(detections, th):

    if len(detections)==0:
        return []

    choose = []

    a1 = detections[:,0]
    b1 = detections[:,1]
    a2 = detections[:,2]
    b2 = detections[:,3]

    arrea = (a2 - a1) * (b2 - b1)
    indexes = np.argsort(b2)

    while len(indexes) > 0:
        lastEl = len(indexes) -1
        i = indexes[lastEl]
        choose.append(i)
        suppress = [lastEl]

        for pos in xrange(0, lastEl):

            j= indexes[pos]
```

```
            aa1 = max(a1[i],a1[j])
            bb1 = max(b1[i],b1[j])
            aa2 = min(a2[i],a2[j])
            bb2 = min(b2[i],b2[j])

            width = max(0, aa2 - aa1 + 1)
            height = max(0, bb2 - bb1 + 1)

            overlap = float(width*height)/ arrea[j]

            if overlap > th:
                suppress.append(pos)

        indexes = np.delete(indexes, suppress)

    return detections[choose]



thereIs= False
numberOfpics = 0
numberOfrecs = 0
n=0
for imageF in glob('*.png'):
    numberOfpics+=1
    n+=1
    if n>100:
        d=598-numberOfpics
        print("left ---->",d)
        n=0
    image = cv2.imread(imageF)
    (body, _) = descriptor.detectMultiScale(image, 0, (1,1), (4,4),
2.75, 2, False);

    for i in range(0,len(body)):
        z=body[i]
        z[2]=z[2]+z[0]
        z[3]=z[3]+z[1]

    body2=nms(body, 0.5)

    for (x,y,w,h) in body2:

        numberOfrecs+=1
        thereIs = True
        cv2.rectangle(image,(x,y),(w,h),(100,255,0),3)

    if thereIs:
        thereIs = False
        cv2.imshow(imageF,image)
        k = cv2.waitKey(0)
        if k == 32:
                cv2.destroyAllWindows()

print("Total no. of images:", numberOfpics)
print("Total no. of rectangles:", numberOfrecs)
```