ΕΘΝΙΚΟΝ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟΝ ΠΑΝΕΠΙΣΤΗΜΙΟΝ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΤΟΜΕΑΣ Μ.Δ.Ε. ΣΤΟΝ ΗΛΕΚΤΡΟΝΙΚΟ ΑΥΤΟΜΑΤΙΣΜΟ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ: ΜΕΘΟΔΟΙ ΑΝΙΧΝΕΥΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΣΩΠΟΥ

Ονοματεπώνυμο: Σκουρλής Ιωάννης

Α.Μ.: 2012529

Επιβλέπων: Μανώλης Χ. Τσίλης, Επίκουρος Καθηγητής

ΑΘΗΝΑ

2016

Στο επερχόμενο μέλος

Περιεχόμενα

Κεφάλαιο 3

ΤΕΧΝΙΚΕΣ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΣΩΠΩΝ

Κεφάλαιο 4

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΣΩΠΟΥ

Κεφάλαιο 5

Πρόλογος

Στην παρούσα διπλωματική εργασία παρουσιάζονται μέθοδοι για την ανίχνευση και αναγνώριση ανθρώπινων προσώπων. Αφού έγινε εκτενής μελέτη των τεχνικών που έχουν προταθεί για τον εντοπισμό και την αναγνώριση προσώπου επιλέχτηκαν διάφοροι αλγόριθμοι ώστε να υλοποιήσουμε ένα σύστημα αναγνώρισης προσώπων. Το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε ήταν αυτό της matlab. Εκεί έγινε η εφαρμογή των αλγορίθμων στην πράξη, μελετήθηκαν οι παράμετροι που επηρεάζουν την απόδοσή τους και προέκυψαν συμπεράσματα.

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Οι εφαρμογές που απαιτούν την εξακρίβωση της ταυτότητας ενός ατόμου πολλαπλασιάζονται συνεχώς και μαζί με αυτές αυξάνεται η ανάγκη για τη χρήση μεθόδων πιστοποίησης που δεν θα απαιτούν από τους ανθρώπους να θυμούνται πολύπλοκους κωδικούς πρόσβασης ή να μεταφέρουν μαγνητικές κάρτες, ταυτότητες ή άλλου είδους έγγραφα που θα επαληθεύουν την ταυτότητα τους. Η χρήση βιομετρικών μεθόδων πιστοποίησης, δηλαδή μεθόδων που στηρίζονται στην αναγνώριση φυσικών χαρακτηριστικών ή χαρακτηριστικών της συμπεριφοράς του ανθρώπου είναι μια λύση σε αυτό το πρόβλημα. Μεταξύ των διαφόρων βιομετρικών μεθόδων που χρησιμοποιούνται, όπως η αναγνώριση ίριδας, των δακτυλικών αποτυπωμάτων, της γεωμετρίας της παλάμης, της στάσης του σώματος και του βαδίσματος, της φωνής, ακόμη και του τρόπου πληκτρολόγησης ή υπογραφής, η αναγνώριση προσώπου είναι σίγουρα μία από τις φιλικότερες προς το χρήστη, γιατί ουσιαστικά μιμείται τον τρόπο που χρησιμοποιούν οι άνθρωποι για να αναγνωρίζουν ο ένας τον άλλο.

Το πρόβλημα της αναγνώρισης προσώπου είναι βασικά ένα πρόβλημα ταξινόμησης όπου κάθε πρόσωπο στη βάση δεδομένων αναπαριστά μια κατηγορία στην οποία τα εισερχόμενα δεδομένα πρόκειται να αντιστοιχιστούν. Το πρόβλημα της ταξινόμησης επιδεινώνεται με το μέγεθος των δεδομένων. Κατά τα τελευταία δέκα χρόνια περίπου, η αναγνώριση προσώπου έχει γίνει ένα δημοφιλές θέμα έρευνας και μελέτης για τους επιστήμονες και επίσης μια από τις πιο πετυχημένες εφαρμογές ανάλυσης εικόνας. Λόγω της φύσης του προβλήματος, δεν είναι μόνο ερευνητές της επιστήμης των υπολογιστών που ενδιαφέρονται για το θέμα αυτό, αλλά νευροεπιστήμονες και ψυχολόγοι επίσης.
Κίνητρο για τη διεξαγωγή αυτής της πτυχιακής εργασίας υπήρξε ο συνεχώς αυξανόμενος ρόλος που διαδραματίζει η τεχνολογία σε όλους τους τομείς της κοινωνίας μας. Ενώ μέχρι πρόσφατα η αναγνώριση προσώπων ήταν δύσκολη υπόθεση, στις μέρες μας ολοένα και περισσότερες τεχνικές αναπτύσσονται με αξιόλογα αποτελέσματα.

Για τον άνθρωπο η αναγνώριση προσώπων είναι μία καθημερινή, φυσική και εύκολη σχετικά διαδικασία. Αντίθετα, η αναγνώριση προσώπων με τη βοήθεια ενός ευφυούς συστήματος έχει αποδειχθεί μία δύσκολη προσπάθεια, ειδικά σε μη ελεγχόμενα περιβάλλοντα, όπου η θέση παρατήρησης, ο φωτισμός, οι μορφασμοί και τα αντικείμενα ή αξεσουάρ που αποκρύπτουν το πρόσωπο, ποικίλουν

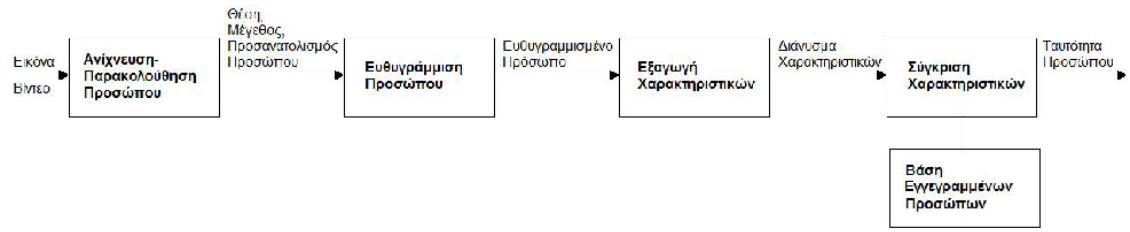
χαρακτηριστικά. Παρόλα αυτά, η απόδοση των συστημάτων αναγνώρισης προσώπου έχει βελτιωθεί σημαντικά από την εποχή που παρουσιάστηκε το πρώτο αυτόματο σύστημα αναγνώρισης προσώπου από τον Kanade, ώστε η αναγνώριση να μπορεί πλέον να διεξαχθεί επιτυχώς σε πραγματικό χρόνο κάτω από συγκεκριμένες ελεγχόμενες συνθήκες.

Ένα σύστημα αναγνώρισης προσώπου (face recognition) αναμένεται να αναγνωρίζει αυτόματα πρόσωπα που βρίσκονται σε φωτογραφίες ή βίντεο. Το σύστημα μπορεί να λειτουργεί με δύο τρόπους :
1. Διακρίβωση/επαλήθευση προσώπου (face verification/authentication), και
2. Ταυτοποίηση προσώπου (face identification).
Η διακρίβωση συνίσταται στην σύγκριση ενός ζητούμενου προσώπου με το συγκεκριμένο δείγμα του προσώπου του οποίου ισχυρίζεται ότι έχει την ταυτότητα και στην επαλήθευση ή όχι της ταυτότητάς του.
Η ταυτοποίηση συνίσταται στην σύγκριση ενός ζητούμενου προσώπου με ένα σύνολο από δείγματα προσώπων μιας βάσης για να προσδιοριστεί σε ποιο ταιριάζει περισσότερο ώστε να καθοριστεί η ταυτότητά του.

1.2 Περιγραφή του προβλήματος της Αναγνώρισης Προσώπων

Η αναγνώριση προσώπου είναι ένα πρόβλημα αναγνώρισης προτύπων (pattern recognition). Ένα δοκιμαστικό πρόσωπο (face probe), ως ένα τρισδιάστατο αντικείμενο που υπόκειται σε μεταβολές ανάλογα με τον φωτισμό, τον προσανατολισμό, τους μορφασμούς κλπ., πρέπει να αναγνωρισθεί συγκρινόμενο με 2-διάστατες ή 3-διάστατεςψεικόνες μιας συλλογής προσώπων (face gallery), που είναι διαθέσιμες σε μία βάση δεδομένων.

Ένα σύστημα αναγνώρισης προσώπου αποτελείται γενικά από τέσσερα υποσυστήματα , όπως φαίνεται στην Εικόνα 1: την ανίχνευση, την ευθυγράμμιση, την εξαγωγή χαρακτηριστικών και την σύγκριση, όπου ο εντοπισμός και η κανονικοποίηση των προσώπων, μέσω της ανίχνευσης και της ευθυγράμμισης, προηγούνται της αναγνώρισης, μέσω της εξαγωγής των χαρακτηριστικών και της σύγκρισης με τη βάση των εγγεγραμμένων προσώπων.



Εικ. 1. Διάγραμμα διαδικασίας αναγνώρισης προσώπου

Η ανίχνευση προσώπου (face detection) αναφέρεται στον έλεγχο αν υπάρχουν πρόσωπα σε μία εικόνα και στον προσδιορισμό της θέσης και των διαστάσεών τους. Ο εντοπισμός προσώπου (face localization) αναφέρεται ειδικότερα στον προσδιορισμό της θέσης και των διαστάσεων του προσώπου, στην περίπτωση που σε μία εικόνα υπάρχει σίγουρα ένα μόνο πρόσωπο. Στην περίπτωση του βίντεο τα πρόσωπα που ανιχνεύονται πρέπει να ανιχνεύονται διαρκώς σε κάθε πλαίσιο, με ένα σύστημα παρακολούθησης προσώπου (face tracking). Η ευθυγράμμιση προσώπου (face alignment/registration) στοχεύει στον εντοπισμό του προσώπου με μεγαλύτερη ακρίβεια, στον προσδιορισμό του μεγέθους και του προσανατολισμού του (πόζα), ενώ η ανίχνευση προσώπου πετυχαίνει απλά ένα χοντρικό προσδιορισμό της θέσης και του μεγέθους του προσώπου. Για την ευθυγράμμιση του προσώπου επιλέγονται χαρακτηριστικά σημεία αναφοράς ή ορόσημα του προσώπου (fiducial points/landmarks), από συγκεκριμένα σημεία ενδιαφέροντος (interest points) που εντοπίζονται στην εικόνα του προσώπου, όπως τα μάτια, τα φρύδια, η μύτη και το στόμα. Τα πρόσωπα κανονικοποιούνται σε σχέση με τις γεωμετρικές τους ιδιότητες, όπως το μέγεθος και ο προσανατολισμός, χρησιμοποιώντας γεωμετρικούς μετασχηματισμούς. Τέλος, μια περαιτέρω κανονικοποίηση μπορεί να εφαρμοστεί, σε σχέση με τον φωτισμό και τις χρωματικές συνιστώσες της εικόνας του προσώπου, ώστε η εικόνα να περιέχει συγκεκριμένα επίπεδα διαβαθμίσεων του γκρι.

Η εξαγωγή χαρακτηριστικών (feature extraction) έρχεται μετά τη γεωμετρική και φωτομετρική κανονικοποίηση για να προσφέρει την απαραίτητη πληροφορία, ώστε να μπορεί να γίνει διάκριση μεταξύ των προσώπων διαφορετικών ατόμων, ανεξάρτητα από τις γεωμετρικές και φωτομετρικές διαφοροποιήσεις που υπάρχουν στις εικόνες.

Κατά την σύγκριση των προσώπων (face matching), το εξαχθέν διάνυσμα χαρακτηριστικών του άγνωστου δοκιμαστικού προσώπου συγκρίνεται με τα εγγεγραμμένα πρόσωπα της βάσης και όταν βρεθεί κατάλληλο ταίρι με επαρκή αξιοπιστία, συνάγεται η ταυτότητα του άγνωστου προσώπου, διαφορετικά η ταυτότητά του παραμένει άγνωστη.

Τα αποτελέσματα της αναγνώρισης προσώπου εξαρτώνται ιδιαιτέρως από τα χαρακτηριστικά που αντιπροσωπεύουν το πρόσωπο και από τις μεθόδους ταξινόμησης που χρησιμοποιούνται για να τα διακρίνουν. Ωστόσο η διαδικασία εντοπισμού και κανονικοποίησης των προσώπων παίζει σημαντικό ρόλο στην εξαγωγή αποτελεσματικών χαρακτηριστικών για την αναγνώριση.

1.3 Αντικείμενο της Διπλωματικής Εργασίας

Οι στόχοι της εργασίας αυτής είναι ο σχεδιασμός, η ανάπτυξη, η εφαρμογή και η αξιολόγηση αλγοριθμικών μεθόδων που αφορούν τον εντοπισμό και την αναγνώριση προσώπου και η υλοποίηση τους με την βοήθεια των δυνατοτήτων που μας παρέχει το προγραμματιστικό περιβάλλον της matlab.

Πιο συγκεκριμένα, στο κομμάτι που αφορά στον εντοπισμό προσώπου σε 2Δ εικόνες θα ασχοληθούμε με τη μέθοδο των Viola & Jones. Όσον αφορά δε στο κομμάτι της αναγνώρισης προσώπου σε 2Δ εικόνες θα ασχοληθούμε με τη μέθοδο του Δισδιάστατου Συντελεστής Συσχέτισης (2-D Coefficient of Correlation), τη μέθοδο των ιστογραμμάτων και τη μέθοδο structural similarity index method.

1.4 Διάρθρωση της Διπλωματικής Εργασίας

Η παρούσα εργασία οργανώνεται ως εξής: Στο Κεφάλαιο 2 παρουσιάζονται οι μέθοδοι ανίχνευσης προσώπου και στο Κεφάλαιο 3 παρουσιάζονται οι μέθοδοι αναγνώρισης προσώπου σε 2Δ εικόνες. Στο Κεφάλαιο 4 γίνεται η προσαρμογή και η εφαρμογή τους σε πειραματικά δεδομένα με σκοπό την αξιολόγηση της αποτελεσματικότητάς τους. Τέλος στο Κεφάλαιο 5 συγκεντρώνονται τα συμπεράσματα της εργασίας, τίθενται οι κατευθύνσεις περαιτέρω έρευνας και αναφέρεται το σύνολο του προγράμματος που υλοποιήθηκε μέσω matlab.

Κεφάλαιο 2

ΑΝΙΧΝΕΥΣΗ ΠΡΟΣΩΠΩΝ

2.1 Γενικά

Η ανίχνευση προσώπου (face detection) είναι το πρώτο βήμα στην διαδικασία αναγνώρισης προσώπου (face recognition). Η αξιοπιστία του συστήματος ανίχνευσης παίζει κυρίαρχο ρόλο στην απόδοση και χρησιμότητα ολόκληρου του συστήματος αναγνώρισης προσώπου.

Αν δοθεί μία φωτογραφία ή ένα βίντεο, ο ιδανικός ανιχνευτής προσώπου πρέπει να είναι ικανός να ανακαλύπτει και να εντοπίζει όλα τα πρόσωπα που είναι παρόντα, ανεξάρτητα από τις συνθήκες φωτισμού, τη θέση, την κλίμακα, τον προσανατολισμό και τους μορφασμούς των προσώπων.

Η μέθοδος των Viola & Jones με τις βελτιώσεις του Lienhart που στηρίζεται στην εκμάθηση με την τεχνική Adaboost και την χρήση χαρακτηριστικών γνωρισμάτων της μορφής Haar, εμφανίζεται μέχρι σήμερα να έχει την καλύτερη απόδοση ως προς την αποτελεσματικότητα και την ταχύτητα.

2.2 Περιγραφή του προβλήματος της Ανίχνευσης Προσώπου

Η ανίχνευση προσώπου μπορεί να θεωρηθεί σαν ένα πρόβλημα ταξινόμησης (classification) δύο κλάσεων, κατά το οποίο μία περιοχή μιας εικόνας ταξινομείται ως "πρόσωπο" ή "μη-πρόσωπο" . Η ανίχνευση προσώπου έγκειται στην ανακάλυψη και εντοπισμό των ανθρώπινων προσώπων σε μια εικόνα ανεξαρτήτως:

- της θέσης τους
- του μεγέθους τους (κλίμακας)
- του προσανατολισμού τους (στροφή στο επίπεδο της εικόνας)
- της στάσης/πόζας τους (στροφή εκτός επιπέδου εικόνας)
- της έκφρασής τους (ύπαρξη μορφασμών)
- της ύπαρξης άλλων δομικών στοιχείων (μούσια, μουστάκια, γυαλιά)
- του περιεχομένου της εικόνας (ύπαρξη εμποδιών ή άλλων προσώπων)
- και του φωτισμού και των συνθηκών αποτύπωσης της εικόνας (ευαισθησία κάμερας, ανάλυση)

Όλα τα παραπάνω αποτελούν τις κύριες αιτίες της μεγάλης δυσκολίας που εμφανίζει το πρόβλημα της ανίχνευσης ενός προσώπου.
Αυτές οι μεγάλες παραλλαγές στην εμφάνιση του προσώπου, στην στάση, στην έκφραση και τον φωτισμό κάνουν πολύ πολύπλοκο το πολύπτυχο (manifold) του χώρο των προσώπων και δυσδιάκριτα τα όρια προσώπων / μη-προσώπων. Ένας μη

γραμμικός ταξινομητής απαιτείται για να είναι εφικτή η διαχείριση αυτής της πολύπλοκης κατάστασης, καθώς επίσης και ένα μεγάλο σύνολο δειγμάτων εκπαίδευσης, τα οποία θα εμπεριέχουν αυτήν την ποικιλότητα στην εμφάνιση των προσώπων. Η ταχύτητα είναι επίσης ένα σημαντικό θέμα για την απόδοση του συστήματος σε πραγματικό χρόνο. Έχει γίνει σημαντική ερευνητική προσπάθεια για την δημιουργία σύνθετων και γρήγορων ταξινομητών και από το 1990 έχει γίνει σημαντική πρόοδος στο θέμα αυτό.

2.3 Μέθοδοι Ανίχνευσης/Εντοπισμού Προσώπου

Οι υπάρχουσες τεχνικές για την ανίχνευση (face detection) ή τον εντοπισμό προσώπων (face localization) σε μονόχρωμες ή έγχρωμες εικόνες κατηγοριοποιούνται κατά τον Yang , ως εξής:

- **Μέθοδοι βασισμένες στη γνώση (knowledge-based methods):**

Αυτές οι μέθοδοι κωδικοποιούν την ανθρώπινη γνώση σε κανόνες αναφορικά με το τί συνιστά ένα τυπικό πρόσωπο. Συνήθως κωδικοποιούνται από τον ερευνητή οι σχέσεις (θέσεις, αποστάσεις) ανάμεσα στα χαρακτηριστικά του προσώπου. Οι δυσκολίες της μεθόδου οφείλονται στην μετατροπή των σχέσεων των χαρακτηριστικών σε συγκεκριμένους κανόνες. Αυτές οι μέθοδοι έχουν σχεδιαστεί για τον εντοπισμό του προσώπου.

- **Προσεγγίσεις αναλλοίωτων χαρακτηριστικών (feature invariant approaches):**

Στόχος αυτών των μεθόδων είναι να βρεθούν τα δομικά χαρακτηριστικά ενός προσώπου που υπάρχουν ακόμα και όταν η στάση, η θέση παρατήρησης ή οι συνθήκες φωτισμού ποικίλουν. Με βάση τα δομικά αυτά χαρακτηριστικά (φρύδια, μάτια, στόμα κλπ.) χτίζονται στατιστικά μοντέλα μορφών (statistical shape models), που στη συνέχεια μπορούν να χρησιμοποιηθούν για τον εντοπισμό του προσώπου. Οι δυσκολίες της μεθόδου οφείλονται στην απόκρυψη των χαρακτηριστικών ή στην ύπαρξη πρόσθετων αντικειμένων που αλλοιώνουν τις μορφές.

- **Μέθοδοι συνταιριάσματος υποδειγμάτων (template matching methods):**

Εδώ χρησιμοποιούνται διάφορα προκαθορισμένα τυπικά υποδείγματα (templates) για να περιγράψουν το πρόσωπο σαν σύνολο ή τα χαρακτηριστικά του προσώπου ξεχωριστά. Για την ανίχνευση υπολογίζεται η συσχέτιση μεταξύ της εικόνας εισόδου και των αποθηκευμένων υποδειγμάτων. Οι μέθοδοι αυτές χρησιμοποιούνται για

ανίχνευση αλλά και για εντοπισμό των προσώπων. Οι δυσκολίες της μεθόδου οφείλονται στην αντιμετώπιση των παραλλαγών στην κλίμακα, την μορφή και τον προσανατολισμό, δεδομένου ότι τα υποδείγματα είναι προκαθορισμένα.

- **Μέθοδοι βασισμένες στην εμφάνιση (appearance-based methods):**

Εδώ τα μοντέλα ή υποδείγματα προσώπου, σε αντίθεση με την μέθοδο των υποδειγμάτων όπου τα υποδείγματα είναι προκαθορισμένα, μαθαίνονται από ένα σύνολο εικόνων εκπαίδευσης που είναι αντιπροσωπευτικές της ποικιλότητας της εμφάνισης των προσώπων. Αυτά τα μοντέλα από εκμάθηση χρησιμοποιούνται στη συνέχεια για την ανίχνευση των προσώπων. Οι μέθοδοι αυτές χρησιμοποιούνται κυρίως για ανίχνευση, αλλά και για εντοπισμό των προσώπων.

Οι πιο επιτυχημένες τεχνικές για ανίχνευση προσώπου είναι οι μέθοδοι που βασίζονται στην εμφάνιση (appearance-based). Αυτές βασίζονται αποκλειστικά στην εμφάνιση των προσώπων, χωρίς να χρησιμοποιούνται άλλα ευρήματα. Η διαδικασία ανίχνευσης έχει ως εξής :

Μια εικόνα εισόδου σαρώνεται σε όλες τις δυνατές θέσεις και κλίμακες από ένα υπο-παράθυρο ανίχνευσης. Η ανίχνευση προσώπου αποφασίζεται από την ταξινόμηση του δείγματος στο υπο-παράθυρο σαν πρόσωπο ή μη-πρόσωπο. Ο ταξινομητής προσώπων / μη-προσώπων εκπαιδεύεται από ένα εκπαιδευτικό σύνολο δειγμάτων από πρόσωπα και μη-πρόσωπα χρησιμοποιώντας στατιστικές μεθόδους εκπαίδευσης.

2.4 Μικρό Ιστορικό των κυριότερων Μεθόδων Ανίχνευσης

Παρακάτω αναφέρουμε τις κυριότερες μεθόδους υποδειγμάτων και εμφάνισης για την ανίχνευση προσώπων, που επηρέασαν ουσιαστικά την εξέλιξη της έρευνας στο συγκεκριμένο τομέα:

- 1992: Craw et al.

Χρήση Υποδειγμάτων Μορφών (Shape Templates)

- 1995: Lanitis et al.

Χρήση Μοντέλων Ενεργών Μορφών (Active Shape Models)

- 1997: Osuna et al.

Χρήση μη γραμμικών Μηχανών Ανυσματικής Στήριξης (SVM)

- 1998: Papageorgiou and Poggio

Χρήση κυματιδίων Haar (Haar wavelets) με SVM ταξινομητή

- 1998: Sung and Poggio

Χρήση Γκαουσιανών κατανομών και πολυστρωματικών αντιλήπτρων (multilayer perceptron)

- 1998: Rowley et al.

Χρήση συνόλου νευρωνικών δικτύων (neural networks)

- 2000: Schneiderman and Kanade

Χρήση στατιστικής για την τοπική εμφάνιση και εκμάθηση με AdaBoost

- 2001: Viola and Jones.

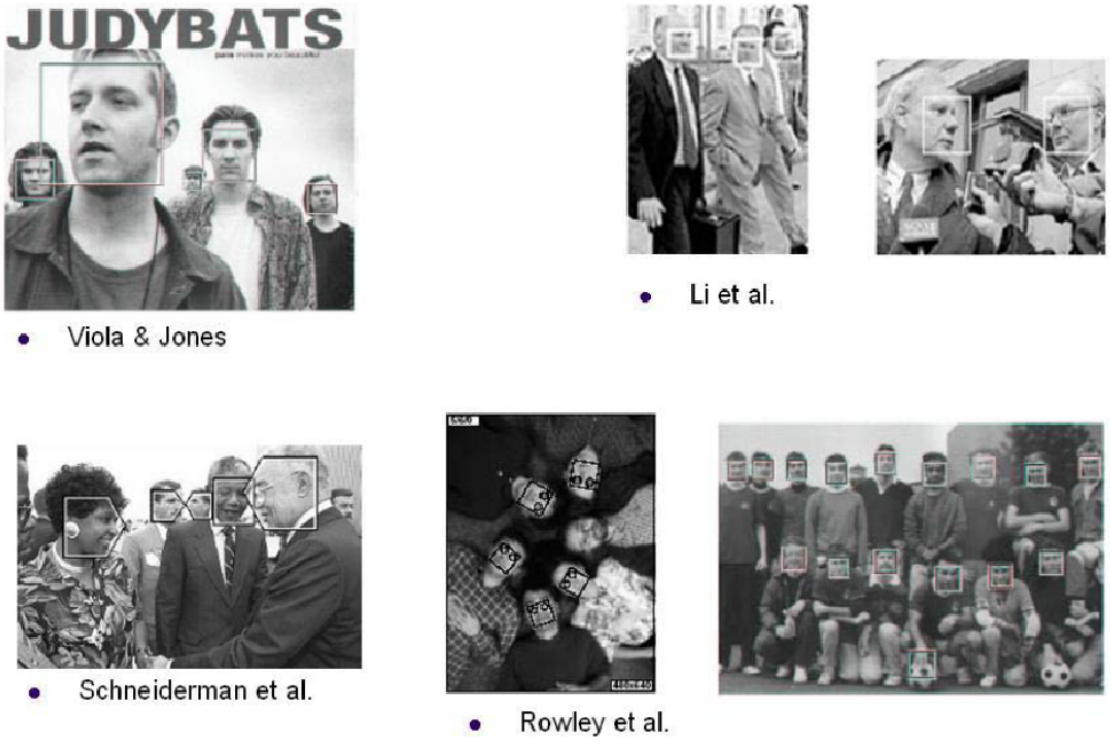
Χρήση των κυματιδίων τύπου Haar σε ταξινομητές ενός-μόνο χαρακτηριστικού, που εκπαιδεύονται με τον AdaBoost αλγόριθμο και οργανώνονται σε μια διαδοχή αποφάσεων

- 2002: Lienhart et al.

Χρήση ενός εκτεταμένου συνόλου περιστραμμένων χαρακτηριστικών Haar για την αντιμετώπιση των στροφών στο επιπεδο εικόνας

- 2002: Li et al.

Χρήση ενός επεκτεταμένου συνόλου περί στραμμένων χαρακτηριστικών Haar και ενός πυραμιδοειδούς ανιχνευτή για την αντιμετώπιση των στροφών εκτός του επιπέδου της εικόνας



Εικ. 2.1 Παραδείγματα ανίχνευσης προσώπων

Η προσέγγιση της AdaBoost εκμάθησης χαρακτηριστικών τύπου Haar των Viola Jones έχει επιτύχει την καλύτερη απόδοση μέχρι σήμερα και ως προς την ακρίβεια και ως προς την ταχύτητα.

2.5 Μέθοδοι Βασισμένες στην Εμφάνιση

Οι μέθοδοι που είναι βασισμένες στην εμφάνιση (appearance-based methods), εμφανίζουν ορισμένα κοινά χαρακτηριστικά κατά την διαδικασία ανίχνευσης των προσώπων:

- Σύνολο εκπαίδευσης θετικών και συνήθως αρνητικών δειγμάτων προσώπων
- Εκπαίδευση και χρήση ενός ταξινομητή
- Συγκεκριμένο τρόπο αναπαράστασης των προσώπων
- Προεπεξεργασία
- Συγκεκριμένη στρατηγική ανίχνευσης/αναζήτησης στο χώρο και την κλίμακα
- Μετεπεξεργασία

Τα παραπάνω κοινά χαρακτηριστικά θα εξετάσουμε αναλυτικότερα στις παρακάτω παραγράφους.

2.5.1 Δείγματα προσώπων και μη-προσώπων

Για την εκπαίδευση του ταξινομητή χρησιμοποιούνται θετικά όσο και αρνητικά δείγματα προσώπων:

- Θετικά δείγματα

Λήψη όσο το δυνατό μεγαλύτερης ποικιλίας δειγμάτων, ώστε να περιλαμβάνονται όσο το δυνατόν περισσότερες εκδοχές του προσώπου που πρέπει να ανιχνεύεται.

Κόψιμο και κανονικοποίηση κάθε εικόνας προσώπου σε ένα συγκεκριμένο μέγεθος, π.χ. 20×20 pixels (Εικ. 2.2).

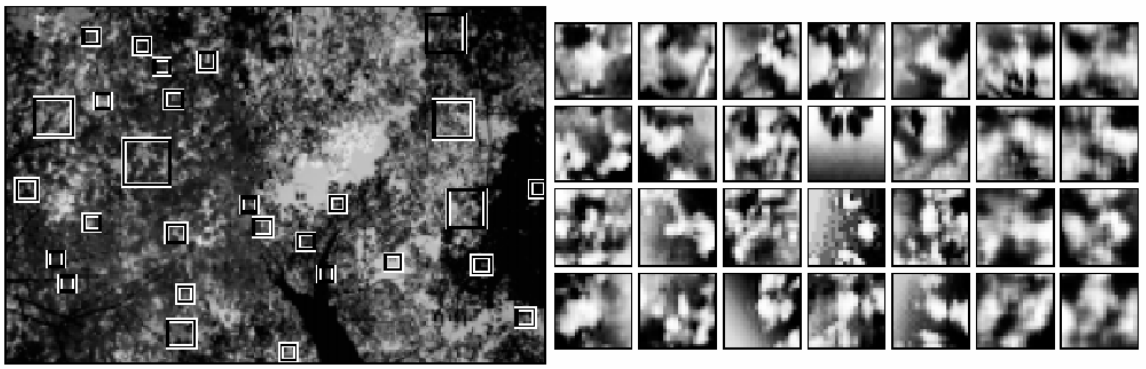
Αύξηση του πλήθους των θετικών δειγμάτων με τη δημιουργία εικονικών δειγμάτων (Εικ. 2.4)

Εικ. 2.2: Πρόσωπα που χρησιμοποιούνται στο σύνολο των θετικών δειγμάτων

- Αρνητικά δείγματα

Τα αρνητικά δείγματα λαμβάνονται από οποιαδήποτε εικόνα που δεν περιέχει πρόσωπα (Εικ. 2.3).
Οι εικόνες δείγματα των μη-προσώπων λαμβάνονται στο ίδιο μέγεθος με τις εικόνες των προσώπων.

Εφαρμόζεται η τεχνική Bootstrapping

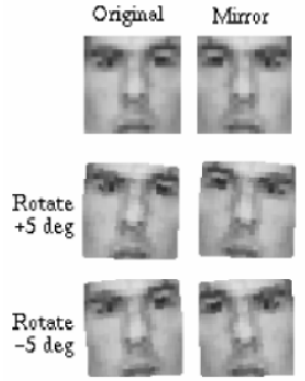


Εικ. 2.3: Μη-πρόσωπα που χρησιμοποιούνται στο σύνολο των αρνητικών δειγμάτων

Κατά τη διάρκεια της εκπαίδευσης, το μερικώς εκπαιδευμένο σύστημα εφαρμόζεται σε εικόνες οι οποίες δεν περιέχουν πρόσωπα (όπως στην Εικ. 2.3 αριστερά). Κάθε περιοχή στην εικόνα η οποία ανιχνεύεται εσφαλμένα ως πρόσωπο (τα οποία παρατίθενται στην Εικ. 2.3 δεξιά) είναι δείγματα μη-προσώπων, τα οποία προστίθενται στο σύνολο των αρνητικών δειγμάτων εκπαίδευσης.

- Εικονικά Θετικά Δείγματα

Για τον πολλαπλασιασμό του συνόλου των δειγμάτων εκπαίδευσης, δημιουργούνται με τεχνητό τρόπο νέες εικόνες-δείγματα από τις υπάρχουσες.



Εικ. 2.4: Τεχνητώς δημιουργούμενα εικονικά δείγματα προσώπων

Έτσι με κατοπτρισμό, μικρές περιστροφές, μετατοπίσεις και κλιμακώσεις των δειγμάτων των προσώπων (Εικ. 2.4), μπορούμε να πετύχουμε αύξηση του αριθμού των δειγμάτων εκπαίδευσης και μικρότερη ευαισθησία στα σφάλματα λήψης των δειγμάτων.

2.5.2 Χρησιμοποιούμενοι Ταξινομητές

Διάφοροι ταξινομητές/αλγόριθμοι έχουν χρησιμοποιηθεί κατά καιρούς σε εργασίες ανίχνευσης προσώπων, οι κυριότεροι από τους οποίους είναι:

- Αλγόριθμοι Προσαρμοστικής Ενίσχυσης (Adaptive Boosting – AdaBoost)
- Νευρωνικά Δίκτυα - Πολυστρωματικό Αντίληπτρο (Neural Networks – Multilayer Perceptron)
- Ανάλυση Θεμελειωδών Συνιστωσών (Principal Component Analysis - PCA),
- Μηχανές Ανυσματικής Στήριξης (Support Vector Machines - SVM)
- Μέθοδοι βασισμένες σε Κατανομές (Distribution-based methods)
- Απλοϊκός Ταξινομητής Bayes (Naïve Bayes classifier)
- Μοντέλα Κρυφών Μεταβλητών Markov (Hidden Markov Models)
- Επαγωγική Μάθηση (Inductive Learning)

2.5.3 Αναπαράσταση Προσώπων

Συγκεκριμένοι τρόποι αναπαράστασης των προσώπων έχουν χρησιμοποιηθεί σε διαφορετικές εργασίες, ανάλογα με τα χαρακτηριστικά (features) που έχουν επιλεγεί:

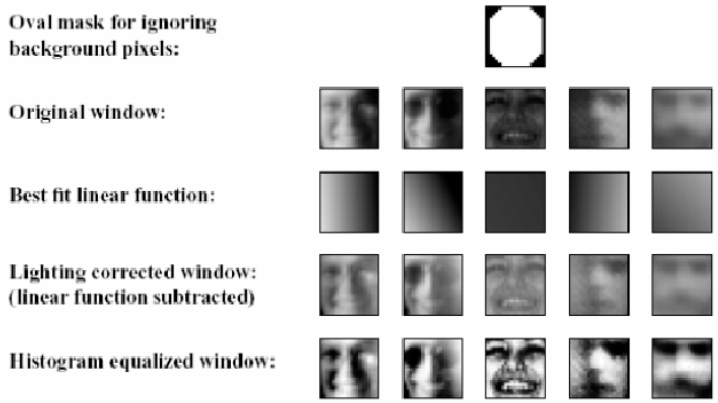
- Ολιστική (Holistic): Κάθε εικόνα προσώπου χρησιμοποιείται ολόκληρη και αναπαρίσταται από ένα διάνυσμα τιμών έντασης.
- Βασισμένη σε μπλοκ (Block-based): Αποσύνθεση κάθε εικόνας προσώπου σε ένα σύνολο επικαλυμμένων ή μη επικαλυμμένων περιοχών (μπλοκ) όπως στην Εικ. 2.5.
- Απεικόνιση σε πολλαπλές κλίμακες
- Επιπλέον επεξεργασία με Διανυσματική Κβάντιση (Vector Quantization), Ανάλυση Θεμελειωδών Συνιστωσών (Principal Component Analysis), Κυματίδια (Wavelets) κλπ.

- Χρήση χαρακτηριστικών γνωρισμάτων (features).

Εικ. 2.5: Διάφορες αναπαραστάσεις στηριγμένες στον τεμαχισμό σε μπλοκ

2.5.4 Προεπεξεργασία

Η προεπεξεργασία με κανονικοποίηση της έντασης των εικονοστοιχείων (pixels) βοηθάει στην διόρθωση των αποκλίσεων των παραμέτρων αποτύπωσης των εικόνων μέσω των καμερών, όπως επίσης και των παραλλαγών στις συνθήκες φωτισμού.



Εικ 2.6: Διαδικασίες προ-επεξεργασίας εικόνας προσώπου

Τα βήματα κατά την διαδικασία προ-επεξεργασίας στο υπο-παράθυρο της εικόνας και κατά την διαμόρφωση των δειγμάτων των προσώπων:

- Τροποποίηση μεγέθους εικόνας: Τροποποίηση όλων των προτύπων προσώπων σε ένα συγκεκριμένο μέγεθος (π.χ. σε 20×20 pixels).

- Μασκάρισμα: περιορισμός των μη επιθυμητών δομών του υποβάθρου (θόρυβος) κοντά στα όρια του πλαισίου ενός προτύπου προσώπου

- Επιδιόρθωση διαβαθμισμένου φωτισμού: ανεύρεση μιας γραμμικής συνάρτησης η οποία να ταιριάζει στην διαβάθμιση των τιμών έντασης του παραθύρου του προσώπου και της οποίας οι τιμές αφού αφαιρεθούν, επιδιορθώνουν ακραίες διαβαθμίσεις φωτισμού (σκιές που δημιουργούνται από ακραίες γωνίες φωτισμού)

- Εξισορρόπηση Ιστογράμματος: αντισταθμίζει τις επιδράσεις φωτισμού εξαιτίας διαφορετικών συνθηκών φωτισμού και διαφορετικών καμπύλων απόκρισης της κάθε κάμερας λήψης

2.5.5 Αναζήτηση στον Χώρο και την Κλίμακα

Το παράθυρο ανίχνευσης σαρώνει την υπό ανίχνευση εικόνα σε διαφορετικές κλίμακες και θέσεις:

- Σάρωση μιας εικόνας εισόδου με βήμα Δ-pixels (π.χ.. 2-pixels) οριζόντια και κάθετα

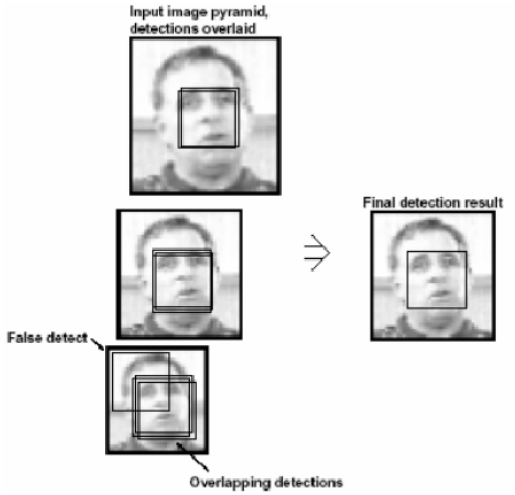
- Υποδειγματοληψία της εικόνας εισόδου κατά έναν παράγοντα και συνέχιση της αναζήτησης

- Συνέχιση της υποδειγματοληψίας της εικόνας εισόδου και αναζήτηση μέχρι η εικόνα που σαρώνεται να γίνει πολύ μικρή, ώστε να είναι αδύνατο να περιέχει πρόσωπο

Η επιλογή του βήματος σάρωσης και του παράγοντα υποδειγματοληψίας επηρεάζει την ταχύτητα, όπως επίσης και την ακρίβεια του ανιχνευτή. Αντί για σμίκρυνση της εικόνας εισόδου μπορούμε να έχουμε αντίστοιχη μεγέθυνση του παραθύρου σάρωσης

2.5.6 Μετεπεξεργασία

Από τη στιγμή που ο ανιχνευτής δεν επηρεάζεται από μικρές μεταβολές στην μετατόπιση και την κλίμακα, πολλαπλές ανιχνεύσεις μπορεί να συμβούν σε κοντινές θέσεις ή σε πολλαπλές κλίμακες κατά την σάρωση μιας εικόνας. Εσφαλμένες ανιχνεύσεις μπορεί επίσης να συμβούν, αλλά συνήθως με λιγότερη συχνότητα από ότι πολλαπλές ανιχνεύσεις προσώπων. Είναι ωστόσο χρήσιμο να γίνει μετεπεξεργασία των θετικών παραθύρων ανίχνευσης, προκειμένου να συνδυαστούν οι επικαλυπτόμενες ανιχνεύσεις σε μια μόνο ανίχνευση και να περιοριστούν οι εσφαλμένες ανιχνεύσεις.



Εικ. 2.7: Συγχώνευση πολλαπλών ανιχνεύσεων και απαλοιφή των λανθασμένων

Ο αριθμός των πολλαπλών επικαλυπτόμενων ανιχνεύσεων στη γειτονιά μιας θέσης μπορεί να χρησιμοποιηθεί σαν μια αποτελεσματική ένδειξη για την ύπαρξη ενός προσώπου σε αυτό το σημείο, ενώ σε αντίθετη περίπτωση να απορριφθεί ως εσφαλμένη ανίχνευση.

Για τον χειρισμό των πολλαπλών ανιχνεύσεων μπορούν να χρησιμοποιηθούν οι παρακάτω τεχνικές :

- Ανεύρεση επικαλυπτόμενων ανιχνεύσεων (η τομή τους να ξεπερνά ένα κατώφλι, π.χ. 80%)
- Συγχώνευση πολλαπλών ανιχνεύσεων (ο αριθμός των πολλαπλών ανιχνεύσεων να ξεπερνά ένα κατώφλι, π.χ. 4)
- Καθορισμός του τελικού παραθύρου (επιλογή του παραθύρου ανίχνευσης με τη μέγιστη εμπιστοσύνη ταξινόμησης H(x)=max, ή εύρεση του μέσου παράθυρου)

2.6 Ανίχνευση Προσώπου κατά Viola & Jones

Ένας από τους αλγόριθμους εντοπισμού με το μεγαλύτερο αντίκτυπο τη δεκαετία του 2000 ήταν αυτός που προτάθηκε από τους Viola και Jones. Αυτή η μέθοδος χρησιμοποιήθηκε και στο πρόγραμμά μας για να κάνουμε ανίχνευση προσώπου σε μια δοσμένη εικόνα. Ο ανιχνευτής τους βασίζεται σε τρεις βασικές ιδέες: τα χαρακτηριστικά τύπου **Haar**, την εκπαίδευση ταξινομιτών με τη μέθοδο **Adaboost**, και τον ταξινομητή **cascade**.

Οι Viola και Jones ανέπτυξαν μια αξιόπιστη μέθοδο για τον εντοπισμό αντικειμένων όπως πρόσωπα σε εικόνες σε πραγματικό χρόνο. Ένα αντικείμενο που πρέπει να εντοπιστεί περιγράφεται από έναν συνδυασμό χαρακτηριστικών τύπου Haar τα οποία φαίνονται στην εικόνα 2.8. Το σύνολο των pixels στα άσπρα κουτιά αφαιρείται από το σύνολο των pixels στις μαύρες περιοχές. Το πλεονέκτημα της

χρήσης αυτών των απλών χαρακτηριστικών είναι ότι μπορούν να υπολογιστούν πολύ γρήγορα χρησιμοποιώντας μια ολοκληρωτική εικόνα (integral image).

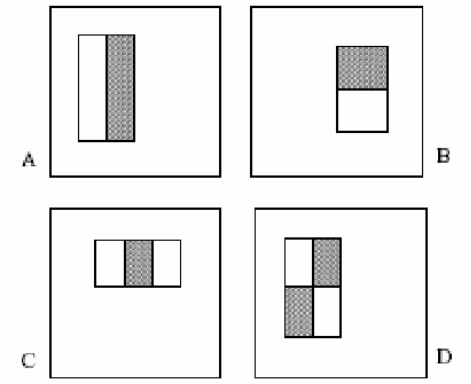
2.6.1 Υπολογισμός Χαρακτηριστικών

- Η διαδικασία ανίχνευσης ταξινομεί εικόνες στηριζόμενη στην τιμή απλών βαθμωτών χαρακτηριστικών (features).

- Η χρήση των χαρακτηριστικών αντί της εικόνας έντασης (intensity image) έχει ως σκόπο την μείωση των διαφοροποιήσεων μέσα στην κλάση (intra-class variability) και την αύξηση των διαφοροποιήσεων μεταξύ των κλάσεων (inter-class variability), ώστε η ταξινόμηση να καταστεί ευκολότερη.

- Τα χαρακτηριστικά αυτά περιέχουν γνώση από συγκεκριμένες περιοχές της εικόνας, που είναι δύσκολο να κωδικοποιηθεί χρησιμοποιώντας πεπερασμένα δεδομένα μάθησης.

- Τα χαρακτηριστικά είναι όμοια με τις συναρτήσεις βάσης τύπου Haar (Haar basis) που έχουν χρησιμοποιηθεί και από τον Papageorgiou κ.ά. για τον ίδιο σκοπό
- Εφαρμόζονται σε ασπρόμαυρες εικόνες και η τιμή τους εξαρτάται από την τιμή της υπολογιζόμενης σταθμισμένης ανάλογα με το εμβαδόν διαφοράς των αθροισμάτων των εντάσεων των εικονοστοιχείων πάνω σε ορθογώνιες περιοχές, θεωρώντας τις γκρίζες περιοχές θετικές και τις λευκές αρνητικές, όπως φαίνεται στην Εικ. 2.8.
- Τα χαρακτηριστικά καθορίζονται από την θέση, τις διαστάσεις και την τιμή τους.
- Το πλήθος των χαρακτηριστικών που δημιουργούνται για δείγματα προσώπων 24×24 pixels είναι ~ 45.000, που είναι ένα υπερπλήρες σύνολο σε σχέση με τις 576 τιμές έντασης του δείγματος. Για αυτόν τον λόγο απαιτείται μια διαδικασία επιλογής των κυριότερων χαρακτηριστικών από αυτά. Σύμφωνα με τους Viola & Jones ακόμα και ανιχνευτές με 2 χαρακτηριστικά είναι αρκετά αποτελεσματικοί

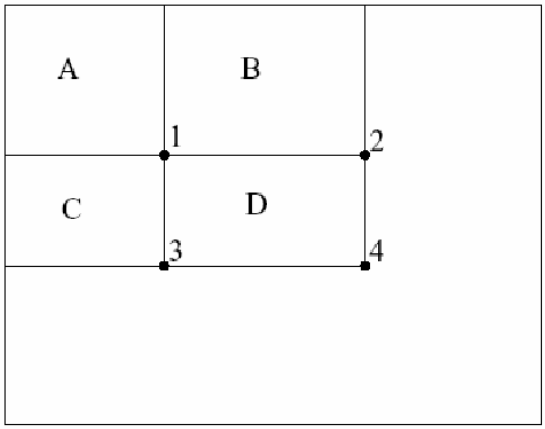


Εικ. 2.8: Μορφή των χαρακτηριστικών Viola & Jones

Τα ορθογώνια χαρακτηριστικά των Viola & Jones μπορούν να υπολογιστούν πολύ αποτελεσματικά με τη χρήση μιας βοηθητικής εικόνας, που αναφέρεται ως "εικόνα ολοκλήρωμα" (integral image). Η εικόνα ολοκλήρωμα, I, έχει τιμή στη θέση (x, y) που καθορίζεται ως άθροισμα των εντάσεων των pixels του ορθογωνίου που ορίζεται από την πάνω αριστερή κορυφή (0, 0) και την κάτω δεξιά κορυφή (x, y):

$$I(x, y) = \sum i(x', y'),\ x' \le x,\ y' \le y\ .$$

όπου i είναι η αρχική εικόνα εισόδου.



Εικ. 2.9: Αναπαράσταση της "Εικόνας Ολοκλήρωμα"

Χρησιμοποιώντας την εικόνα ολοκλήρωμα κάθε ορθογώνιο άθροισμα μπορεί να υπολογιστεί σε σταθερό χρόνο με τέσσερις αναφορές στις τιμές ενός πίνακα. Έτσι το άθροισμα εντός του D (Εικ. 2.9) μπορεί να υπολογιστεί σαν $I_4+I_1-(I_2+I_3)$.

2.6.2 Επιλογή Χαρακτηριστικών με την AdaBoost

Στο γενικότερο πρόβλημα της ενίσχυσης (boosting), συνδυάζεται ένα μεγάλο σύνολο λειτουργιών ταξινόμησης, αποδίδοντας μεγαλύτερο βάρος σε κάθε καλή λειτουργία ταξινόμησης και μικρότερο βάρος σε κάθε χειρότερη λειτουργία. Οι ταξινομητές που συνδυάζονται, ώστε να αποτελέσουν τον τελικό *ισχυρό ταξινομητή* (strong classifier), ονομάζονται *αδύναμοι ταξινομητές* (weak classifiers), και αρκεί να αποφασίζουν λίγο καλύτερα από την τυχαιότητα. Ο πιο δημοφιλής αλγόριθμος ενίσχυσης είναι ο αλγόριθμος *προσαρμοστικής ενίσχυσης* (adaptive boosting) AdaBoost, ο οποίος ονομάζεται και *διακριτός* (discrete) AdaBoost, μιας και αποδίδει διακριτές τιμές εξόδου.

Όπως ήδη έχει αναφερθεί, το πλήθος των ορθογώνιων χαρακτηριστικών που δημιουργούνται είναι υπερβολικά μεγάλο, ώστε να απαιτείται η επιλογή των πλέον αποτελεσματικών χαρακτηριστικών που θα χρησιμοποιηθούν στον τελικό ταξινομητή. Θεωρώντας μια αντιστοίχιση μεταξύ αδύναμων ταξινομητών και χαρακτηριστικών οι Viola & Jones, χρησιμοποίησαν τον αλγόριθμο AdaBoost σαν μια αποτελεσματική διαδικασία για την ανεύρεση ενός μικρού αριθμού "καλών" χαρακτηριστικών που επιπλέον είναι σημαντικά διαφοροποιημένα. Μία απλή και πρακτική μέθοδος για την

ολοκλήρωση αυτής της αντιστοίχισης είναι ο περιορισμός του αδύναμου ταξινομητή σε λειτουργίες ταξινόμησης που εξαρτώνται από ένα μόνο χαρακτηριστικό .
Έτσι, η μέθοδος AdaBoost στοχεύει στην επίλυση των παρακάτω 3 θεμελιωδών προβλημάτων

(1) εκμάθηση των πιο αποτελεσματικών χαρακτηριστικών από ένα μεγάλο σύνολο χαρακτηριστικών,

(2) κατασκευή αδύναμων ταξινομητών, καθένας από τους οποίους στηρίζεται σε ένα μόνο από τα δημιουργηθέντα χαρακτηριστικά, και

(3) συνδυασμός των αδύναμων ταξινομητών για την κατασκευή ενός ισχυρού ταξινομητή.

Σύμφωνα με τους Viola & Jones κατά την εφαρμογή του AdaBoost:

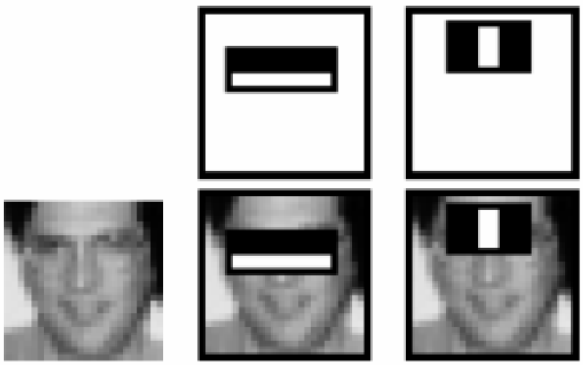
- Σε κάθε πέρασμα t εκπαιδεύεται ένας νέος αδύναμος ταξινομητής ht που προστίθεται στο σύνολο με μεγαλύτερο συντελεστή at όσο μικρότερο σφάλμα ταξινόμησης ε δίνει.

- Το βάρος κάθε δείγματος ενημερώνεται έτσι ώστε σε κάθε επόμενο πέρασμα τα δείγματα που ταξινομούνται σωστά να έχουν μικρότερη βαρύτητα κατά βt

  .

- Ο τελικός ισχυρός ταξινομητής αποτελείται από Τ αδύναμους ταξινομητές που αντιστοιχούν στα ισχυρότερα χαρακτηριστικά, με βάρη αντιστρόφως ανάλογα προς το σφάλμα ταξινόμησης.

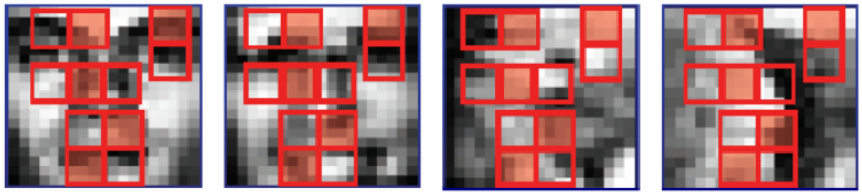
- Όπως αποδεικνύεται από τους Freund & Schapire ο ισχυρός ταξινομητής μπορεί να πετύχει αυθαίρετα υψηλό ρυθμό ορθών ταξινομήσεων με αυθαίρετα χαμηλό ρυθμό εσφαλμένων ταξινομήσεων, αρκεί το πλήθος των αδύναμων ταξινομητών να είναι αρκετά μεγάλο.



Εικ. 2.10: Τα 2 κυριότερα χαρακτηριστικά εφαρμοσμένα σε ένα τυπικό πρόσωπο

2.6.3 Κατασκευή του Αδύναμου Ταξινομητή

Ο απλούστερος τύπος ενός αδύναμου ταξινομητή είναι η "ρίζα" ("stump") ενός δένδρου απόφασης (decision tree). Όταν το χαρακτηριστικό παίρνει πραγματικές τιμές, μπορεί να κατασκευαστεί μία ρίζα απόφασης, συγκρίνοντας απλά την τιμή του επιλεγμένου χαρακτηριστικού με μια συγκεκριμένη τιμή κατωφλίου. Έτσι ο αδύναμος αλγόριθμος μάθησης σχεδιάζεται ώστε να μπορεί να επιλέγει εκείνο το μοναδικό χαρακτηριστικό που διαχωρίζει καλύτερα τα θετικά από τα αρνητικά δείγματα. Για κάθε χαρακτηριστικό, ο αδύναμος ταξινομητής καθορίζει το ιδανικό κατώφλι λειτουργίας της ταξινόμησης, έτσι ώστε να ελαχιστοποιείται ο αριθμός δειγμάτων που ταξινομείται εσφαλμένα.



Εικ. 2.11: Υπολογισμός της τιμής χαρακτηριστικών πάνω σε πρόσωπα και μη-πρόσωπα

Έτσι ο αδύναμος ταξινομητής hj(x) αποτελείται από ένα χαρακτηριστικό j και ένα κατώφλι θj:

- Για κάθε χαρακτηριστικό j, υπολογίζεται η fj(x), μια βαθμωτή τιμή του χαρακτηριστικού (εδώ οι διαφορές αθροισμάτων), όπου x είναι ένα θετικό ή αρνητικό δείγμα
- Κάθε χαρακτηριστικό χρησιμοποιείται σαν ένας αδύναμος ταξινομητής Καθορισμός τιμής κατωφλίου θj για κάθε χαρακτηριστικό έτσι ώστε τα περισσότερα δείγματα να ταξινομούνται σωστά:

$$h_j(x) = \begin{cases} 1, & \text{if } f_j(x) < \theta_j \quad (\text{or } f_j(x) > \theta_j), \quad x : \text{positive} \\ 0, & \text{otherwise}, \quad x : \text{negative} \end{cases}$$

- Επιλογή χαρακτηριστικού και κατωφλίου με το χαμηλότερο σταθμισμένο σφάλμα ταξινόμησης

- Διαδοχική εκτίμηση όλων των χαρακτηριστικών

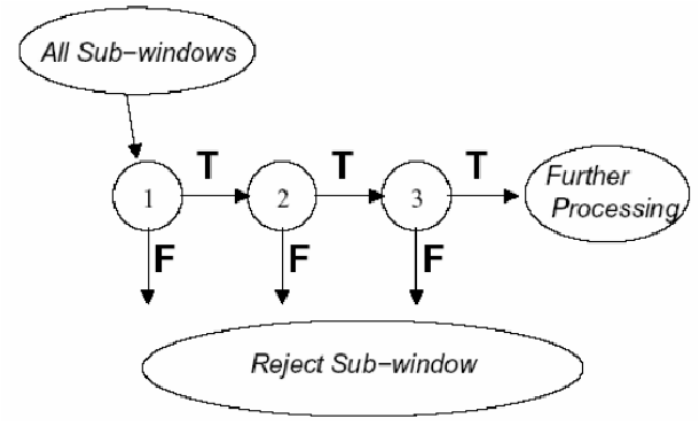
2.6.4 Ταξινόμηση με έναν Καταρράκτη Ταξινομητών

Για να αποφύγουν την σύγκλιση του AdaBoost που απαιτεί μεγάλο αριθμό δειγμάτων και καταλήγει σε μεγάλο πλήθος αδύναμων ταξινομητών που απαιτούν μεγάλο υπολογιστικό κόστος, οι Viola & Jones εισήγαγαν την έννοια του *καταρράκτη ταξινομητών* (cascade of classifiers).

Με τη μέθοδο αυτή κατασκευάζεται μία διαδοχή ταξινομητών στη μορφή καταρράκτη (cascade) που επιτυγχάνει αυξημένη απόδοση στην ανίχνευση και μειώνει ριζικά τον χρόνο υπολογισμού. Η ιδέα είναι ότι μπορούν να κατασκευαστούν μικροί και ωστόσο αποτελεσματικοί, συνδυασμένοι ταξινομητές που απορρίπτουν πολλά από τα αρνητικά, ενώ ανιχνεύουν σχεδόν όλα τα θετικά περιστατικά.

Η μέθοδος του καταρράκτη ταξινομητών στηρίζεται στο γεγονός ότι σε μία οποιαδήποτε εικόνα η πλειονότητα των παραθύρων ανίχνευσης δεν περιλαμβάνει πρόσωπα. Έτσι πιο απλοποιημένοι και λιγότερο χρονοβόροι ταξινομητές χρησιμοποιούνται για να απορρίψουν την πλειονότητα των παραθύρων ανίχνευσης ως αρνητικά, προτού χρησιμοποιηθούν οι πιο σύνθετοι και περισσότερο χρονοβόροι ταξινομητές που θα επεξεργαστούν τις πιο πολύπλοκες περιπτώσεις και θα επιτύχουν χαμηλά επίπεδα εσφαλμένων θετικών ανιχνεύσεων.

Η συνολική διαδικασία ανίχνευσης είναι παρόμοια με ένα δένδρο απόφασης (decision tree). Ένα θετικό αποτέλεσμα από τον ταξινομητή πρώτου επιπέδου οδηγείται στον ταξινομητή δευτέρου επιπέδου, του οποίου το θετικό αποτέλεσμα οδηγείται στον ταξινομητή τρίτου επίπεδου κ.ο.κ. όπως στην Εικ. 2.12. Τα αρνητικά αποτελέσματα σε κάθε επίπεδο απορρίπτονται χωρίς να επανελέγχονται. Έτσι οι ταξινομητές των αρχικών επιπέδων ασχολούνται με τα εύκολα περιστατικά, ενώ οι επόμενοι αντιμετωπίζουν πιο δύσκολες περιπτώσεις.



Εικ. 2.12: Σχηματική παράσταση μιας ανίχνευσης με καταρράκτη ταξινομητών

Η εκπαίδευση του καταρράκτη ταξινομητή γίνεται χρησιμοποιώντας τον AdaBoost, και καθορίζει:

- τον αριθμό των επιπέδων του καταρράκτη ταξινομητή
- τον αριθμό των χαρακτηριστικών σε κάθε επίπεδο
- το κατώφλι σε κάθε επίπεδο

ώστε να ελαχιστοποιείται ο αριθμός των χρησιμοποιούμενων χαρακτηριστικών.

Στην Matlab ο αλγόριθμος αυτός καλείται με την εντολή:

vision.CascadeObjectDetector()

2.6.5 MATLAB code for using the cascadeObjectDetector() function on pictures

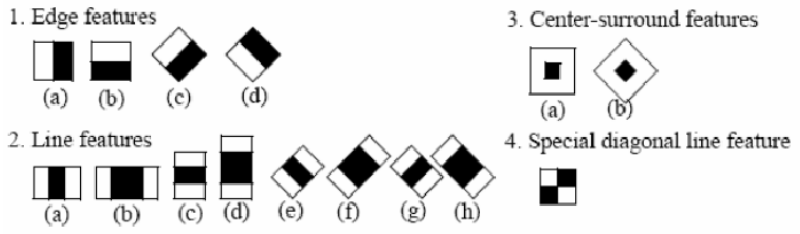
```
function [ ] = Viola_Jones_img( Img )
%Viola_Jones_img( Img )
% Img - input image
% Example how to call function:
Viola_Jones_img(imread('name_of_the_picture.jpg'))

faceDetector = vision.CascadeObjectDetector;
bboxes = step(faceDetector, Img);
figure, imshow(Img), title('Detected faces');hold on
for i=1:size(bboxes,1)
   rectangle('Position',bboxes(i,:),'LineWidth',2,'EdgeColor','y');
end
end
```

2.7 Ανίχνευση Προσώπου κατά Lienhart

Ο Lienhart κ.ά εισήγαγαν δύο κύριες βελτιώσεις στην μέθοδο ανίχνευσης των Viola & Jones, κατά πρώτον ένα εκτεταμένο σύνολο από ορθογώνια χαρακτηριστικά τύπου Haar, και κατά δεύτερον την χρήση της gentle AdaBoost για την κατασκευή του ισχυρού ταξινομητή σε συνδυασμό με τη χρήση CART δένδρων στη θέση των αδύναμων ταξινομητών.

- Το εκτεταμένο σύνολο από Haar χαρακτηριστικά:

Ο Lienhart εμπλούτισε το βασικό σύνολο των Haar χαρακτηριστικών με ένα πιο αποτελεσματικό σύνολο από 45 χαρακτηριστικά, προσθέτοντας επιπλέον γνώση στο σύστημα εκπαίδευσης και βελτιώνοντας την απόδοση του συστήματος.



Εικ. 2.13: Εκτεταμένο σύνολο στραμμένων Haar χαρακτηριστικών

Τα χαρακτηριστικά αυτά φαίνονται στην Εικ. 2.13 και αποτελούνται από τέσσερα χαρακτηριστικά ακμών (Εικ. 2.13.1), οκτώ χαρακτηριστικά γραμμών (Εικ. 2.13.2) και δύο χαρακτηριστικά κέντρων (Εικ. 2.13.3), ενώ το χαρακτηριστικό διαγώνιας γραμμής (Εικ. 2.13.4) δεν χρησιμοποιείται σε αντίθεση με τους Viola & Jones.

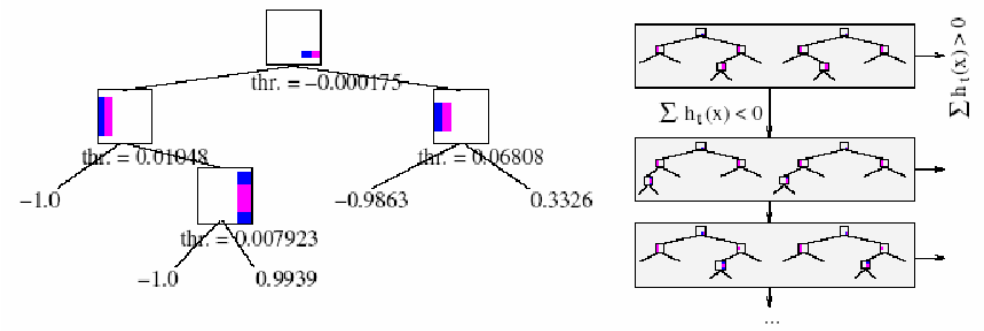
Τα χαρακτηριστικά καθορίζονται από τα ορθογώνια r = (x,y,w,h,α), α = {0° ,45°}, και παίρνουν τιμές την σταθμισμένη διαφορά των αθροισμάτων των εντάσεων στις λευκές και μαύρες περιοχές:

$$\text{feat} = \sum \omega_i \cdot \text{ReSum}(r_i)$$

Το πλήθος των ορθογώνιων χαρακτηριστικών του Lienhart για δείγματα 24×24 pixels ανέρχεται σε ~ 118.000.

- CART δένδρα ως αδύναμοι ταξινομητές:

Επίσης ο Lienhart χρησιμοποίησε ως αδύναμους ταξινομητές αντί για απλές ρίζες απόφασης (stumps), μικρά CART δένδρα με έως 4 χαρακτηριστικά. Τα Δένδρα Ταξινόμησης και Παλινδρόμησης (Classification and Regression Trees – CART) είναι μικροί ταξινομητές δένδρων απόφασης, καλοί στην εκμάθηση εξαρτήσεων μεταξύ των χαρακτηριστικών. Η συνάρτηση $h_t(x)$ καθορίζεται σύμφωνα με την διαδρομή μέσα στο δένδρο απόφασης.



Εικ. 2.14: Ενα δένδρο ταξινόμησης CART και ένας καταρράκτης από CART

Η χρήση μικρών δένδρων απόφασης εμφανίζει καλύτερη απόδοση από τις απλές ρίζες απόφασης.

- Χρήση παραλλαγών της AdaBoost:

Από τον Lienhart μελετήθηκαν τρείς παραλλαγές της AdaBoost: η Discrete AdaBoost, που χρησιμοποιήθηκε κι από τους Viola & Jones, και επι πλέον, η Real AdaBoost και η Gentle AdaBoost. Απ΄αυτές η Gentle AdaBoost εμφάνισε τα καλύτερα αποτελέσματα και σε μικρότερους χρόνους.

Σε κάθε γύρο της ενίσχυσης (boosting) προστίθεται ο ταξινομητής που ταξινομεί ορθότερα τα δείγματα εκπαίδευσης με τη χρήση του αποτελεσματικότερου CART δένδρου. Ο κάθε ισχυρός ταξινομητής F είναι ένα σταθμισμένο άθροισμα αδύναμων ταξινομητών f_i από CART δένδρα ενός ή περισσότερων κόμβων:

$$F = \text{sign}(c_1 f_1 + c_2 f_2 + \ldots + c_n f_n).$$

Τέλος, οι ισχυροί ταξινομητές συνδυάζονται σε διάταξη καταρράκτη (cascade). Καθώς αυξάνεται το επίπεδο του καταρράκτη ανιχνευτή, ο αριθμός των αδύναμων ταξινομητών που αποτελούν το ισχυρό ταξινομητή αυξάνεται, ώστε να επιτυγχάνεται ένας συγκεκριμένος ρυθμός εσφαλμένων ανιχνεύσεων για δεδομένο αριθμό επιτυχών ανιχνεύσεων.

2.8 Σύνοψη του Συστήματος Ανίχνευσης Προσώπου

- Υπολογίζονται απλά βαθμωτά χαρακτηριστικά. Υπάρχει ένας μεγάλος αριθμός από υποψήφια χαρακτηριστικά τύπου Haar.

- Επιλέγεται ένα μικρό υποσύνολο από αυτά και οι αντίστοιχοι αδύναμοι ταξινομητές από δένδρα απόφασης εκπαιδεύονται χρησιμοποιώντας AdaBoost.

- Ο ισχυρός ταξινομητής κατασκευάζεται σαν ένας γραμμικός συνδυασμός των αδύναμων και είναι το αποτέλεσμα του αλγόριθμου AdaBoost.

- Ο τελικός ανιχνευτής δημιουργείται από έναν ή από μία διαδοχή, σε διάταξη καταρράκτη, ισχυρών ταξινομητών που χρησιμοποιούν διαρκώς περισσότερα χαρακτηριστικά.

- Η προεπεξεργασία προετοιμάζει την εικόνα που θα ανιχνευτεί.

- Η μετεπεξεργασία συγχωνεύει ή απορρίπτει τις ανιχνεύσεις.

2.9 Μέθοδος Haar+AdaBoost : Συμπεράσματα

Αναφορικά με την προσέγγιση Haar+AdaBoost μπορούμε να εξάγουμε τα παρακάτω συμπεράσματα σχετικά με τα σύνολα των χαρακτηριστικών, τους αλγόριθμους boosting, τους αδύναμους ταξινομητές, τα μεγέθη των υποπαραθύρων, και τα μεγέθη του συνόλου εκπαίδευσης:

- Οι AdaBoost μέθοδοι εκπαίδευσης στην ανίχνευση προσώπων είναι οι πιο αποτελεσματικές μέχρι σήμερα. Αναφορικά με τα επίπεδα ανίχνευσης και σφαλμάτων, συγκρίνονται με την μέθοδο των νευρωνικών δικτύων του Rowley κ.ά. , αλλά σε μερικές περιπτώσεις είναι αρκετές φορές ταχύτερες.

- Ένα υπερπλήρες σύνολο από χαρακτηριστικά τύπου Haar είναι αποτελεσματικό για ανίχνευση προσώπου. Η χρήση της μεθόδου εικόνας ολοκληρώματος κάνει τον υπολογισμό αυτών των χαρακτηριστικών εφικτό και ανεξάρτητο από την κλίμακα.

- Τα εκτεταμένα χαρακτηριστικά τύπου Haar βοηθούν στην ανίχνευση των περιστραμμένων προσώπων.

- Η εκπαίδευση AdaBoost μπορεί να επιλέξει το καλύτερο υποσύνολο από ένα ευρύ σύνολο χαρακτηριστικών και να κατασκευάσει έναν ισχυρό μη γραμμικό ταξινομητή.

- Η διάταξη καταρράκτη βελτιώνει σημαντικά την ταχύτητα ανίχνευσης και μειώνει αποτελεσματικά τα σφάλματα με μικρό κόστος στους χρόνους ανίχνευσης.

- Πιο σύνθετοι αδύναμοι ταξινομητές όπως μικρά δένδρα CART μπορούν να μοντελοποιήσουν δεύτερης ή/και τρίτης τάξης εξαρτήσεις των χαρακτηριστικών, και μπορούν να είναι επωφελείς σε μη γραμμική επεξεργασία της ανίχνευσης προσώπου.

- Το ιδανικό μέγεθος του υπο-παραθύρου για την επεξεργασία ανίχνευσης προσώπων φαίνεται να είναι 20×20~24×24 pixels.

- Πιθανές βελτιώσεις μπορεί να είναι εφικτές με τον σχεδιασμό επιπλέον χαρακτηριστικών συμπληρωματικά στα ήδη υπάρχοντα, που υιοθετούν πιο προχωρημένες τεχνικές εκπαίδευσης, και που θα μπορούσαν να καταλήξουν σε πιο σύνθετους ταξινομητές, αποφεύγοντας το πρόβλημα της υπερπροσαρμογής (overfitting).

- Γρήγορο και σχεδόν εύρωστο σύστημα που τρέχει σε πραγματικό χρόνο

- Αρνητικά στοιχεία :

  ο Απαιτείται η αναζήτηση στο χώρο και την κλίμακα

  ο Απαιτεί αρκετά θετικά και αρνητικά δείγματα

  ο Ανάλωση πολύ χρόνου στη φάση της εκπαίδευσης (μπορεί να απαιτεί μέρες εκπαίδευσης)

  ο Περιορισμένη προσέγγιση των περιπτώσεων διαφορετικού προσανατολισμού των προσώπων

  ο Απαιτεί αρκετή εργασία υλοποίησης.

Κεφάλαιο 3

Τεχνικές αναγνώρισης προσώπων

Τα τελευταία χρόνια πολλές τεχνικές έχουν προταθεί για το πρόβλημα της αναγνώρισης προσώπου. Παρακάτω θα αναλύσουμε κάποιες από αυτές και μέσω του εργαστηριακού μέρους θα εκπονήσουμε διάφορα συμπεράσματα.

3.1 Ανάλυση σε Κύριες Συνιστώσες (Principal Component Analysis,PCA)

Η ανάλυση σε Κύριες Συνιστώσες (Principal Component Analysis) ή αλλιώς γνωστή ως τεχνική eigenfces έχει χαρακτηρισθεί ως ένα από τα πιο πολύτιμα εργαλεία που προσέφερε η γραμμική άλγεβρα στην επιστημονική κοινότητα. Χρησιμοποιείται ευρέως σε πληθώρα επιστημονικών εφαρμογών, από την νευροφυσιολογία ως και τα γραφικά υπολογιστών. Αποτελεί μια εφαρμογή του μετασχηματισμού Karhunen-Loeve.

Είναι μια μέθοδος που χρησιμοποιείται ως στατιστική προσέγγιση αναγνώρισης μοτίβων, επεξεργασίας σημάτων και μείωσης του όγκου των δεδομένων (συμπίεση) όπως και κατά την εξαγωγή χαρακτηριστικών ενός σήματος.

Μια εικόνα προσώπου σε δύο διαστάσεις με μέγεθος NxN μπορεί να θεωρηθεί ένα μονοδιάστατο διάνυσμα με διάσταση N^2. Ένα σύνολο εικόνων μπορεί να «χαρτογραφηθεί» ως ένα σύνολο σημείων στον πολυδιάστατο αυτόν χώρο. Οι εικόνες προσώπων έχοντας παρόμοιες συνθέσεις, δεν κατανέμονται τυχαία στον χώρο αυτό και εμπίπτουν χωροταξικά σε ένα μικρό κομμάτι του. Η κεντρική ιδέα των κύριων συνιστωσών (Principal Components) είναι η εύρεση εκείνων των διανυσμάτων των οποίων οι γραμμικοί συνδυασμοί περιγράφουν ικανοποιητικά τις κατανομές των εικόνων προσώπων και ορίζουν ένα υποσύνολο που ονομάζεται

«χώρος προσώπων» (face space). Καθένα απ' αυτά τα διανύσματα μεγέθους N^2 περιγράφει μια εικόνα NxN.

Η προσέγγιση αναγνώρισης προσώπου με την χρήση των Κύριων Συνιστωσών είναι μια μέθοδος κατά την οποία ένα μικρό σύνολο «χαρακτηριστικών» χρησιμοποιούνται για να περιγράψουν τις διαφορές ανάμεσα στις εικόνες προσώπων.

Τα χαρακτηριστικά αυτά μπορεί παραδείγματος χάρη να είναι οι συντεταγμένες μιας εικόνας στον χώρο προσώπων. Στη συνέχεια κάθε πρόσωπο περιγράφεται σαν γραμμικός συνδυασμός των ιδιοδιανυσμάτων. Η αναγνώριση εκτελείται με α) την προβολή της νέας εικόνας στη βάση του χώρου προσώπων και β) την εύρεση της «κοντινότερης» γνωστής εικόνας βάσης σύμφωνα με κάποια μετρική ομοιότητας.

3.1.1 Μαθηματική ανάλυση αλγορίθμου

Θεωρούμε ένα σύνολο εικόνων προσώπων Γ_1, Γ_2, Γ_3, ..., Γ_M σε μορφή διανυσμάτων. Ο μέσος όρος του συνόλου (μέσο πρόσωπο) ορίζεται από την σχέση:

$$\Psi = \frac{1}{M} \sum_{\nu=1}^{M} \Gamma_\nu$$

Ένα παράδειγμα συνόλου εικόνων προσώπου φαίνεται στην Εικόνα 3.1. Το μέσο πρόσωπο Ψ του συνόλου αυτού φαίνεται στην Εικόνα 3.2. Κάθε πρόσωπο διαφέρει από το μέσο βάση του διανύσματος:

$$\Phi_i = \Gamma_i - \Psi$$

Το σύνολο αυτό των διανυσμάτων, αποτελεί στη συνέχεια αντικείμενο εφαρμογής της Ανάλυσης Κύριων Συνιστωσών, που αναζητά ένα υποσύνολο Μ ορθοκανονικών διανυσμάτων, που περιγράφει με τον καλύτερο δυνατό τρόπο την κατανομή της πληροφορίας των προσώπων. Το κ-οστό διάνυσμα Uκ επιλέγεται έτσι ώστε να μεγιστοποιείται η ποσότητα:

$$\lambda_\kappa = \frac{1}{M}\sum_{\nu=1}^{M}(U_k^T \Phi_\nu)^2$$

υπό τον περιορισμό ότι:

$$U_I^T U_k = \delta_{I\kappa} = \begin{cases} 1 \text{ εάν } I = k \\ 0 \text{ εάν } I \neq k \end{cases}$$

Τα διανύσματα U_k και οι τιμές λ_κ είναι τα ιδιοδιανύσματα και οι ιδιοτιμές του πίνακα συνδιακύμανσης:

$$C = \frac{1}{M}\sum_{\nu=1}^{M} \Phi_\nu \Phi_\nu^T = AA^T$$

όπου $A=[\Phi 1\ \Phi 2 \ldots \Phi M]$. Ο πίνακας συνδιακύμανσης C είναι ένας $N^2 \times N^2$ πίνακας με πραγματικά στοιχεία και ο υπολογισμός των N^2 ιδιοδιανυσμάτων είναι αυξημένης πολυπλοκότητας για τυπικά μεγέθη εικόνων. Για το λόγο αυτό χρησιμοποιούμε την παρακάτω διαδικασία.

Έστω ιδιοδιανύσματα v_i του A^TA τέτοια ώστε:

$$A^T A v_i = \mu_i v_i.$$

Πολλαπλασιάζοντας και τα δύο μέρη με το A έχουμε:

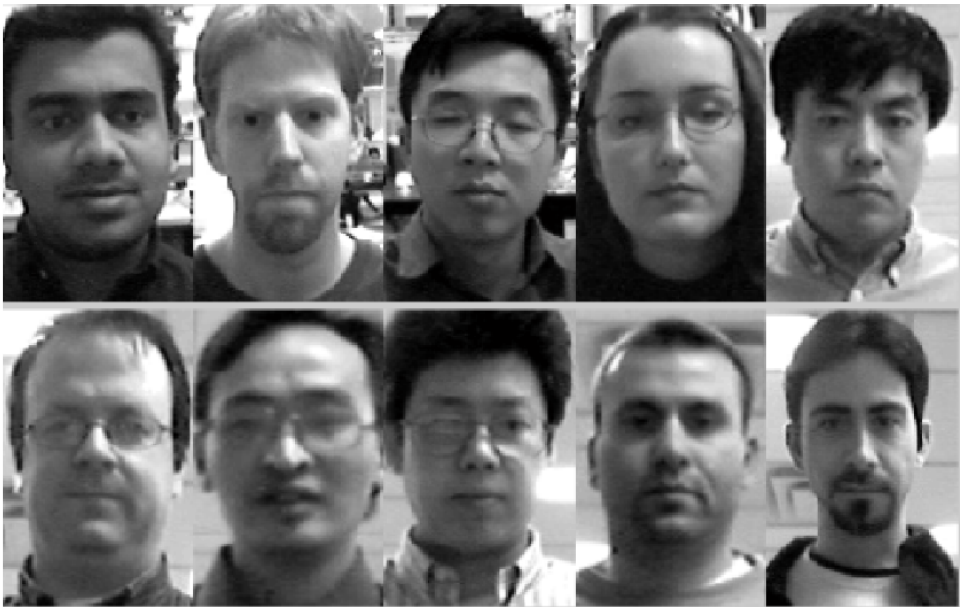
$$AA^T Av_i = A\mu_i v_i.$$

Έστω $Av_i=Q$, αντικαθιστώντας έχουμε:

$$AA^T Q = \mu_i Q$$

όπου $Q=Av_i$ είναι τα ιδιοδιανύσματα και μ_i οι ιδιοτιμές του C. Βάση της παραπάνω ανάλυσης αρκεί να δημιουργήσουμε έναν M×M πίνακα $L=A^TA$, όπου $Lmn=\Phi m^T \Phi n$, και να βρούμε τα M ιδιοδιανύσματα v_i του L. Τα διανύσματα αυτά καθορίζουν γραμμικούς συνδυασμούς των M εικόνων προσώπων για τη δημιουργία των ιδιοπροσώπων U_I:

$$U_I = \sum_{k=1}^{M} v_{Ik}\Phi_k, \; I = 1, \ldots . M$$



Εικ. 3.1: Δείγμα εικόνων συνόλου εκπαίδευσης

Εικ. 3.2: *Μέσο πρόσωπο*

Εικ. 3.3: Τα πρώτα 5 ιδιοπρόσωπα

Με την ανάλυση αυτή οι υπολογισμοί μειώνονται δραστικά, από το βαθμό του αριθμού των εικονοστοιχείων των εικόνων (N^2) στον βαθμό του αριθμού των εικόνων του συνόλου *M*. Στην πράξη το σύνολο των εικόνων είναι αρκετά μικρό ($M<<N^2$), και οι υπολογισμοί είναι λιγότερο πολύπλοκοι. Οι αντίστοιχες ιδιοτιμές επιτρέπουν τον χαρακτηρισμό τον ιδιοδυανυσμάτων βάσει της «χρησιμότητας» τους ως προς τον χαρακτηρισμό της διαφορετικότητας ανάμεσα στις εικόνες.

Οι εικόνες ιδιοπροσώπων που υπολογίζονται από τα ιδιοδιανύσματα L δημιουργούν ένα σύνολο που μπορεί να περιγράψει τις εικόνες προσώπων. Στην Εικόνα 3.3 φαίνονται τα πρώτα πέντε ιδιοπρόσωπα του συνόλου της Εικόνας 3.1. Οι Sirovich και Kirby εφάρμοσαν την παραπάνω διαδικασία σε ένα σύνολο από 115 εικόνες (M=115) καυκάσιων ανδρών ψηφιοποιημένες με συγκεκριμένες παραμέτρους και βρήκαν ότι 40 ιδιοπρόσωπα (M΄=40) ήταν αρκετά για μια πολύ καλή περιγραφή των εικόνων προσώπων. Στην πράξη ένα μικρότερο M΄ μπορεί να είναι ικανό για την αναγνώριση μιας και δεν απαιτείται η ακριβής ανακατασκευή της εικόνας. Για την αναγνώριση προσώπων η διαδικασία είναι περισσότερο μια διαδικασία αναγνώρισης μοτίβου παρά μια ανακατασκευή της εικόνας. Τα ιδιοπρόσωπα καταλαμβάνουν ένα M΄- διάστατο υποσύνολο του αρχικού N^2 «χώρου προσώπου» και έτσι τα M΄ ιδιοδιανύσματα του πίνακα L μαζί με τις αντίστοιχες ιδιοτιμές είναι αρκετά για την

αξιόπιστη αναπαράσταση των εικόνων στων χώρο προσώπων που χαρακτηρίζεται από τα ιδιοπρόσωπα.

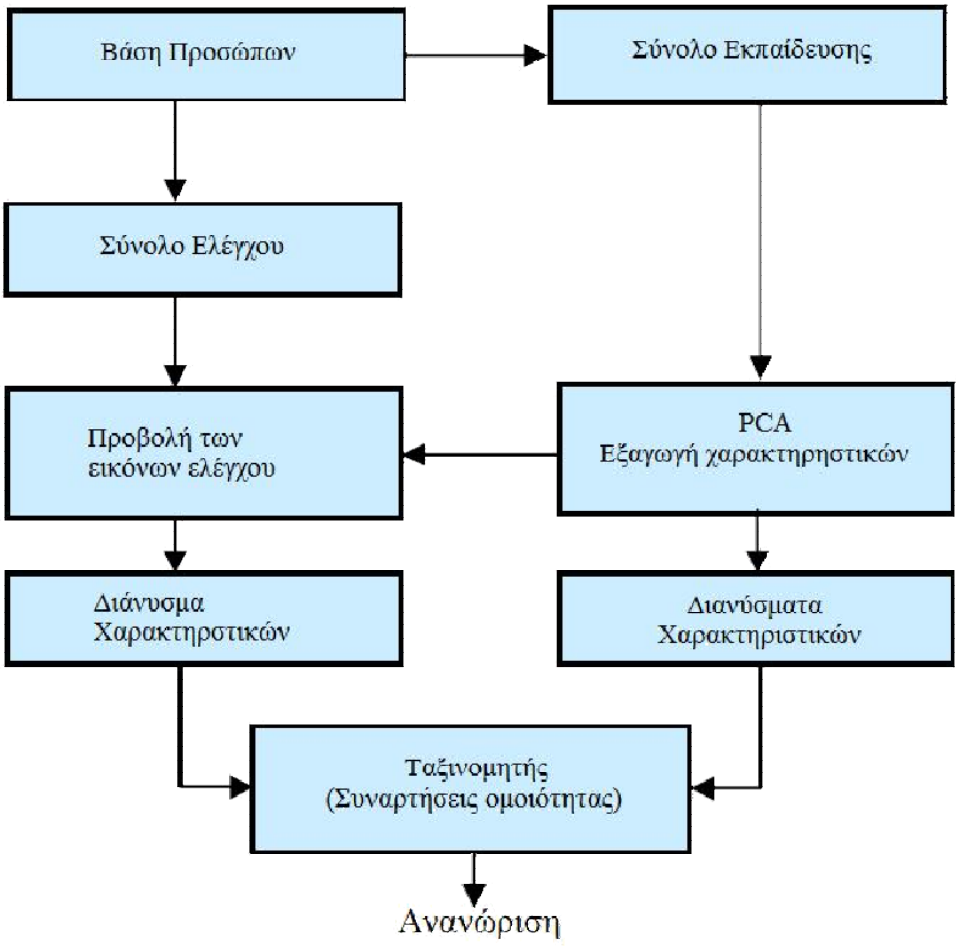
Μια νέα εικόνα προσώπου Γ περιγράφεται ως συνιστώσα των ιδιοπροσώπων (προβολή στον «χώρο εικόνας») με μια απλή διαδικασία

$$w_k = U_k^T . (\Gamma - \Psi)$$

Για *k=1,…., M΄*. Τα βάρη $w_κ$ δημιουργούν ένα διάνυσμα προβολής

$$\Omega_\kappa^T = [w_1\ w_1\ \ldots .\ w_{M'}]$$

το οποίο περιγράφει τη συμβολή κάθε ιδιοπροσώπου στην αναπαράσταση της εικόνας που εισάγεται, χρησιμοποιώντας έτσι τα ιδιοπρόσωπα σαν μία βάση για τις εικόνες προσώπων. Η κλάση προσώπου μπορεί να υπολογιστεί βρίσκοντας το μέσο ιδιοπρόσωπο από έναν μικρό αριθμών εικόνων προσώπου του κάθε ατόμου. Η ταξινόμηση επιτυγχάνεται με την σύγκριση των προβαλλόμενων διανυσμάτων των εικόνων εκπαίδευσης με το προβαλλόμενο διάνυσμα της εικόνας που εισάγεται (εικόνα ελέγχου). Η σύγκριση αυτή γίνεται με την χρήση συναρτήσεων ομοιότητας μεταξύ των κλάσεων των προσώπων και της εικόνας ελέγχου. Βρίσκοντας την κλάση της βάσης προσώπων για την οποία η συνάρτηση ομοιότητας έχει τη βέλτιστη τιμή ολοκληρώνεται η διαδικασία αναγνώρισης.



Εικ. 3.4: *Ο αλγόριθμος αναγνώρισης προσώπων PCA*

3.2 Δυσδιάστατος συντελεστής συσχέτισης (2d correlation coefficient)

Ένας συντελεστής συσχέτισης είναι ένας αριθμός που ποσοτικοποιεί κάποιο είδος συσχέτισης και εξάρτησης, πράγμα που σημαίνει στατιστική σχέση μεταξύ δύο ή περισσοτέρων τυχαίων μεταβλητών .

Δυο τυχαίες μεταβλητές X και Y μπορεί να συσχετίζονται με κάποιο τρόπο. Αυτό συμβαίνει όταν επηρεάζει η μία την άλλη, ή αν δεν αλληλοεπηρεάζονται όταν επηρεάζονται και οι δύο από μια άλλη μεταβλητή.

3.2.1 Ο συντελεστής συσχέτισης ρ

Ας θεωρήσουμε δύο τυχαίες μεταβλητές Χ και Υ με διασπορά σ^2_X και σ^2_Y αντίστοιχα και συνδιασπορά

$\sigma_{XY} \equiv Cov(X, Y) = E(X, Y) - E(X)E(Y)$.

σχέση 3.2.1

Η συνδιασπορά εκφράζει τη γραμμική συσχέτιση δύο τυχαίων μεταβλητών, δηλαδή την αναλογική μεταβολή (αύξηση ή μείωση) της μιας τυχαίας μεταβλητής που αντιστοιχεί σε μεταβολή της άλλης μεταβλητής. Η συνδιασπορά είναι ένα ποσοτικό μέγεθος κι η μονάδα μέτρησης της εξαρτάται από τις μονάδες μέτρησης των δύο τ.μ. Χ και Υ . Γι αυτό για να μετρήσουμε καλύτερα τον βαθμό της γραμμικής συσχέτισης δύο τ.μ. χρησιμοποιούμε τον συντελεστή συσχέτισης (correlation coefficient) ρ που ορίζεται ως

$$\rho = \frac{\sigma_{XY}}{\sigma_X \, \sigma_Y}$$

σχέση 3.2.2

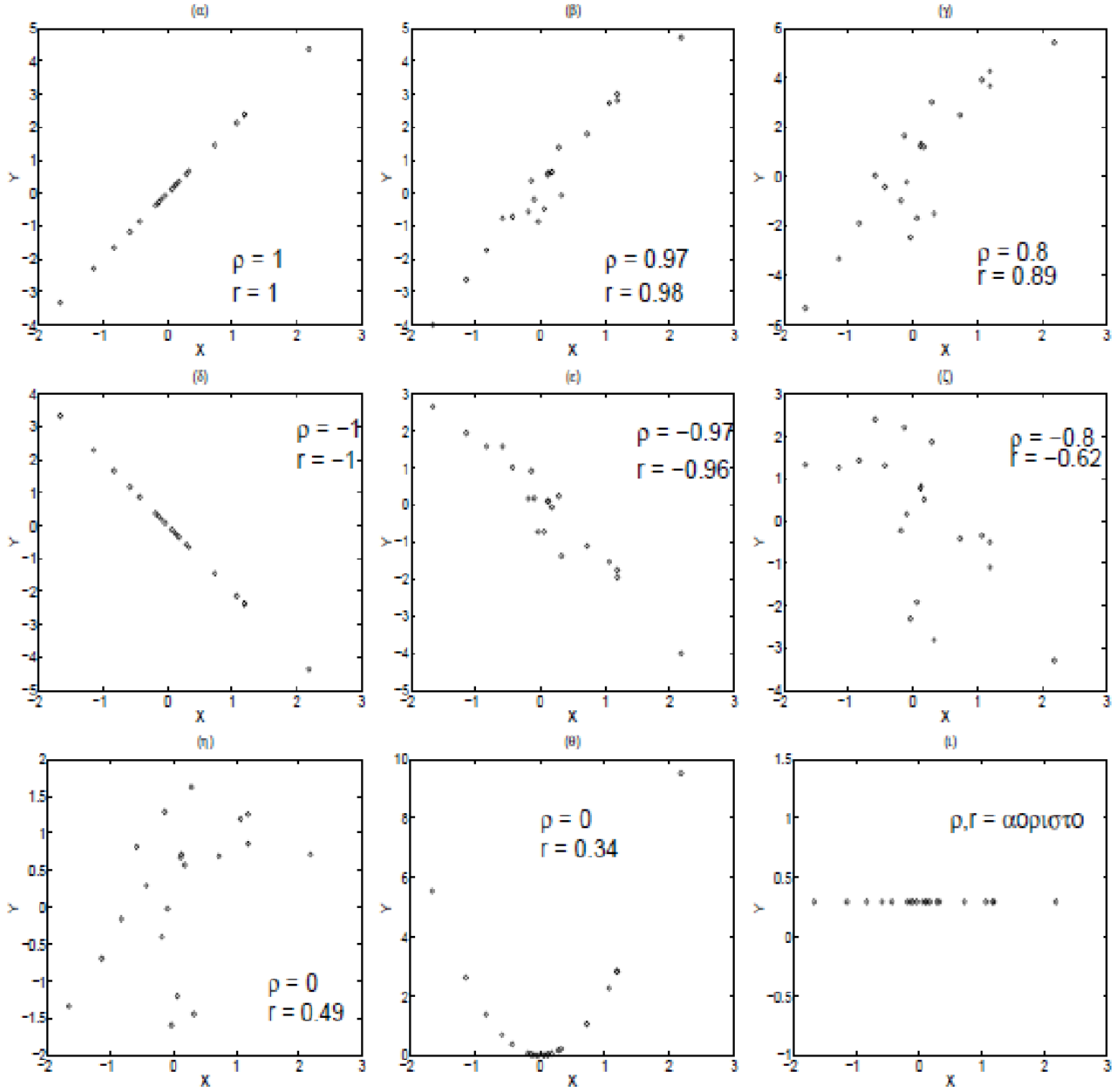
Ο συντελεστής συσχέτισης ρ παίρνει τιμές στο διάστημα [−1, 1]:

- ρ = 1: υπάρχει τέλεια θετική σχέση μεταξύ των Χ και Υ ,
- ρ = 0: δεν υπάρχει καμιά (γραμμική) σχέση μεταξύ των Χ και Υ ,
- ρ = −1: υπάρχει τέλεια αρνητική σχέση μεταξύ των Χ και Υ .

Όταν ρ = ±1 η σχέση είναι αιτιοκρατική κι όχι πιθανοκρατική γιατί γνωρίζοντας την τιμή της μιας τ.μ. γνωρίζουμε και την τιμή της άλλης τ.μ. ακριβώς. Όταν ο συντελεστής συσχέτισης είναι κοντά στο −1 ή 1 η γραμμική συσχέτιση των δύο τ.μ. είναι ισχυρή (συνήθως χαρακτηρίζουμε ισχυρές τις σχέσεις όταν $|\rho| > 0.9$) ενώ όταν είναι κοντά στο 0 οι τ.μ. είναι πρακτικά ασυσχέτιστες.
Όπως φαίνεται από τον ορισμό στη σχέση (3.2.2), ο συντελεστής συσχέτισης ρ δεν εξαρτάται από τη μονάδα μέτρησης των Χ και Υ και είναι συμμετρικός ως προς τις Χ και Υ .

3.2.2 Σημειακή εκτίμηση του συντελεστή συσχέτισης

΄Οταν έχουμε παρατηρήσεις των δύο τ.μ. Χ και Υ κατά ζεύγη
{(x1, y1), (x2, y2), . . . , (xn, yn)},
μπορούμε να εκτιμήσουμε τη συσχέτιση τους ποιοτικά από το **διάγραμμα διασποράς** (scatter diagram), που είναι η απεικόνιση των σημείων (x_i, y_i), i = 1, . . . , n, σε καρτεσιανό σύστημα συντεταγμένων. Στο Σχήμα 3.2 παρουσιάζονται τυπικά διαγράμματα διασποράς για ισχυρές κι ασθενείς συσχετίσεις δύο τ.μ. Χ και Υ . Στα Σχήματα 3.2α και 3.12 η σχέση είναι τέλεια (ρ = 1 και ρ = −1 αντίστοιχα), στα Σχήματα 3.2β και 3.2ε είναι ισχυρή (θετική με ρ = 0.97 κι αρνητική με ρ = −0.97 αντίστοιχα) και στα Σχήματα 3.2γ και 3.2C είναι λιγότερο ισχυρή (θετική με ρ = 0.8 κι αρνητική με ρ = −0.8 αντίστοιχα). Στο Σχήμα 3.2 η είναι ρ = 0 γιατί οι τ.μ. Χ και Υ είναι ανεξάρτητες ενώ στο Σχήμα 3.2θ είναι πάλι ρ = 0 αλλά οι Χ και Υ δεν είναι ανεξάρτητες αλλά συσχετίζονται μόνο μη-γραμμικά. Τέλος για το Σχήμα 3.2ι ο συντελεστής συσχέτισης δεν ορίζεται γιατί η Υ είναι σταθερή ($\sigma_Y = 0$ στον ορισμό του ρ στην (3.2.2)).



Σχήμα 3.2: Διάγραμμα διασποράς δύο τ.μ. X και Y από n = 20 παρατηρήσεις που παρουσιάζουν θετική σχέση στα σχήματα (α), (β) και (γ), αρνητική σχέση στα σχήματα (δ), (ε) και (ζ) και καμιά συσχέτιση στα σχήματα (η), (θ) και (ι). Σε κάθε σχήμα δίνεται η πραγματική τιμή του συντελεστή συσχέτισης ρ κι η δειγματική r. Στο (ι) ο συντελεστής συσχέτισης δεν ορίζεται.

Η σημειακή εκτίμηση του συντελεστή συσχέτισης ρ του πληθυσμού από το δείγμα των n ζευγαρωτών παρατηρήσεων των X και Y γίνεται με την αντικατάσταση στη σχέση (3.2.2) της συνδιασποράς σ_{XY} και των διασπορών σ^2_X και σ^2_Y από τις αντίστοιχες εκτιμήσεις από το δείγμα

$$\hat{\rho} \equiv r = \frac{s_{XY}}{s_X \, s_Y}$$

σχέση 3.2.3

Οι αμερόληπτες εκτιμήτριες s_{XY} , s^2_X και s^2_Y δίνονται ως

$$s_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}\right)$$

$$s_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)$$

όπου $\bar{x}$ και $\bar{y}$ είναι οι δειγματικές μέσες τιμές των X και Y . Από τα παραπάνω προκύπτει η έκφραση της εκτιμήτρια r

$$r = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)\left(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2\right)}}$$

Σχέση 3.2.4

Στο Σχήμα 3.2 δίνεται ο δειγματικός συντελεστής συσχέτισης r για κάθε περίπτωση. Επειδή το δείγμα είναι μικρό (n = 20) η τιμή του r δεν είναι πάντα κοντά στην πραγματική τιμή ρ. Αυτό συμβαίνει γιατί η εκτιμήτρια r όπως δίνεται στη σχέση (3.2.4) είναι μια τ.μ. που εξαρτάται από τις τιμές και το πλήθος των ζευγών των παρατηρήσεων.

Στην υλοποίηση που κάναμε σε matlab η συνάρτηση που μας δίνει αυτή την τιμή είναι η συνάρτηση corr2(A,B) η οποία υπολογίζει τον συντελεστή συσχέτισης μεταξύ των A και B. Τα A και B είναι πίνακες ή διανύσματα του ίδιου μεγέθους.

```
function r = corr2(varargin)
%CORR2 2-D correlation coefficient.
%   R = CORR2(A,B) computes the correlation coefficient between A
%   and B, where A and B are matrices or vectors of the same size.
%
%   Class Support
%   -------------
%   A and B can be numeric or logical.
%   R is a scalar double.
%
%   Example
%   -------
%   I = imread('pout.tif');
```

```
%   J = medfilt2(I);
%   R = corr2(I,J)
%
%   See also CORRCOEF, STD2.

%   Copyright 1992-2010 The MathWorks, Inc.
%   $Revision: 5.18.4.11 $  $Date: 2011/08/09 17:49:25 $

[a,b] = ParseInputs(varargin{:});

a = a - mean2(a);
b = b - mean2(b);
r = sum(sum(a.*b))/sqrt(sum(sum(a.*a))*sum(sum(b.*b)));

%--------------------------------------------------------
function [A,B] = ParseInputs(varargin)

narginchk(2,2);

A = varargin{1};
B = varargin{2};

validateattributes(A, {'logical' 'numeric'}, {'real'}, mfilename, 'A', 1);
validateattributes(B, {'logical' 'numeric'}, {'real'}, mfilename, 'B', 2);

if any(size(A)~=size(B))
   error(message('images:corr2:notSameSize'))
end

if (~isa(A,'double'))
   A = double(A);
end

if (~isa(B,'double'))
   B = double(B);
end
```

3.3 Αναγνώριση προσώπων με LDA

Η τεχνική LDA (Linear Discriminant Analysis) είναι μία προσέγγιση που βασίζεται στη στατιστική ταξινόμηση των δειγμάτων, που ανήκουν σε μία άγνωστη κλάση, σε κλάσεις που δημιουργήθηκαν από τα δεδομένα εκπαίδευσης.

Η εν΄λόγω τεχνική έχει σαν στόχο την μεγιστοποίηση της διασποράς μεταξύ διαφορετικών κλάσεων και την ελαχιστοποίηση της διασποράς εντός της κάθε κλάσης. Όταν έχουμε να κάνουμε με δεδομένα που χαρακτηρίζονται με διανύσματα πολλών διαστάσεων, για να μην υπάρξουν προβλήματα, θα πρέπει τα δεδομένα εκπαίδευσης να είναι σχετικά πολλά.

Έστω οι κλάσεις των εικόνων εκπαίδευσης c_1, c_2, …c_L με N_1, N_2, …N_L στοιχεία η κάθε μία αντίστοιχα. Έστω επίσης οι μέσες εικόνες κάθε κλάσης M1, M2, …ML και η ολική μέση εικόνα M. Υπολογίζουμε τους εντός κλάσης και ανάμεσα στις κλάσεις πίνακες διασποράς :

$$F_w = \sum_{i=1}^{L} p(c_i) A_i = \sum_{i=1}^{L} E\{[x - M_i][x - M_i]^T\}$$

$$F_b = \sum_{i=1}^{L} p(c_i)[M_i - M][M_i - M]^T$$

όπου p(ci) η εκ των προτέρων πιθανότητα της κλάσης c_i και A_i ο πίνακας συμμεταβλητότητας της κλάσης ci. Η μέθοδος LDA δημιουργεί έναν πίνακα Ψ, τα διανύσματα του οποίου αποτελούν μια βάση του χώρου R^n και ο οποίος μεγιστοποιεί τον λόγο

$$\frac{\Psi^T F_b \Psi}{\Psi^T F_w \Psi}$$

Για τη μεγιστοποίηση του λόγου αυτού ο πίνακας Ψ πρέπει να αποτελείται από τα ιδιοδυανύσματα του πίνακα $F_w^{-1}F_b$:

$$F_w^{-1}F_b\Psi = \Psi D$$

όπου οι πίνακες Ψ, D ανήκουν R^{nxn} είναι οι πίνακες ιδιοδυανυσμάτων και ιδιοτιμών του πίνακα $F^{w-1}F_b$. Ενώ οι πίνακες F_w και F_b είναι συμμετρικοί ο πίνακας δεν είναι

$F_w^{-1}F_b$ πάντα συμμετρικός. Σε κάθε περίπτωση όμως οι πίνακες Ψ και D μπορούν να υπολογιστούν ως αποτέλεσμα ταυτόχρονης διαγωνιοποίησης των Fw και Fb. Η μέθοδος LDA αποφεύγει ένα από τα μειονεκτήματα της PCA (eigenfaces) και συγκεκριμένα αυτό της ικανότητας διαχωρισμού των κλάσεων. Η μέθοδος LDA αποφεύγει ένα από τα μειονεκτήματα της PCA τεχνικής, το πρόβλημα της ικανότητητας. Αυτό επιτυγχάνεται με την εύρεση των αξόνων που μεγιστοποιείται ταυτόχρονα η ολική διασπορά του πληθυσμού αλλά και των επιμέρους κλάσεων όπως αυτές εκφράζονται από το κέντρο τους. Για την ικανοποίηση αυτής της απαίτησης καταλήγουμε σε μη ορθογώνιους άξονες, πράγμα που δεν συμβαίνει με την PCA.

Στην περίπτωση της αναγνώρισης προσώπων η μέθοδος LDA παρουσιάζει δύο σημαντικά προβλήματα :

- Ο μεγάλος αριθμός κλάσεων με λίγες εικόνες σε κάθε κλάση, δημιουργεί πρόβλημα όσων αφορά την εκτίμηση των πινάκων A_i και κατά συνέπεια του πίνακα F_b.

- Λόγου του μεγάλου μεγέθους του πίνακα $F_w^{-1}F_b$ η διαδικασία διαγωνιοποίησης του γίνεται ιδιαίτερα δύσκολη.

3.4 Ο αλγόριθμος EGM

Μία ευρέως γνωστή τεχνική για την αναγνώριση και επαλήθευση προσώπου είναι ο αλγόριθμος ταιριάσματος ελαστικού γράφου (Elastic Graph Matching) ή (EGM). Στον EGM, ο γράφος αναφοράς ενός αντικειμένου δημιουργείται με την τοποθέτηση ενός ορθογώνιου αραιού ελαστικού γράφου στην εικόνα αντικειμένου και τον υπολογισμό μιας απόκρισης κυματιδίων Gabor σε κάθε κόμβο του γράφου. Η διαδικασία ταιριάσματος των γράφων υλοποιείται από μια στοχαστική βελτιστοποίηση μιας συνάρτησης κόστους που λαμβάνει υπόψη και τις παραμορφώσεις του πλέγματος.

Από την εφεύρεσή του, ο EGM έχει μία πολύ σημαντική προσφορά στην έρευνα για την αναγνώριση και επαλήθευση προσώπου.

Στο πρώτο βήμα του αλγορίθμου EGM ένας αραιός γράφος κατάλληλος για την αντιπροσώπευση του προσώπου επιλέγεται. Η περιοχή του προσώπου εικόνας αναλύεται και ένα σύνολο τοπικών περιγραφών εξάγεται σε κάθε κόμβο του γράφου. Η ανάλυση εκτελείται συνήθως με την οικοδόμηση μιας πυραμίδας πληροφοριών χρησιμοποιώντας τεχνικές scale-space. Στον απλό Ε Μ, ένα δισδιάστατο φίλτρο gabor χρησιμοποιείται για την ανάλυση εικόνας . Η παραγωγή των μορφολογικών

διαδικασιών διαστολή-διάβρωσης ή η μορφολογική αποσύνθεση σημάτων σε διάφορες κλίμακες είναι μη γραμμικές παραλλαγές των φίλτρων Gabor για πολυδιάστατη ανάλυση, και τα δύο έχουν χρησιμοποιηθεί επιτυχώς για την ανάλυση της εικόνας προσώπου. Σε κάθε κόμβο λ του γράφου που έχει x^l συντεταγμένες στην εικόνα, ένα jet (διάνυσμα χαρακτηριστικών) j (x) διαμορφώνεται :

$$J(x^l) = [f_1(x^l), ..., f_m(x^l)]^T$$

σχέση 3.4.1

Όπου το $fi(x^l)$ είναι το αποτέλεσμα ενός τοπικού τελεστή που εφαρμόζεται στην εικόνα *f* στην *i* διάσταση και *M* είναι η διάσταση του *jet*. Το επόμενο βήμα του EGM είναι να μεταφέρει και να παραμορφώσει τον γράφο αναφοράς στην εικόνα δοκιμής ούτως ώστε να βρει την απόκριση του γράφου αναφοράς στην εικόνα δοκιμής . Αυτό επιτυγχάνεται ελαχιστοποιώντας μία συνάρτηση κόστους που ανταποκρίνεται τις στις ομοιότητες των jets των κόμβων και στις σχέσεις γειτονιάς κόμβων. Έστω *t* και *r* μία εικόνα δοκιμής και αναφοράς αντίστοιχα (ή γράφος).

$$C_f(j(x_t^l), j(x_r^l)) = ||j(x_r^l) - j(x_t^l)||$$

σχέση 3.4.2

Έστω *V* το σύνολο των ακμών ενός γράφου ορισμένης εικόνας προσώπου. Οι γράφοι που εξετάζονται σε αυτήν την περίπτωση είναι ορθογώνιοι γράφοι οι όποιοι είναι τοπολογικά ισοδύναμοι με ένα ορθογώνιο υποσύνολο του Z (το Z είναι το σύνολο ακέραιων αριθμών). Κατά συνέπεια, όλοι οι κόμβοι, εκτός από τους ακριανούς κόμβους, έχουν ακριβώς τέσσερις συνδεδεμένους κόμβους. Το σχήμα 3.4.1 δείχνει ένα τυπικό ορθογώνιο γράφο αναφοράς που χρησιμοποιείται σε αυτή την περίπτωση. Έστω H(l) η γειτονιά του κόμβου l. Για να ποσοτικοποιήσουμε τις σχέσεις γειτνίασης χρησιμοποιώντας μία μετρική, η τοπική παραμόρφωση είναι :

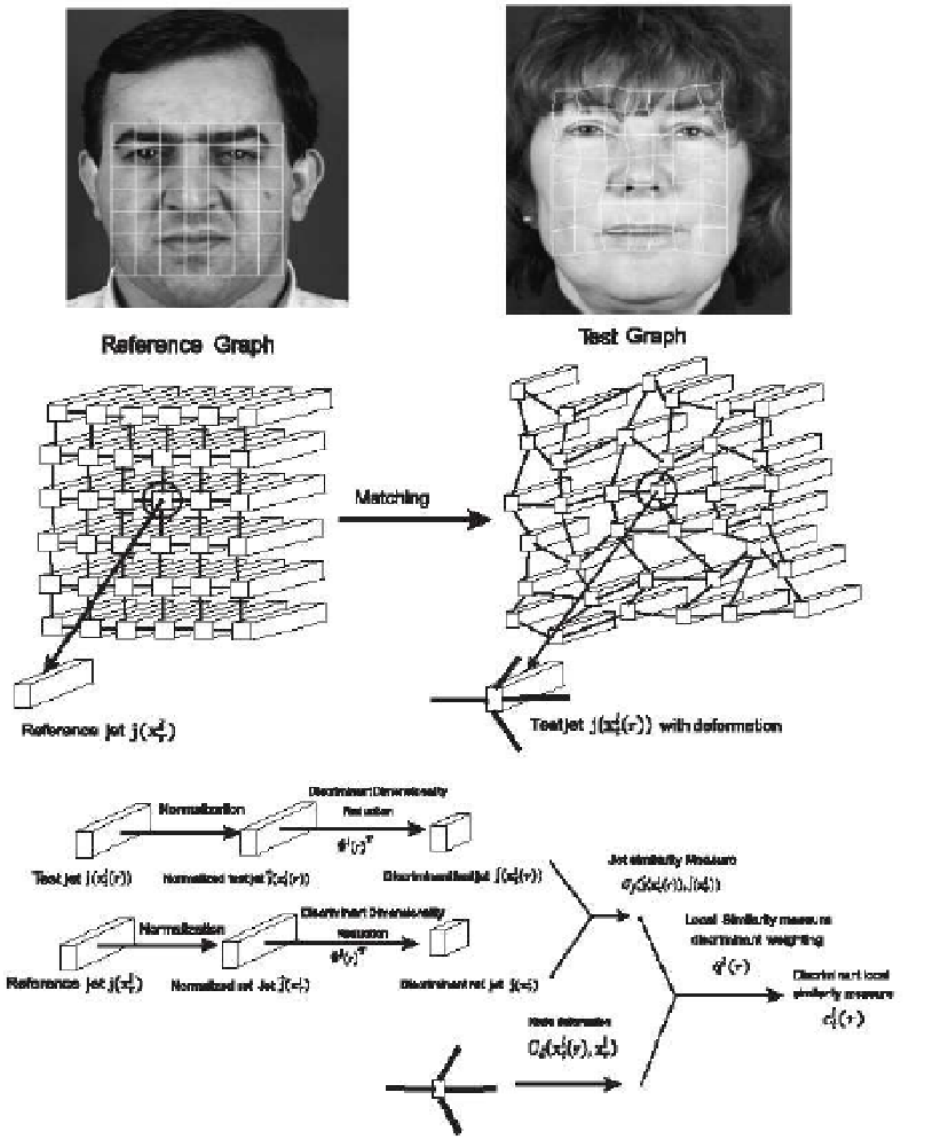
$$C_d(x_t^l, x_r^l) = \sum_{\xi \in H(l)} ||(x_t^l - x_r^l) - (x_t^\xi - x_r^\xi)||$$

σχέση 3.4.3

Σκοπός είναι να βρεθεί ένα σύνολο από ακμές $\{x^l_t(r), l \text{ ανήκει } V\}$ στην εικόνα δοκιμής τέτοιο ώστε να ελαχιστοποιεί την συνάρτηση κόστους:

$$C(x_t^l) = \sum_{l \in \mathcal{V}} \{C_f(j(x_t^l), j(x_r^l)) + C_d(x_t^l, x_r^l)\}$$

σχέση 3.4.4



Σχήμα 3.4.1 Τα βήματα του EGM

Το jet του κόμβου l που έχει παραχθεί μετά από την διαδικασία ταιριάσματος με διαδικασία του γράφου του προσώπου αναφοράς στην εικόνα του προσώπου δοκιμής τ προσδιορίζεται ως $j(x^l_t(r))$. Αυτή η αναφορά γίνεται γιατί διαφορετικοί γράφοι αναφοράς μας δίνουν άλλα τεστ $j(x_l^t(r))$. Κατά συνέπεια, ο l κόμβος του γράφου τ είναι μια συνάρτηση του γράφου αναφοράς r. Ο συμβολισμός $j(x^l_t)$ χρησιμοποιείται μόνο όταν ο l κόμβος είναι σε μια προεπιλεγμένη θέση της εικόνας προσώπου.

Η βελτιστοποίηση της σχέσης 3.4.4 έχει υλοποιηθεί ως προσομοιωμένη ανόπτηση με ποινές που επιβάλλονται από τις παραμορφώσεις των γράφων. Συγκεκριμένα, η σχέση 3.4.4 μπορεί να πάρει την μορφή:

$$D_t(r) = \sum_{l \in \mathcal{V}} \{C_f(j(x_t^l), j(x_r^l))\}$$

σχέση 3.4.5

υπό την προϋπόθεση

$$x_t^l = x_r^l + s + \delta_l, ||\delta_l|| \leq \delta_{max}$$

σχέση 3.4.6

όπου το *s* είναι η ολική μετατόπιση του γράφου και το δ_l υποδηλώνει μία τοπική διαταραχή των κόμβων του γράφου. Η επιλογή των *l* στην σχέση 3.4.4 και του δ_{max} στην 3.4.4 ελέγχουν την πλαστικότητα του γράφου. Προφανώς και οι δύο σχέσεις ορίζουν ένα μέτρο ομοιότητας μεταξύ δύο προσώπων. Μετά την διαδικασία ταιριάσματος η απόσταση $D_t(r)$ χρησιμοποιείται σαν ποσοτικό μέτρο για την ομοιότητα δύο προσώπων .

Εξετάζοντας προσεκτικά τη διαδικασία EGM από την σκοπιά της αναγνώρισης προτύπων, οι ακόλουθες ερωτήσεις προκύπτουν : Όλες οι διαστάσεις του *jet* κατέχουν χρήσιμες πληροφορίες ; Η παραμόρφωση κόμβων κατέχουν αξιοποιήσιμη πληροφορία ; Είναι όλοι οι κόμβοι του γράφου εξίσου σημαντικοί για την επαλήθευση της ταυτότητας μιας εικόνας προσώπου ;

Προκειμένου να απαντήσουμε σε όλες αυτές τις ερωτήσεις, ένα γενικό πλαίσιο που ενισχύει την απόδοση του EGM με έναν εποπτευμένο τρόπο προτείνεται. Λεπτομερέστερα, οι διακρίνουσες τεχνικές χρησιμοποιούνται για την επιλογή των πιο διακρινόντων χαρακτηριστικών τις κάθε κλάσης προσώπου. Το μέτρο ομοιότητας των *jet* συνδυάζεται κατά διακρίνοντα τρόπο με την παραμόρφωση των κόμβων προκειμένου να διαμορφωθεί ένα τοπικό διακρίνων μέτρο ομοιότητας μεταξύ των κόμβων. Η χρήση της παραμόρφωσης κατά έναν διακρίνων τρόπο μπορεί να εξηγηθεί διαισθητικά ως εξής. Ο γράφος του προσώπου έχει τους κόμβους που μπορούν να αντιστοιχούν στα ορόσημα (τα ορόσημα αντιστοιχούν στα σημεία του προσώπου) των οποίων η παραμόρφωση μπορεί να θεωρηθεί άκαμπτη είτε ελαστική για ένα ιδιαίτερο πρόσωπο. Για παράδειγμα, οι κόμβοι που αντιστοιχούν στα σημάδια προσώπου ενός ατόμου που είναι σε κάποια άκαμπτη περιοχή όπως το μέτωπο ή τη μύτη δεν μπορούν να κινηθούν εύκολα, ενώ μερικοί κόμβοι που αντιστοιχούν στα σημεία μέσα στα χείλια μπορούν να κινηθούν πιο ελεύθερα.

Εάν είχαμε εκ των προτέρων διαθέσιμη την πληροφορία για την ελαστικότητα ή ακαμψία κάθε περιοχής του προσώπου, θα μπορούσαμε να την είχαμε ενσωματώσει στην διαδικασία ταιριάσματος του πλέγματος. Εντούτοις, αυτή η πληροφορία είναι συγκεκριμένη για κάθε πρόσωπο και έτσι πρέπει να ανακτηθεί χρησιμοποιώντας μια διαδικασία εκπαίδευσης και να ληφθεί υπ' όψιν όταν σχηματίζουμε το μέτρο τοπικής ομοιότητας μεταξύ των κόμβων.

3.5 Structural Similarity Index Method (SSIM)

3.5.1 Γενικά

Ο δείκτης δομικής ομοιότητας (SSIM) είναι μια μέθοδος για την πρόβλεψη της αντιλαμβανόμενης ποιότητας της ψηφιακής τηλεόρασης και των κινηματογραφικών εικόνων, καθώς και άλλων ειδών ψηφιακών εικόνων και βίντεο. Αναπτύχθηκε για πρώτη φορά στο Εργαστήριο Μηχανικών Εικόνας και Video (LIVE) στο Πανεπιστήμιο του Τέξας στο Όστιν και σε επακόλουθη συνεργασία με το Πανεπιστήμιο της Νέας Υόρκης.

Ο αλγόριθμος SSIM χρησιμοποιείται για τη μέτρηση της ομοιότητας μεταξύ δύο εικόνων. Ο αλγόριθμος λέγεται ότι είναι ένας full reference αλγόριθμος. Με άλλα λόγια, η μέτρηση ή πρόβλεψη της ποιότητας της εικόνας βασίζεται σε μια αρχική ασυμπίεστη ή χωρίς παραμόρφωση εικόνας ως σημείο αναφοράς. Ο SSIM έχει σχεδιαστεί για να βελτιώσει τις παραδοσιακές μεθόδους, όπως η peak signal to noise ratio (PSNR) και του μέσου τετραγωνικού σφάλματος (MSE), οι οποίες έχουν αποδειχθεί ότι είναι ασυμβίβαστες με την ανθρώπινη οπτική αντίληψη.

3.5.2 Αλγόριθμος

Η διαφορά σε σχέση με άλλες τεχνικές όπως η MSE ή η PSNR είναι ότι αυτές οι προσεγγίσεις υπολογίζουν τα absolute errors. Από την άλλη πλευρά, ο SSIM είναι ένα μοντέλο που βασίζεται στην αντίληψη ότι θεωρεί την υποβάθμιση της εικόνας ως μια αλλαγή στην δομική πληροφορία, ενώ ενσωματώνει επίσης σημαντικά φαινόμενα, όπως η φωτεινότητα και η αντίθεση. Δομικές πληροφορίες είναι η ιδέα ότι τα pixels έχουν ισχυρές αλληλεξαρτήσεις ειδικά όταν είναι πολύ κοντά στο χώρο. Αυτές οι εξαρτήσεις μεταφέρουν σημαντικές πληροφορίες σχετικά με τη δομή των αντικειμένων στην οπτική σκηνή. Η φωτεινότητα συγκάλυψης είναι ένα φαινόμενο κατά το οποίο οι στρεβλώσεις της εικόνας (στο πλαίσιο αυτό) τείνουν να είναι λιγότερο ορατές σε φωτεινές περιοχές, ενώ η συγκάλυψη αντίθεσης είναι ένα φαινόμενο κατά το οποίο οι στρεβλώσεις γίνονται λιγότερο ορατές, όπου υπάρχει σημαντική δραστηριότητα ή "υφή" στην εικόνα.

Αν θεωρήσουμε σαν x και y τις 2 συγκρίσιμες εικόνες μεγέθους NxN ως είσοδο, θα χρησιμοποιήσουμε την παρακάτω σχέση για να βρούμε τη συσχέτιση:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

σχέση 3.5.1

Όπου:

- μ_x είναι ο μέσος όρος του x
- μ_y είναι ο μέσος όρος του y
- σ_x^2 είναι ο variance του x
- σ_y^2 είναι ο variance του y
- σ_{xy} είναι ο covariance των x και y
- $c_1=(k_1 L)^2$, $c_2=(k_2 L)^2$ Είναι οι 2 μεταβλητές που σταθεροποιούν τη διαίρεση του αδύναμου παρονομαστή
- L είναι το δυναμικό εύρος της τιμής των pixel.(συνήθως αυτή η τιμή είναι: $(2^{\#bits\ per\ pixel}-1)$)
- σταθερές $k_1=0.01$ και $k_2=0.03$

Η συνάρτηση SSIM είναι συμμετρική, άρα ισχύει:

SSIM(x,y) = SSIM(y,x).

Ο αλγόριθμος SSIM βασίζεται σε 3 παραμέτρους για να κάνει τη σύγκριση 2 εικόνων:

Τη φωτεινότητα – luminance (l)
Την αντίθεση – contrast (c)
Την δομή – structure (s)

Οι μεμονωμένες συναρτήσεις σύγκρισης είναι οι εξής:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}$$

σχέση 3.5.2

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}$$

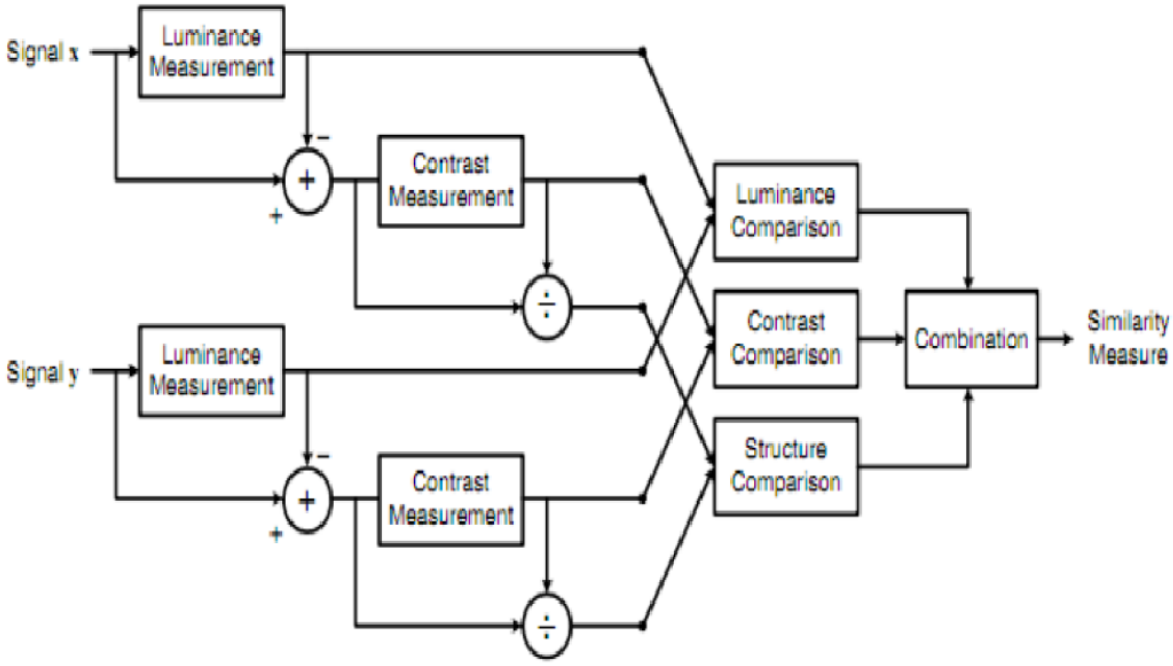
σχέση 3.5.3

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x \sigma_y + c_3}$$

σχέση 3.5.4

Όπου $c_3=c_2/2$

Παρακάτω φαίνεται το σχεδιάγραμμα του SSIM:



Σχήμα 3.5.1

3.5.3 Η συνάρτηση ssim στο matlab:

```
function [mssim, ssim_map] = ssim(img1, img2, K, window, L)

%========================================================================
%SSIM Index, Version 1.0
%Copyright(c) 2003 Zhou Wang
%All Rights Reserved.
%
%This is an implementation of the algorithm for calculating the
%Structural SIMilarity (SSIM) index between two images. Please refer
%to the following paper:
%
```

```
%Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
%quality assessment: From error visibility to structural similarity"
%IEEE Transactios on Image Processing, vol. 13, no. 4, pp.600-612,
%Apr. 2004.
%
%Kindly report any suggestions or corrections to zhouwang@ieee.org
%
%----------------------------------------------------------------------
%
%Input : (1) img1: the first image being compared
%       (2) img2: the second image being compared
%       (3) K: constants in the SSIM index formula (see the above
%           reference). defualt value: K = [0.01 0.03]
%       (4) window: local window for statistics (see the above
%           reference). default widnow is Gaussian given by
%           window = fspecial('gaussian', 11, 1.5);
%       (5) L: dynamic range of the images. default: L = 255
%
%Output: (1) mssim: the mean SSIM index value between 2 images.
%           If one of the images being compared is regarded as
%           perfect quality, then mssim can be considered as the
%           quality measure of the other image.
%           If img1 = img2, then mssim = 1.
%       (2) ssim_map: the SSIM index map of the test image. The map
%           has a smaller size than the input images. The actual size:
%           size(img1) - size(window) + 1.
%
%Default Usage:
%   Given 2 test images img1 and img2, whose dynamic range is 0-255
%
%   [mssim ssim_map] = ssim_index(img1, img2);
%
%Advanced Usage:
%   User defined parameters. For example
%
%   K = [0.05 0.05];
%   window = ones(8);
%   L = 100;
%   [mssim ssim_map] = ssim_index(img1, img2, K, window, L);
%
%See the results:
%
%   mssim                       %Gives the mssim value
```

```
%   imshow(max(0, ssim_map).^4)  %Shows the SSIM index map
%
%========================================================================


if (nargin < 2 | nargin > 5)
   ssim_index = -Inf;
   ssim_map = -Inf;
   return;
end

if (size(img1) ~= size(img2))
   ssim_index = -Inf;
   ssim_map = -Inf;
   return;
end

[M N] = size(img1);

if (nargin == 2)
   if ((M < 11) | (N < 11))   % ?????
         ssim_index = -Inf;
         ssim_map = -Inf;
     return
   end
   window = fspecial('gaussian', 11, 1.5);       % ???1.511*11????
   K(1) = 0.01;                                                % default settings
   K(2) = 0.03;                                                %
   L = 255;                                %
end

if (nargin == 3)
   if ((M < 11) | (N < 11))
         ssim_index = -Inf;
         ssim_map = -Inf;
     return
   end
   window = fspecial('gaussian', 11, 1.5);
   L = 255;
   if (length(K) == 2)
     if (K(1) < 0 | K(2) < 0)
                ssim_index = -Inf;
                ssim_map = -Inf;
```

```
            return;
    end
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end

if (nargin == 4)
  [H W] = size(window);
  if ((H*W) < 4 | (H > M) | (W > N))
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  L = 255;
  if (length(K) == 2)
    if (K(1) < 0 | K(2) < 0)
            ssim_index = -Inf;
            ssim_map = -Inf;
            return;
    end
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end

if (nargin == 5)
  [H W] = size(window);
  if ((H*W) < 4 | (H > M) | (W > N))
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  if (length(K) == 2)
    if (K(1) < 0 | K(2) < 0)
            ssim_index = -Inf;
            ssim_map = -Inf;
            return;
    end
```

```
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end
%%
C1 = (K(1)*L)^2;    % C1Lxy?
C2 = (K(2)*L)^2;    % C2??Cxy?
window = window/sum(sum(window)); %??
img1 = double(img1);
img2 = double(img2);

mu1   = filter2(window, img1, 'valid');  % ????
mu2   = filter2(window, img2, 'valid');  % ????

mu1_sq = mu1.*mu1;     % Ux??
mu2_sq = mu2.*mu2;     % Uy??
mu1_mu2 = mu1.*mu2;    % Ux*Uy?

sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;  % sigmax ??
sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;  % sigmay ??
sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;   % sigmaxy??

if (C1 > 0 & C2 > 0)
  ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq +
C1).*(sigma1_sq + sigma2_sq + C2));
else
  numerator1 = 2*mu1_mu2 + C1;
  numerator2 = 2*sigma12 + C2;
  denominator1 = mu1_sq + mu2_sq + C1;
  denominator2 = sigma1_sq + sigma2_sq + C2;
  ssim_map = ones(size(mu1));
  index = (denominator1.*denominator2 > 0);
  ssim_map(index) =
(numerator1(index).*numerator2(index))./(denominator1(index).*denominator2(in
dex));
  index = (denominator1 ~= 0) & (denominator2 == 0);
  ssim_map(index) = numerator1(index)./denominator1(index);
end

mssim = mean2(ssim_map);
return
```

3.6 Histogram

3.6.1 Γενικά

Το Ιστόγραμμα είναι γραφική απεικόνιση στατιστικών συχνοτήτων περιοχών τιμών ενός μεγέθους. Σχηματίζεται από παρακείμενα ορθογώνια. Η επιφάνεια κάθε ορθογωνίου είναι μέτρο της συχνότητας εμφάνισης της συγκεκριμένης περιοχής τιμών ενώ το ύψος του ισούται με το λόγο της συχνότητας προς το εύρος των τιμών που αντιπροσωπεύει το ορθογώνιο. Πρόκειται για τη συνηθέστερη επιλογή γραφικής παράστασης συνεχών μεταβλητών. Στα συνεχή δεδομένα, οι τιμές της μεταβλητής ομαδοποιούνται και οι ομάδες διατάσσονται στον οριζόντιο άξονα κατ΄αύξουσα σειρά. Στη συνέχεια από κάθε ομάδα υψώνουμε ορθογώνια ,το ύψος των οποίων αντιστοιχεί στη συχνότητα κάθε ομάδας.

Ουσιαστικά ένα ιστόγραμμα είναι γράφημα που δείχνει την κατανομή των δεδομένων.

Τα ιστογράμματα στην επεξεργασία εικόνας χρησιμοποιούνται για να δείξουν την κατανομή των τιμών των pixels σε μια εικόνα.
Το ιστόγραμμα μίας ψηφιακής εικόνας με επίπεδα του γκρι στο διάστημα [0, L-1] είναι μία διακριτή συνάρτηση , όπου είναι το k επίπεδο και είναι το πλήθος των pixels της εικόνας, που έχουν τιμή $h(r_k)=n_k$ επιπέδου γκρι . Συνήθως, κανονικοποιούμε το ιστόγραμμα, διαιρώντας κάθε τιμή με τον συνολικό αριθμό των pixels της εικόνας, έστω n. Τότε, το κανονικοποιημένο ιστόγραμμα δίνεται από την συνάρτηση $p(r_k)=n_k/n$, για k=0,1,…,L-1. Θα μπορούσαμε να πούμε, ότι η p δίνει μία προσέγγιση της πιθανότητας της εμφάνισης ενός γκρι επιπέδου r_k.

Τα ιστογράμματα μπορεί να φανούν πολύ χρήσιμα γιατί μας βοηθούν να εξάγουμε συμπεράσματα για τη μορφή μιας εικόνας. Για παράδειγμα:

- Μία σκούρα εικόνα (dark image)

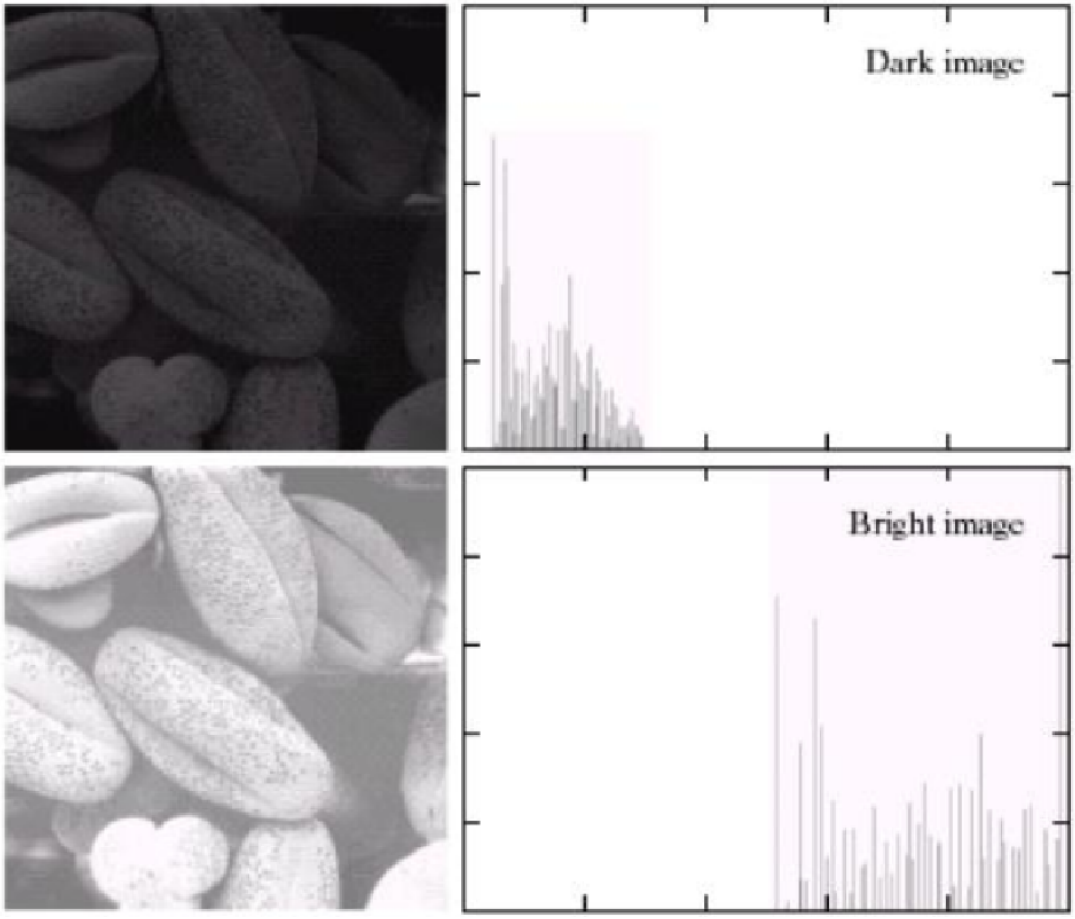
οι τιμές του γκρίζου θα είναι συγκεντρωμένες στα χαμηλότερα επίπεδα.

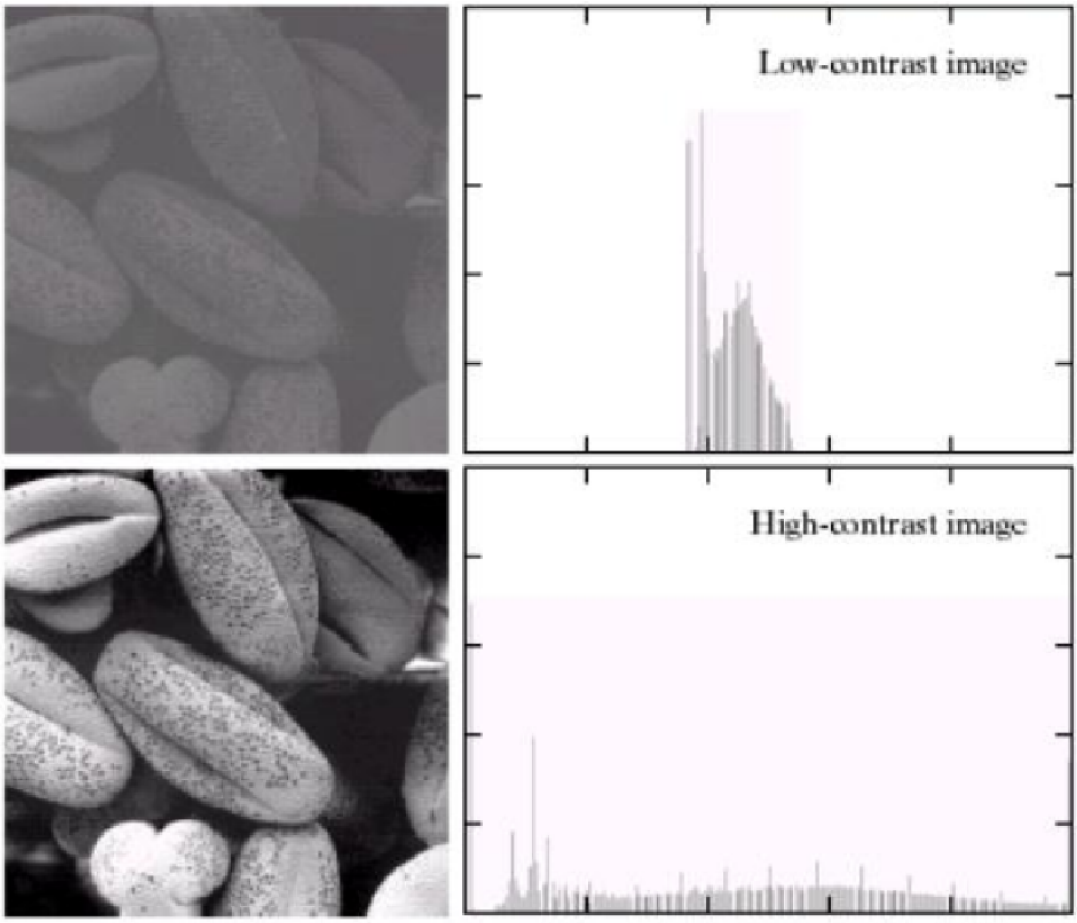
- Σε μία φωτεινή εικόνα αντίθετα οι τιμές του γκρίζου θα είναι συγκεντρωμένες σε υψηλότερα επίπεδα.

- Μία εικόνα με χαμηλό contrast θα έχει για παράδειγμα τις τιμές συγκεντρωμένες στο κέντρο.

Για να τονιστεί η χρησιμότητα του ιστογράμματος στην ψηφιακή επεξεργασία εικόνας παρατηρήστε το επόμενο σχήμα, όπου και απεικονίζεται η ίδια εικόνα σε 4 διαφορετικές μορφές, με τα αντίστοιχα ιστογράμματά τους. Ο οριζόντιος άξονας

κάθε ιστογράμματος αντιστοιχίζεται στα r_k επίπεδα του γκρι , ενώ ο κάθετος άξονας στην $p(r_k)$. Στο πρώτο, π.χ. ιστόγραμμα βλέπουμε ότι τα περισσότερα pixels συγκεντρώνονται στις μικρές τιμές των γκρι επιπέδων, και άρα η εικόνα είναι αρκετά 'σκούρα', σε αντίθεση με το δεύτερο ιστόγραμμα. Στην τρίτη περίπτωση, παρατηρούμε ότι οι τιμές των pixels περιορίζονται σε ένα μικρό πεδίο των επιπέδων γκρι, και η εικόνα είναι χαμηλής αντίθεσης.





εικόνα 3.6.1

Θα μπορούσαμε να πούμε ότι από τις παραπάνω 4 περιπτώσεις, η ιδανική είναι η τελευταία, όπου το ιστόγραμμα «απλώνεται» σε όλο το εύρος των διαθέσιμων γκρι επιπέδων και η εικόνα είναι κατανοητή από το ανθρώπινο μάτι.

3.6.2 Συνάρτηση μέσω matlab

Στο matlab αυτό το καταφέρνουμε με τη συνάρτηση imhist( ).

```
function [yout,x] = imhist(varargin)
%IMHIST Display histogram of image data.
%   IMHIST(I) displays a histogram for the intensity image I whose number of
%   bins are specified by the image type.  If I is a grayscale image, IMHIST
%   uses 256 bins as a default value. If I is a binary image, IMHIST uses
%   only 2 bins.
%
%   IMHIST(I,N) displays a histogram with N bins for the intensity image I
%   above a grayscale colorbar of length N.  If I is a binary image then N
%   can only be 2.
%
%   IMHIST(X,MAP) displays a histogram for the indexed image X. This
%   histogram shows the distribution of pixel values above a colorbar of the
%   colormap MAP. The colormap must be at least as long as the largest index
%   in X. The histogram has one bin for each entry in the colormap.
```

```
%
%   [COUNTS,X] = imhist(...) returns the histogram counts in COUNTS and the
%   bin locations in X so that stem(X,COUNTS) shows the histogram. For
%   indexed images, it returns the histogram counts for each colormap entry;
%   the length of COUNTS is the same as the length of the colormap.
%
%   Class Support
%   -------------
%   An input intensity image can be uint8, int8, uint16, int16, uint32,
%   int32, single, double, or logical. An input indexed image can be uint8,
%   uint16, single, double, or logical.
%
%   Note
%   ----
%   For intensity images, the N bins of the histogram are each half-open
%   intervals of width A/(N-1).
%
%   For uint8, uint16, and uint32 intensity images, the p-th bin is the
%   half-open interval:
%
%       A*(p-1.5)/(N-1)  <= x  <  A*(p-0.5)/(N-1)
%
%   For int8, int16, and int32 intensity images, the p-th bin is the
%   half-open interval:
%
%       A*(p-1.5)/(N-1) - 32768  <= x  <  A*(p-0.5)/(N-1) - 32768
%
%   The intensity value is represented by "x". The scale factor A depends
%   on the image class.  A is 1 if the intensity image is double or single;
%   A is 255 if the intensity image is uint8 or int8; A is 65535 if the
%   intensity image is uint16 or int16; A is 4294967295 if the intensity
%   image is uint32 or int32.
%
%   Example
%   -------
%       I = imread('pout.tif');
%       imhist(I)
%
%   See also HISTEQ, HIST, IMHISTMATCH.

%   Copyright 1992-2012 The MathWorks, Inc.
%   $Revision: 5.24.4.19 $ $Date: 2012/03/01 02:21:59 $

[a, n, isScaled, top, map] = parse_inputs(varargin{:});

if islogical(a)
   if (n ~= 2)
      error(message('images:imhist:invalidParameterForLogical'))
   end
   y(2) = sum(a(:));
   y(1) = numel(a) - y(2);
   y = y';
```

```
elseif isa(a,'int8')
   y = imhistc(int8touint8(a), n, isScaled, top); % Call MEX file to do work.
elseif isa(a, 'int16')
   y = imhistc(int16touint16(a), n, isScaled, top); % Call MEX file to do work.
elseif isa(a, 'int32')
   y = imhistc(int32touint32(a), n, isScaled, top); % Call MEX file to do work.
else
   y = imhistc(a, n, isScaled, top); % Call MEX file to do work.
end

range = getrangefromclass(a);

if ~isScaled
   if isfloat(a)
      x = 1:n;
   else
      x = 0:n-1;
   end
elseif islogical(a)
   x = range';
else
   % integer or float
   x = linspace(range(1), range(2), n)';
end

if (nargout == 0)
   plot_result(x, y, map, isScaled, class(a), range);
else
   yout = y;
end


%%%
%%% Function plot_result
%%%
function plot_result(x, y, cm, isScaled, classin, range)

n = length(x);
stem(x,y, 'Marker', 'none')
hist_axes = gca;

h_fig = ancestor(hist_axes,'figure');

% Get x/y limits of axes using axis
limits = axis(hist_axes);
if n ~= 1
  limits(1) = min(x);
else
  limits(1) = 0;
end
limits(2) = max(x);
var = sqrt(y'*y/length(y));
```

```
limits(4) = 2.5*var;
axis(hist_axes,limits);


% Cache the original axes position so that axes can be repositioned to
% occupy the space used by the colorstripe if nextplot clears the histogram
% axes.
original_axes_pos = get(hist_axes,'Position');

% In GUIDE, default axes units are characters. In order for axes repositiong
% to behave properly, units need to be normalized.
hist_axes_units_old = get(hist_axes,'units');
set(hist_axes,'Units','Normalized');
% Get axis position and make room for color stripe.
pos = get(hist_axes,'pos');
stripe = 0.075;
set(hist_axes,'pos',[pos(1) pos(2)+stripe*pos(4) pos(3) (1-stripe)*pos(4)])
set(hist_axes,'Units',hist_axes_units_old);

set(hist_axes,'xticklabel','')

% Create axis for stripe
stripe_axes = axes('Parent',get(hist_axes,'Parent'),...
                'Position', [pos(1) pos(2) pos(3) stripe*pos(4)]);

limits = axis(stripe_axes);

% Create color stripe
if isScaled,
   binInterval = 1/n;
   xdata = [binInterval/2 1-(binInterval/2)];
   limits(1:2) = range;
   switch classin
    case {'uint8', 'uint16', 'uint32'}
      xdata = range(2)*xdata;
      C = (1:n)/n;
    case {'int8','int16', 'int32'}
      xdata = (range(2)-range(1))* xdata + range(1);
      C = (1:n)/n;
    case {'double','single'}
      C = (1:n)/n;
    case 'logical'
      C = [0 1];
    otherwise
      error(message('images:imhist:internalError'))
   end

   % image(X,Y,C) where C is the RGB color you specify.
   image(xdata,[0 1],repmat(C, [1 1 3]),'Parent',stripe_axes);
else
   if length(cm)<=256
      image([1 n],[0 1],1:n,'Parent',stripe_axes);
```

```
        set(h_fig,'Colormap',cm);
        limits(1) = 0.5;
        limits(2) = n + 0.5;
    else
        image([1 n],[0 1],permute(cm, [3 1 2]),'Parent',stripe_axes);
        limits(1) = 0.5;
        limits(2) = n + 0.5;
    end
end

set(stripe_axes,'yticklabel','')
axis(stripe_axes,limits);

% Put a border around the stripe.
line(limits([1 2 2 1 1]),limits([3 3 4 4 3]),...
      'LineStyle','-',...
      'Parent',stripe_axes,...
      'Color',get(stripe_axes,'XColor'));

% Special code for a binary image
if strcmp(classin,'logical')
    % make sure that the stripe's X axis has 0 and 1 as tick marks.
    set(stripe_axes,'XTick',[0 1]);

    % remove unnecessary tick marks from axis showing the histogram
    set(hist_axes,'XTick',0);

    % make the histogram lines thicker
    h = get(hist_axes,'children');
    obj = findobj(h,'flat','Color','b');
    lineWidth = 10;
    set(obj,'LineWidth',lineWidth);
end

set(h_fig,'CurrentAxes',hist_axes);

% Tag for testing.
set(stripe_axes,'tag','colorstripe');

wireHistogramAxesListeners(hist_axes,stripe_axes,original_axes_pos);

% Link the XLim of histogram and color stripe axes together.
% In calls to imhist in a tight loop, the histogram and colorstripe axes
% are destroyed and recreated repetitively. Use linkprop rather than
% linkaxes to link xlimits together to solve deletion timing problems.
h_link = linkprop([hist_axes,stripe_axes],'XLim');
setappdata(stripe_axes,'linkColorStripe',h_link);

%%%
%%% Function wireHistogramAxesListeners
%%%
function wireHistogramAxesListeners(hist_axes,stripe_axes,original_axes_pos)
```

```
% If the histogram axes is deleted, delete the color stripe associated with
% the histogram axes.
cb_fun = @(obj,evt) removeColorStripeAxes(stripe_axes);
lis.histogramAxesDeletedListener = iptui.iptaddlistener(hist_axes,...
    'ObjectBeingDestroyed',cb_fun);

% This is a dummy hg object used to listen for when the histogram axes is cleared.
deleteProxy = text('Parent',hist_axes,...
    'Visible','Off', ...
    'Tag','axes cleared proxy',...
    'HandleVisibility','off');

% deleteProxy is an invisible text object that is parented to the histogram
% axes.  If the ObjectBeingDestroyed listener fires, the histogram axes has
% been cleared. This listener is triggered by newplot when newplot clears
% the current axes to make way for new hg objects being drawn. This
% listener does NOT fire as a result of the parent axes being deleted.
prox_del_cb = @(obj,evt) histogramAxesCleared(obj,stripe_axes,original_axes_pos);
lis.proxydeleted = iptui.iptaddlistener(deleteProxy,...
    'ObjectBeingDestroyed',prox_del_cb);

setappdata(stripe_axes,'ColorStripeListeners',lis);


%%%
%%% Function removeColorStripeAxes
%%%
function removeColorStripeAxes(stripe_axes)

if ishghandle(stripe_axes)
    delete(stripe_axes);
end


%%%
%%% Function histogramAxesCleared
%%%
function histogramAxesCleared(hDeleteProxy,stripe_axes,original_axes_pos)

removeColorStripeAxes(stripe_axes);

h_hist_ax = get(hDeleteProxy,'parent');
set(h_hist_ax,'Position',original_axes_pos);


%%%
%%% Function parse_inputs
%%%
function [a, n, isScaled, top, map] = parse_inputs(varargin)

narginchk(1,2);
```

```
a = varargin{1};
validateattributes(a, {'double','uint8','int8','logical','uint16','int16','single','uint32', 'int32'}, ...
            {'2d','nonsparse'}, mfilename, ['I or ' 'X'], 1);
n = 256;

switch (class(a))
case {'double', 'single'}
   isScaled = 1;
   top = 1;
   map = [];

case {'uint8', 'int8'}
   isScaled = 1;
   top = 255;
   map = [];

case 'logical'
   n = 2;
   isScaled = 1;
   top = 1;
   map = [];

case {'int16', 'uint16'}
   isScaled = 1;
   top = 65535;
   map = [];

case {'int32', 'uint32'}
   isScaled = 1;
   top = double(intmax('uint32'));
   map = [];

otherwise
   % shouldn't happen.
end

if (nargin ==2)
   if (numel(varargin{2}) == 1)
      % IMHIST(I, N)
      n = varargin{2};
      validateattributes(n, {'numeric'}, {'real','positive','integer'}, mfilename, ...
                  'N', 2);

   elseif (size(varargin{2},2) == 3)
     if isa(a,'int16')
      error(message('images:imhist:invalidIndexedImage'))
     end

     % IMHIST(X,MAP) or invalid second argument
     n = size(varargin{2},1);
     isScaled = 0;
     top = n;
```

```
    map = varargin{2};

  else
    error(message('images:imhist:invalidSecondArgument'))
  end
end
```

Όταν εφαρμόσουμε το ιστόγραμμα στις 2 εικόνες που έχουμε προς σύγκριση έπειτα θα πρέπει να συγκρίνουμε αυτά τα 2 ιστογράμματα. Αυτό το επιτυγχάνουμε εφαρμόζοντας ευκλείδεια απόσταση στα 2 ιστογράμματα.

Η έννοια της απόστασης είναι θεμελιώδης στην πολυμεταβλητή ανάλυση. Σκοπός της απόστασης είναι να μετρήσει «πόσο απέχουν» δύο παρατηρήσεις. Στη διαδικασία αναγνώρισης που εφαρμόζεται στα διάφορα συστήματα αναγνώρισης προσώπου η έννοια της απόστασης λαμβάνει την σημασία της ομοιότητας των προσώπων. Οι αλγόριθμοι αναγνώρισης προσώπου παράγουν διανύσματα χαρακτηριστικών τόσο για τις εικόνες που ελέγχουν όσο και για τις εικόνες που σχηματίζουν τη βάση εκπαίδευσης.

3.6.3 Ευκλείδεια απόσταση

3.6.3.1 Ορισμός:

Η ευκλείδεια απόσταση ή μετρική είναι μία συνάρτηση: $d : R^n \times R^n \rightarrow R$ που αντιστοιχεί σε δύο διανύσματα x , y του n-διάστατου διανυσματικού χώρου R^n

$$d(\mathbf{x},\mathbf{y}) = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \cdots + (y_n - x_n)^2} = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}.$$

σχέση 3.6.3.1.

όπου:

$$\mathbf{x} = \{x_1, \ldots, x_n\}, \mathbf{y} = \{y_1, \ldots, y_n\}$$

Η συνάρτηση μετράει τη "συνήθη"(Ευκλείδεια) απόσταση μεταξύ δύο σημείων στον επίπεδο , n−διάστατο χώρο κάνοντας επανειλημμένη χρήση του Πυθαγόρειου θεωρήματος.

3.6.3.2 Συνάρτηση μέσω matlab

Στο matlab η Ευκλείδεια απόσταση δίδεται από τη συνάρτηση pdist2( )

```
function [D,I] = pdist2(X,Y,dist,varargin)
%PDIST2 Pairwise distance between two sets of observations.
%   D = PDIST2(X,Y) returns a matrix D containing the Euclidean distances
%   between each pair of observations in the MX-by-N data matrix X and
%   MY-by-N data matrix Y. Rows of X and Y correspond to observations,
%   and columns correspond to variables. D is an MX-by-MY matrix, with the
%   (I,J) entry equal to distance between observation I in X and
%   observation J in Y.
%
%   D = PDIST2(X,Y,DISTANCE) computes D using DISTANCE.  Choices are:
%
%     'euclidean'   - Euclidean distance (default)
%     'seuclidean'  - Standardized Euclidean distance. Each coordinate
%                     difference between rows in X and Y is scaled by
%                     dividing by the corresponding element of the
%                     standard deviation computed from X, S=NANSTD(X).
%                     To specify another value for S, use
%                     D = PDIST2(X,Y,'seuclidean',S).
%     'cityblock'   - City Block distance
%     'minkowski'   - Minkowski distance. The default exponent is 2. To
%                     specify a different exponent, use
%                     D = PDIST2(X,Y,'minkowski',P), where the
%                     exponent P is a scalar positive value.
%     'chebychev'   - Chebychev distance (maximum coordinate difference)
%     'mahalanobis' - Mahalanobis distance, using the sample covariance
%                     of X as computed by NANCOV.  To compute the
%                     distance with a different covariance, use
%                     D = PDIST2(X,Y,'mahalanobis',C), where the matrix C
%                     is symmetric and positive definite.
%     'cosine'      - One minus the cosine of the included angle
%                     between observations (treated as vectors)
%     'correlation' - One minus the sample linear correlation between
%                     observations (treated as sequences of values).
%     'spearman'    - One minus the sample Spearman's rank correlation
```

```
%                   between observations (treated as sequences of
%                   values)
%     'hamming'     - Hamming distance, percentage of coordinates
%                   that differ
%     'jaccard'     - One minus the Jaccard coefficient, the
%                   percentage of nonzero coordinates that differ
%     function      - A distance function specified using @, for example
%                   @DISTFUN
%
%   A distance function must be of the form
%
%       function D2 = DISTFUN(ZI,ZJ),
%
%   taking as arguments a 1-by-N vector ZI containing a single observation
%   from X or Y, an M2-by-N matrix ZJ containing multiple observations from
%   X or Y, and returning an M2-by-1 vector of distances D2, whose Jth
%   element is the distance between the observations ZI and ZJ(J,:).
%
%   For built-in distance metrics, the distance between observation I in X
%   and observation J in Y will be NaN if observation I in X or observation
%   J in Y contains NaNs.
%
%   D = PDIST2(X,Y,DISTANCE,'Smallest',K) returns a K-by-MY matrix D
%   containing the K smallest pairwise distances to observations in X for
%   each observation in Y. PDIST2 sorts the distances in each column of D
%   in ascending order. D = PDIST2(X,Y,DISTANCE, 'Largest',K) returns the K
%   largest pairwise distances sorted in descending order. If K is greater
%   than MX, PDIST2 returns an MX-by-MY distance matrix. For each
%   observation in Y, PDIST2 finds the K smallest or largest distances by
%   computing and comparing the distance values to all the observations in
%   X.
%
%   [D,I] = PDIST2(X,Y,DISTANCE,'Smallest',K) returns a K-by-MY matrix I
%   containing indices of the observations in X corresponding to the K
%   smallest pairwise distances in D. [D,I] = PDIST2(X,Y,DISTANCE,
%   'Largest',K) returns indices corresponding to the K largest pairwise
%   distances.
%
%   Example:
%     % Compute the ordinary Euclidean distance
%     X = randn(100, 5);
%     Y = randn(25, 5);
%     D = pdist2(X,Y,'euclidean');         % euclidean distance
%
%     % Compute the Euclidean distance with each coordinate difference
%     % scaled by the standard deviation
%     Dstd = pdist2(X,Y,'seuclidean');
%
%     % Use a function handle to compute a distance that weights each
%     % coordinate contribution differently.
%     Wgts = [.1 .3 .3 .2 .1];            % coordinate weights
%     weuc = @(XI,XJ,W)(sqrt(bsxfun(@minus,XI,XJ).^2 * W'));
```

```
%     Dwgt = pdist2(X,Y, @(Xi,Xj) weuc(Xi,Xj,Wgts));
%
%   See also PDIST, KNNSEARCH, CREATENS, KDTreeSearcher,
%            ExhaustiveSearcher.

%   An example of distance for data with missing elements:
%
%     X = randn(100, 5);     % some random points
%     Y = randn(25, 5);      % some more random points
%     X(unidrnd(prod(size(X)),1,20)) = NaN; % scatter in some NaNs
%     Y(unidrnd(prod(size(Y)),1,5)) = NaN; % scatter in some NaNs
%     D = pdist2(X, Y, @naneucdist);
%
%     function D = naneucdist(XI, YJ) % euclidean distance, ignoring NaNs
%     [m,p] = size(YJ);
%     sqdxy = bsxfun(@minus,XI,YJ) .^ 2;
%     pstar = sum(~isnan(sqdxy),2); % correction for missing coordinates
%     pstar(pstar == 0) = NaN;
%     D = sqrt(nansum(sqdxy,2) .* p ./ pstar);
%
%
%   For a large number of observations, it is sometimes faster to compute
%   the distances by looping over coordinates of the data (though the code
%   is more complicated):
%
%     function D = nanhamdist(XI, YJ) % hamming distance, ignoring NaNs
%     [m,p] = size(YJ);
%     nesum = zeros(m,1);
%     pstar = zeros(m,1);
%     for q = 1:p
%        notnan = ~(isnan((XI(q)) | isnan(YJ(:,q)));
%        nesum = nesum + (XI(q) ~= YJ(:,q)) & notnan;
%        pstar = pstar + notnan;
%     end
%     D = nesum ./ pstar;

%   Copyright 2009-2011 The MathWorks, Inc.


if nargin < 2
   error(message('stats:pdist2:TooFewInputs'));
end

[nx,p] = size(X);
[ny,py] = size(Y);
if py ~= p
   error(message('stats:pdist2:SizeMismatch'));
end

additionalArg = [];

if nargin < 3
```

```
    dist = 'euc';
else %distance is provided
    if ischar(dist)
        methods = {'euclidean'; 'seuclidean'; 'cityblock'; 'chebychev'; ...
            'mahalanobis'; 'minkowski'; 'cosine'; 'correlation'; ...
            'spearman'; 'hamming'; 'jaccard'};
        i = find(strncmpi(dist,methods,length(dist)));
        if length(i) > 1
            error(message('stats:pdist2:AmbiguousDistance', dist));
        elseif isempty(i)
            error(message('stats:pdist2:UnrecognizedDistance', dist));
        else
            dist = methods{i}(1:3);

            if ~isempty(varargin)
                arg = varargin{1};

                % Get the additional distance argument from the inputs
                if isnumeric(arg)
                    switch dist
                        case {'seu' 'mah' 'min'}
                            additionalArg = arg;
                            varargin = varargin(2:end);
                    end
                end
            end
        end
    elseif isa(dist, 'function_handle')
        distfun = dist;
        dist = 'usr';
    else
        error(message('stats:pdist2:BadDistance'));
    end
end

pnames = {'smallest' 'largest' 'radius'};
dflts =  {       []       []  []};
[smallest,largest,radius] = internal.stats.parseArgs(pnames, dflts, varargin{:});

smallestLargestFlag = [];
if sum([~isempty(largest) ~isempty(smallest) ~isempty(radius)]) > 1
    error(message('stats:pdist2:SmallestAndLargest'));
end
if ~isempty(smallest)
    if ~(isscalar(smallest) && isnumeric(smallest) && smallest >= 1 && round(smallest) == smallest)
        error(message('stats:pdist2:BadSmallest'));
    end
    smallestLargestFlag = min(smallest,nx);
elseif ~isempty(largest)
    if ~(isscalar(largest) && isnumeric(largest) && largest >= 1 && round(largest) == largest)
        error(message('stats:pdist2:BadLargest'));
    end
```

```
    smallestLargestFlag = -min(largest,nx);
elseif ~isempty(radius)
    if ~(isscalar(radius) && isnumeric(radius) && radius >= 0 )
        error(message('stats:pdist2:BadRadius'));
    end
elseif nargout > 1
    error(message('stats:pdist2:TooManyOutputs'));
end

% For a built-in distance, integer/logical/char/anything data will be
% converted to float. Complex floating point data can't be handled by
% a built-in distance function.
%if ~strcmp(dist,'usr')

try
    outClass = superiorfloat(X,Y);
catch
    if isfloat(X)
        outClass = class(X);
    elseif isfloat(Y)
        outClass = class(Y);
    else
        outClass = 'double';
    end
end

if ~strcmp(dist,'usr')
    if ~strcmp(class(X),outClass) || ~strcmp(class(Y),outClass)
        warning(message('stats:pdist2:DataConversion', outClass));
    end
    X = cast(X,outClass);
    Y = cast(Y,outClass);
    if  ~isreal(X) || ~isreal(Y)
        error(message('stats:pdist2:ComplexData'));
    end
end

% Degenerate case, just return an empty of the proper size.
if (nx == 0) || (ny == 0)
    if ~isempty(radius)
        D = repmat({zeros(1,0, outClass)},1,ny);
        I = repmat({zeros(1,0, outClass)},1,ny);
    else
        if ~isempty(smallestLargestFlag)
            nD = abs(smallestLargestFlag);
        else
            nD = nx;
        end
        D = zeros(nD,ny,outClass); % X and Y were single/double, or cast to double
        I = zeros(nD,ny,outClass);
    end
    return;
```

```
end

switch dist
   case 'seu' % Standardized Euclidean weights by coordinate variance
      if isempty(additionalArg)
         additionalArg =  nanvar(X,[],1);
         if any(additionalArg == 0)
            warning(message('stats:pdist2:ConstantColumns'));
         end
         additionalArg = 1./ additionalArg;
      else
         if ~(isvector(additionalArg) && length(additionalArg) == p...
               && all(additionalArg >= 0))
            error(message('stats:pdist2:InvalidWeights'));
         end
         if any(additionalArg == 0)
            warning(message('stats:pdist2:ZeroInverseWeights'));
         end
         additionalArg = 1./ (additionalArg .^2);
      end

   case 'mah' % Mahalanobis
      if isempty(additionalArg)
         if nx == 1
            error(message('stats:pdist2:tooFewXRowsForMah'));
         end
         additionalArg = nancov(X);
         [T,flag] = chol(additionalArg);
      else %provide the covariance for mahalanobis
         if ~isequal(size(additionalArg),[p,p])
            error(message('stats:pdist2:InvalidCov'));
         end
         %cholcov will check whether the covariance is symmetric
         [T,flag] = cholcov(additionalArg,0);
      end

      if flag ~= 0
         error(message('stats:pdist2:InvalidCov'));
      end

      if ~issparse(X) && ~issparse(Y)
         additionalArg = T \ eye(p); %inv(T)
      end

   case 'min' % Minkowski
      if isempty(additionalArg)
         additionalArg = 2;
      elseif ~(isscalar(additionalArg) && additionalArg > 0)
         error(message('stats:pdist2:BadMinExp'));
      end
   case 'cos' % Cosine
      [X,Y,flag] = normalizeXY(X,Y);
```

```
        if flag
            warning(message('stats:pdist2:ZeroPoints'));
        end

    case 'cor' % Correlation
        X = bsxfun(@minus,X,mean(X,2));
        Y = bsxfun(@minus,Y,mean(Y,2));
        [X,Y,flag] = normalizeXY(X,Y);
        if flag
            warning(message('stats:pdist2:ConstantPoints'));
        end

    case 'spe'
        X = tiedrank(X')'; % treat rows as a series
        Y = tiedrank(Y')';
        X = X - (p+1)/2; % subtract off the (constant) mean
        Y = Y - (p+1)/2;
        [X,Y,flag] = normalizeXY(X,Y);
        if flag
            warning(message('stats:pdist2:TiedPoints'));
        end

    otherwise

end

% Note that if the above switch statement is modified to include the
% 'che', 'euc', or 'cit' distances, that code may need to be repeated
% in the corresponding block below.
if strcmp(dist,'min') % Minkowski distance
    if isinf(additionalArg) %the exponent is inf
        dist = 'che';
        additionalArg = [];
    elseif additionalArg == 2 %the exponent is 2
        dist = 'euc';
        additionalArg = [];
    elseif additionalArg == 1 %the exponent is 1
        dist = 'cit';
        additionalArg = [];
    end
end

% Call a mex file to compute distances for the build-in distance measures
% on non-sparse real float (double or single) data.
if ~strcmp(dist,'usr') && (~issparse(X) && ~issparse(Y))
    additionalArg = cast(additionalArg,outClass);

    if nargout < 2
        D = pdist2mex(X',Y',dist,additionalArg,smallestLargestFlag,radius);
    else
        [D,I] = pdist2mex(X',Y',dist,additionalArg,smallestLargestFlag,radius);
    end
```

```
    % The following MATLAB code implements the same distance calculations as
    % the mex file. It assumes X and Y are real single or double.  It is
    % currently only called for sparse inputs, but it may also be useful as a
    % template for customization.
elseif ~strcmp(dist,'usr')
    if any(strcmp(dist, {'ham' 'jac' 'che'}))
        xnans = any(isnan(X),2);
        ynans = any(isnan(Y),2);
    end

    if ~isempty(radius)
        D = cell(1,ny);
        I = cell(1,ny);
    elseif isempty(smallestLargestFlag)
        D = zeros(nx,ny,outClass);
    else
        D = zeros(abs(smallestLargestFlag),ny,outClass);
        I = zeros(abs(smallestLargestFlag),ny,outClass);

    end

    switch dist
        case 'euc'    % Euclidean
            for i = 1:ny
                dsq = zeros(nx,1,outClass);
                for q = 1:p
                    dsq = dsq + (X(:,q) - Y(i,q)).^2;
                end
                dsq = sqrt(dsq);
                if ~isempty(radius)
                    [D{i},I{i}] = radiusSort(dsq,radius);
                elseif isempty(smallestLargestFlag)
                    D(:,i) = dsq;
                else
                    [D(:,i),I(:,i)] = partialSort(dsq,smallestLargestFlag);
                end
            end

        case 'seu'    % Standardized Euclidean
            wgts = additionalArg;
            for i = 1:ny
                dsq = zeros(nx,1,outClass);
                for q = 1:p
                    dsq = dsq + wgts(q) .* (X(:,q) - Y(i,q)).^2;
                end
                dsq = sqrt(dsq);
                if ~isempty(radius)
                    [D{i},I{i}] = radiusSort(dsq,radius);
                elseif isempty(smallestLargestFlag)
                    D(:,i) = dsq;
                else
```

```
            [D(:,i),I(:,i)] = partialSort(dsq,smallestLargestFlag);
        end
    end

case 'cit'    % City Block
    for i = 1:ny
        dsq = zeros(nx,1,outClass);
        for q = 1:p
            dsq = dsq + abs(X(:,q) - Y(i,q));
        end

        if ~isempty(radius)
            [D{i},I{i}] = radiusSort(dsq,radius);
        elseif isempty(smallestLargestFlag)
            D(:,i) = dsq;
        else
            [D(:,i),I(:,i)] = partialSort(dsq,smallestLargestFlag);
        end
    end

case 'mah'    % Mahalanobis

    for i = 1:ny
        del = bsxfun(@minus,X,Y(i,:));
        dsq = sum((del/T) .^ 2, 2);
        dsq = sqrt(dsq);
        if ~isempty(radius)
            [D{i},I{i}] = radiusSort(dsq,radius);
        elseif isempty(smallestLargestFlag)
            D(:,i) = dsq;
        else
            [D(:,i),I(:,i)] = partialSort(dsq,smallestLargestFlag);
        end
    end

case 'min'    % Minkowski
    expon = additionalArg;
    for i = 1:ny
        dpow = zeros(nx,1,outClass);
        for q = 1:p
            dpow = dpow + abs(X(:,q) - Y(i,q)).^expon;
        end
        dpow = dpow .^ (1./expon);
        if ~isempty(radius)
            [D{i},I{i}] = radiusSort(dpow,radius);
        elseif isempty(smallestLargestFlag)
            D(:,i) = dpow;
        else
            [D(:,i),I(:,i)] = partialSort(dpow,smallestLargestFlag);
        end

    end
```

```
case {'cos' 'cor' 'spe'}   % Cosine, Correlation, Rank Correlation
   % This assumes that data have been appropriately preprocessed
   for i = 1:ny
      d = zeros(nx,1,outClass);
      for q = 1:p
         d = d + (X(:,q).*Y(i,q));
      end
      d(d>1) = 1; % protect against round-off, don't overwrite NaNs
      d = 1 - d;
      if ~isempty(radius)
         [D{i},I{i}] = radiusSort(d,radius);
      elseif isempty(smallestLargestFlag)
         D(:,i) = d;
      else
         [D(:,i),I(:,i)] = partialSort(d,smallestLargestFlag);
      end

   end
case 'ham'    % Hamming
   for i = 1:ny
      nesum = zeros(nx,1,outClass);
      for q = 1:p
         nesum = nesum + (X(:,q) ~= Y(i,q));
      end
      nesum(xnans|ynans(i)) = NaN;
      nesum = (nesum ./ p);
      if ~isempty(radius)
         [D{i},I{i}] = radiusSort(nesum,radius);
      elseif isempty(smallestLargestFlag)
         D(:,i) = nesum;
      else
         [D(:,i),I(:,i)] = partialSort(nesum,smallestLargestFlag);
      end

   end
case 'jac'    % Jaccard
   for i = 1:ny
      nzsum = zeros(nx,1,outClass);
      nesum = zeros(nx,1,outClass);
      for q = 1:p
         nz = (X(:,q) ~= 0 | Y(i,q) ~= 0);
         ne = (X(:,q) ~= Y(i,q));
         nzsum = nzsum + nz;
         nesum = nesum + (nz & ne);
      end
      nesum(xnans | ynans(i)) = NaN;
      d = (nesum ./ nzsum);
      if ~isempty(radius)
         [D{i},I{i}] = radiusSort(d,radius);
      elseif isempty(smallestLargestFlag)
         D(:,i) = d;
```

```
            else
               [D(:,i),I(:,i)] = partialSort(d,smallestLargestFlag);
            end

         end
      case 'che'    % Chebychev
         for i = 1:ny
            dmax = zeros(nx,1,outClass);
            for q = 1:p
               dmax = max(dmax, abs(X(:,q) - Y(i,q)));
            end
            dmax(xnans | ynans(i)) = NaN;
            if ~isempty(radius)
               [D{i},I{i}] = radiusSort(dmax,radius);
            elseif isempty(smallestLargestFlag)
               D(:,i) =  dmax;
            else
               [D(:,i),I(:,i)] = partialSort(dmax,smallestLargestFlag);
            end
         end
   end

   % Compute distances for a caller-defined distance function.
else % if strcmp(dist,'usr')
   try
      D = feval(distfun,Y(1,:),X(1,:));
   catch ME
      if strcmp('MATLAB:UndefinedFunction', ME.identifier) ...
            && ~isempty(strfind(ME.message, func2str(distfun)))
         error(message('stats:pdist2:DistanceFunctionNotFound', func2str( distfun )));
      end
      % Otherwise, let the catch block below generate the error message
      D = [];
   end

   if ~isnumeric(D)
      error(message('stats:pdist2:OutputBadType'));
   end

   if ~isempty(radius)
      D = cell(1,ny);
      if nargout >= 2
         I = cell(1,ny);
      end

      for i = 1:ny
         try
            temp = feval(distfun,Y(i,:),X);
         catch ME
            if isa(distfun, 'inline')
               m = message('stats:pdist2:DistanceInlineError');
               ME2 = MException(m.Identifier,'%s',getString(m));
```

```
            throw(addCause(ME2,ME));
        else
            m = message('stats:pdist2:DistanceFunctionError',func2str(distfun));
            ME2 = MException(m.Identifier,'%s',getString(m));
            throw(addCause(ME2,ME));
        end
    end

    if nargout < 2
        D{i} = radiusSort(temp,radius);
    else
        [D{i},I{i}] = radiusSort(temp, radius);
    end
  end
elseif ~isempty(smallestLargestFlag)
  D = zeros(abs(smallestLargestFlag),ny,class(D));
  if nargout > 1
    I = zeros(abs(smallestLargestFlag),ny,class(D));
  end

  for i = 1:ny

    try
        temp = feval(distfun,Y(i,:),X);
    catch ME
        if isa(distfun, 'inline')
            m = message('stats:pdist2:DistanceInlineError');
            ME2 = MException(m.Identifier,'%s',getString(m));
            throw(addCause(ME2,ME));
        else
            m = message('stats:pdist2:DistanceFunctionError',func2str(distfun));
            ME2 = MException(m.Identifier,'%s',getString(m));
            throw(addCause(ME2,ME));
        end
    end

    if nargout < 2
        D(:,i) = partialSort(temp,smallestLargestFlag);
    else
        [D(:,i),I(:,i)] = partialSort(temp,smallestLargestFlag);
    end
  end

else  %compute all the pairwise distance
  % Make the return have whichever numeric type the distance function
  % returns.
  D = zeros(nx,ny,class(D));

  for i = 1:ny
    try
        D(:,i) = feval(distfun,Y(i,:),X);
```

```
        catch ME
            if isa(distfun, 'inline')
                m = message('stats:pdist2:DistanceInlineError');
                ME2 = MException(m.Identifier,'%s',getString(m));
                throw(addCause(ME2,ME));
            else
                m = message('stats:pdist2:DistanceFunctionError',func2str(distfun));
                ME2 = MException(m.Identifier,'%s',getString(m));
                throw(addCause(ME2,ME));
            end
        end
    end
end

%---------------------------------------------
% Normalize the data matrices X and Y to have unit norm
function [X,Y,flag] = normalizeXY(X,Y)
Xmax = max(abs(X),[],2);
X2 = bsxfun(@rdivide,X,Xmax);
Xnorm = sqrt(sum(X2.^2, 2));

Ymax = max(abs(Y),[],2);
Y2 = bsxfun(@rdivide,Y,Ymax);
Ynorm = sqrt(sum(Y2.^2, 2));
% Find out points for which distance cannot be computed.

% The norm will be NaN for rows that are all zeros, fix that for the test
% below.
Xnorm(Xmax==0) = 0;
Ynorm(Ymax==0) = 0;

% The norm will be NaN for rows of X that have any +/-Inf. Those should be
% Inf, but leave them as is so those rows will not affect the test below.
% The points can't be normalized, so any distances from them will be NaN
% anyway.

% Find points that are effectively zero relative to the point with largest norm.
flag =  any(Xnorm <= eps(max(Xnorm))) || any(Ynorm <= eps(max(Ynorm)));
Xnorm = Xnorm .* Xmax;
Ynorm = Ynorm .* Ymax;
X = bsxfun(@rdivide,X,Xnorm);
Y = bsxfun(@rdivide,Y,Ynorm);


function [D,I] = partialSort(D,smallestLargest)
if smallestLargest > 0
    n = smallestLargest;
else
    %sort(D,'descend') puts the NaN values at the beginning of the sorted list.
    %That is not what we want here.
```

```
    D = D*-1;
    n = -smallestLargest;
end

if nargout < 2
    D = sort(D,1);
    D = D(1:n,:);
else
    [D,I] = sort(D,1);
    D = D(1:n,:);
    I = I(1:n,:);
end

if smallestLargest < 0
    D = D * -1;
end

function [D,I] = radiusSort(D,radius)
I = find (D <= radius);
D = D(I);
if nargout < 2
    D = sort(D,1)'; %return a row vector
else
    [D,I2] = sort(D,1);
    D= D';
    I = I(I2)';
end
```

Η συνάρτηση αυτή επιστρέφει μια τιμή την οποία αποθηκεύουμε στη μεταβλητή d. Όσο πιο κοντά είναι αυτή η τιμή στο 0, τόσο μεγαλύτερη είναι η ομοιότητα των 2 εικόνων που είχα προς σύγκριση.

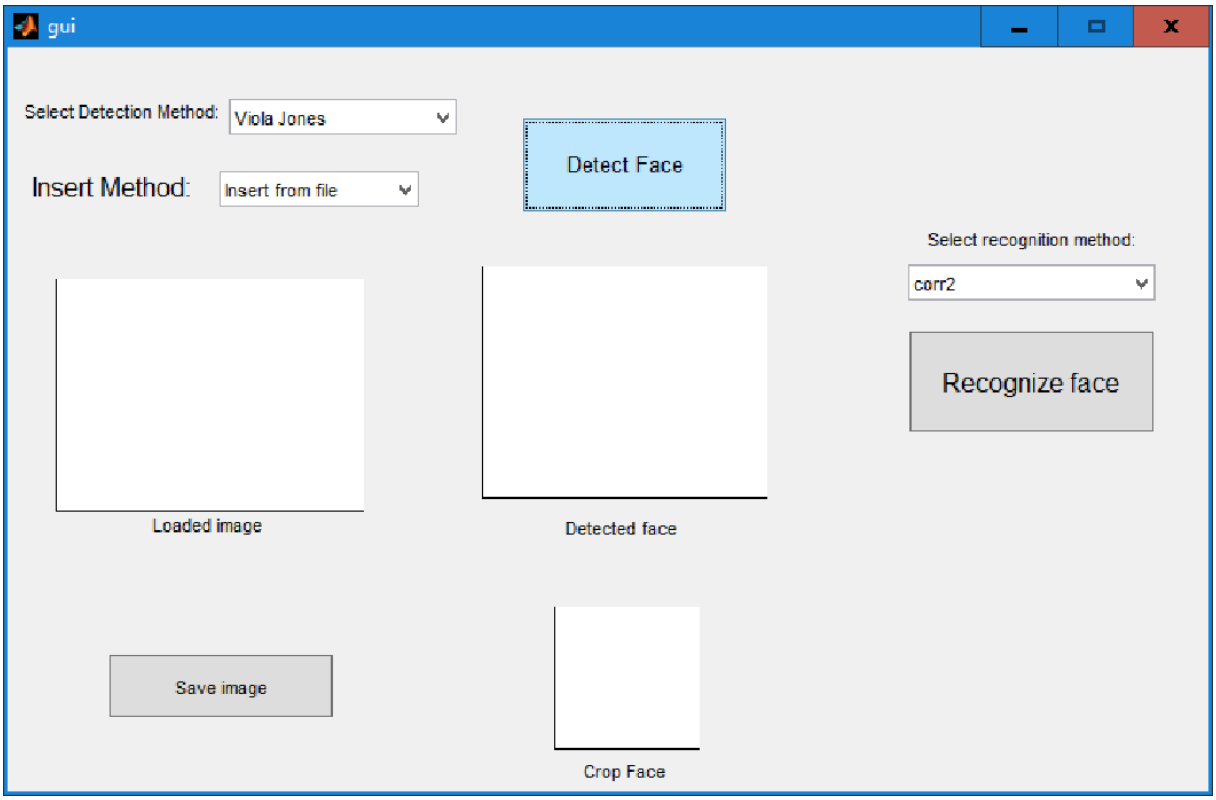
Κεφάλαιο 4

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΠΡΟΣΩΠΟΥ

4.1 Γενικά

Προκειμένου να γίνει η υλοποίηση ενός συστήματος ανίχνευσης και αναγνώρισης προσώπου χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον της matlab version 8.1.0.430 - R2013a. Χρησιμοποιήθηκε το γραφικό περιβάλλον γνωστό και ως GUI (Graphical User Interface) με σκοπό την δημιουργία ενός προγράμματος εύκολου και όμορφου προς το χρήστη. Ουσιαστικά το πρόγραμμά μας είναι χωρισμένο σε 2 μέρη. Το πρώτο μέρος αφορά την ανίχνευση ενός προσώπου και το δεύτερο αφορά την αναγνώριση.

Αρχικά ανοίγουμε το matlab και ανοίγουμε το φάκελο που περιέχει τις συναρτήσεις μας. Έπειτα πηγαίνουμε στο command line και πληκτρολογούμε τη λέξη gui. Αμέσως μας ανοίγει το γραφικό περιβάλλον και εμφανίζεται το παρακάτω παράθυρο:

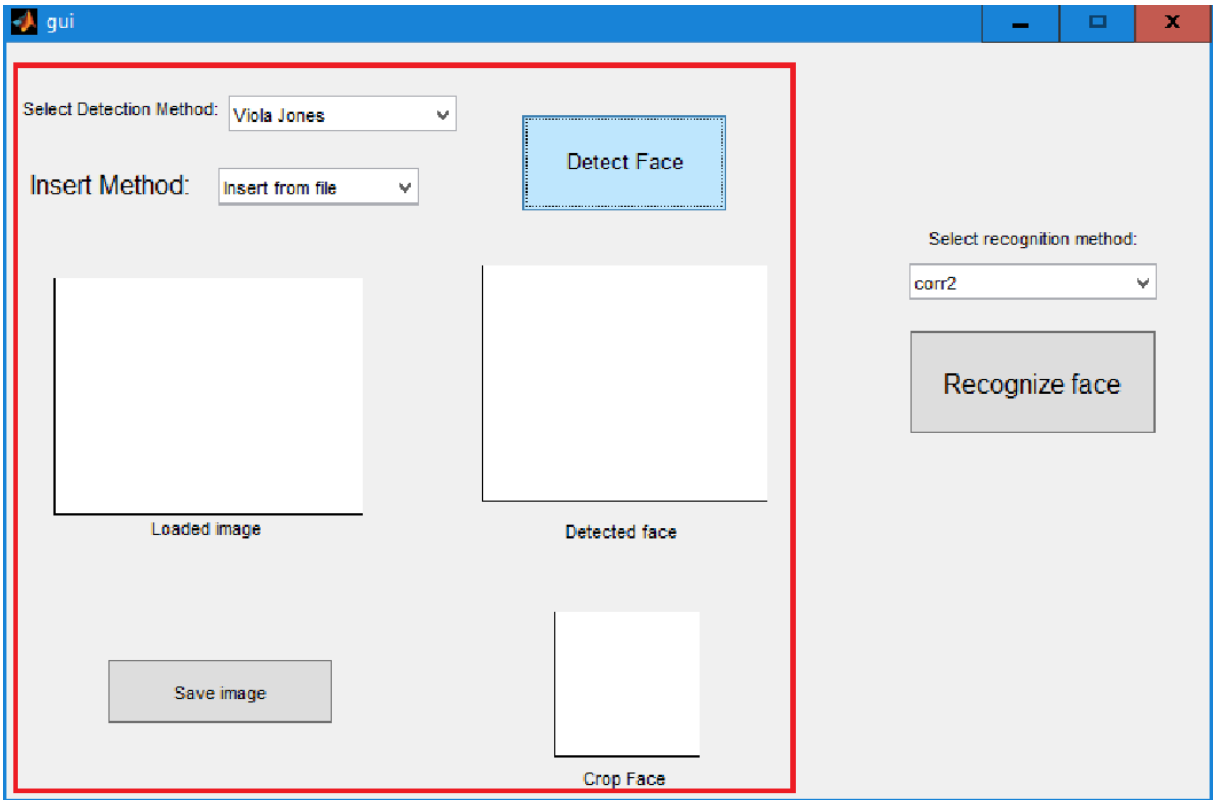


4.2 Ανίχνευση προσώπου

Ο αλγόριθμος που χρησιμοποίησα προκειμένου να κάνω ανίχνευση προσώπου είναι ο Viola Jones. Αυτό στο matlab υλοποιείται με τις εξής συναρτήσεις:

```
faceDetector = vision.CascadeObjectDetector;

bbox = step(faceDetector, photo);

out = insertObjectAnnotation(photo,'rectangle',bbox,'Face');
```

Στην παρακάτω εικόνα με κόκκινο πλαίσιο φαίνεται ποιο μέρος του gui αφορά το face recognition:



Αρχικά επιλέγουμε από ένα drop down menu τη μέθοδο που θέλουμε να φορτώσουμε μια εικόνα. Οι επιλογές μας είναι:

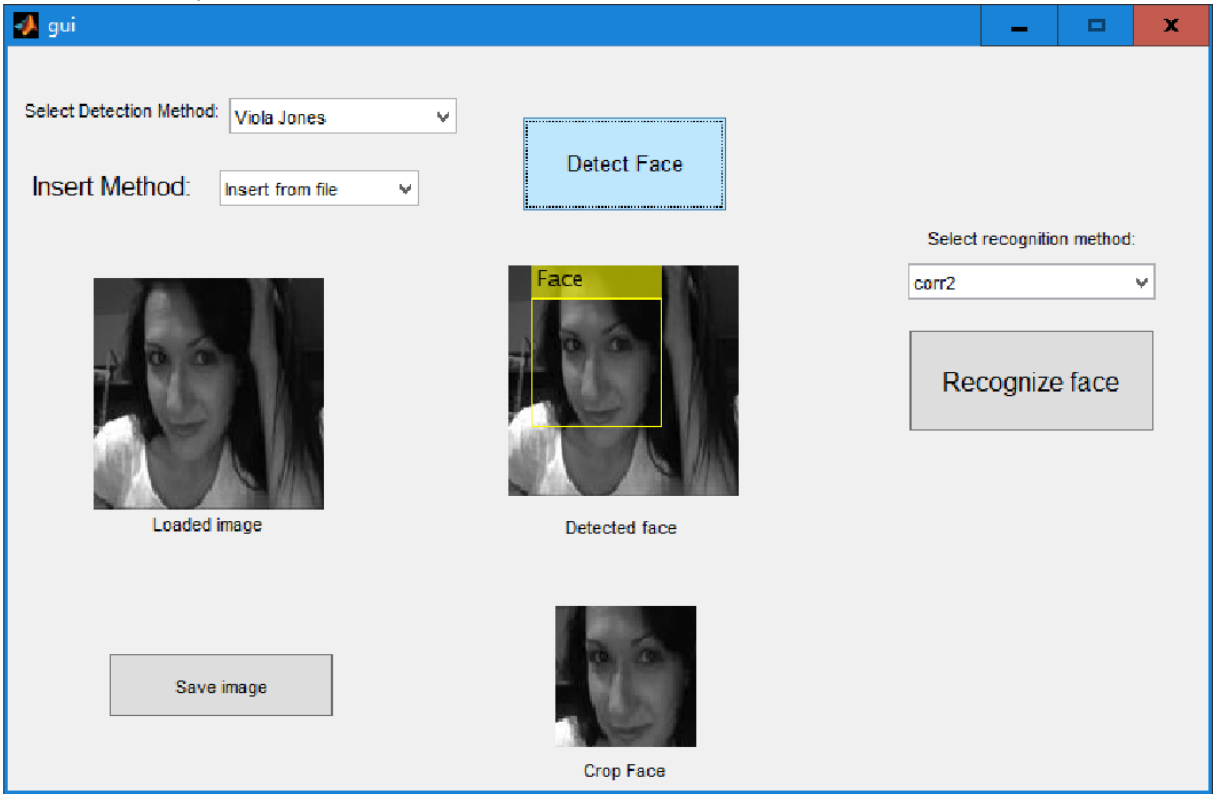
1) Insert from file

2) Insert from camera

Μόλις επιλέξουμε τη μέθοδο πατάμε το κουμπί Detect Face και τρέχει η συνάρτηση function DetectFace_Callback(hObject, eventdata, handles).
Εάν επιλέξουμε insert from file ανοίγει ένα νέο παράθυρο και περιμένει να επιλέξουμε μια εικόνα που είναι ήδη αποθηκευμένη στον υπολογιστή μας. Μόλις επιλεγεί μια εικόνα τότε γίνονται οι εξής ενέργειες:

1) Φορτώνεται η εικόνα και εμφανίζεται στο παράθυρο Loaded Image του gui
2)Γίνεται ανίχνευση προσώπου, ζωγραφίζεται ένα κίτρινο πλαίσιο γύρω από το πρόσωπο και το αποτέλεσμα αυτό εμφανίζεται στο παράθυρο Detected Face στο gui μας.
3)Εμφανίζεται στο παράθυρο Crop Face του gui μόνο το πρόσωπο που ανιχνεύθηκε

Ακολουθεί παράδειγμα:



Εάν επιλέξουμε insert from camera και πατήσουμε το κουμπί detect face γίνονται οι παρακάτω ενέργειες:
1) Αυτομάτως ανοίγει η εσωτερική κάμερα του laptop και ¨τραβάει¨ μια φωτογραφία η οποία εμφανίζεται στο παράθυρο Loaded Image του gui
2)Γίνεται ανίχνευση προσώπου, ζωγραφίζεται ένα κίτρινο πλαίσιο γύρω από το πρόσωπο και το αποτέλεσμα αυτό εμφανίζεται στο παράθυρο Detected Face στο gui μας.
3)Εμφανίζεται στο παράθυρο Crop Face του gui μόνο το πρόσωπο που ανιχνεύθηκε

Η διαδικασία που περιγράψαμε παραπάνω μπορεί να επαναληφθεί πολλές φορές έως ότου δούμε στην οθόνη μας το επιθυμητό αποτέλεσμα.

Όταν λοιπόν πάρουμε την επιθυμητή εικόνα είτε επιλέγοντας insert from file είτε επιλέγοντας insert from camera υπάρχει η δυνατότητα να αποθηκεύσουμε την εικόνα αυτή σε ένα φάκελο. Αυτό γίνεται πατώντας το κουμπί save image. Πατώντας αυτό το κουμπί ανοίγει ένα παράθυρο και μας ζητάει να επιλέξουμε που θα αποθηκευτεί η εικόνα μας και με ποιο όνομα.

4.3 Αναγνώριση προσώπου

Προκειμένου να γίνει αναγνώριση προσώπου πρέπει πρώτα απ' όλα να υπάρχει μια βάση δεδομένων στην οποία θα ψάχνει το πρόγραμμά μας να κάνει ταυτοποίηση προσώπου. Αυτή βρίσκεται στην επιφάνεια εργασίας σε ένα φάκελο με όνομα FACES. Προσοχή: Ο φάκελος FACES πρέπει να περιέχει εικόνες τύπου .jpg και τα ονόματα των εικόνων θα πρέπει να ξεκινάνε από το νούμερο 1 και να συνεχίζουν με αύξουσα σειρά.

Αφού λοιπόν θα έχουμε δημιουργήσει μια βάση δεδομένων τώρα θα πρέπει να φορτώσουμε μια εικόνα όπως περιγράψαμε στην ενότητα 4.2, έπειτα θα πρέπει να επιλέξουμε με ποια μέθοδο θέλουμε να γίνει η αναγνώριση - ταυτοποίηση προσώπου και να πατήσουμε το κουμπί Recognize Face.

Οι μέθοδοι που χρησιμοποιήθηκαν για να γίνει η αναγνώριση προσώπου είναι οι εξής:

1) 2-d correlation coefficient

2) Histogram and Euclidean distance

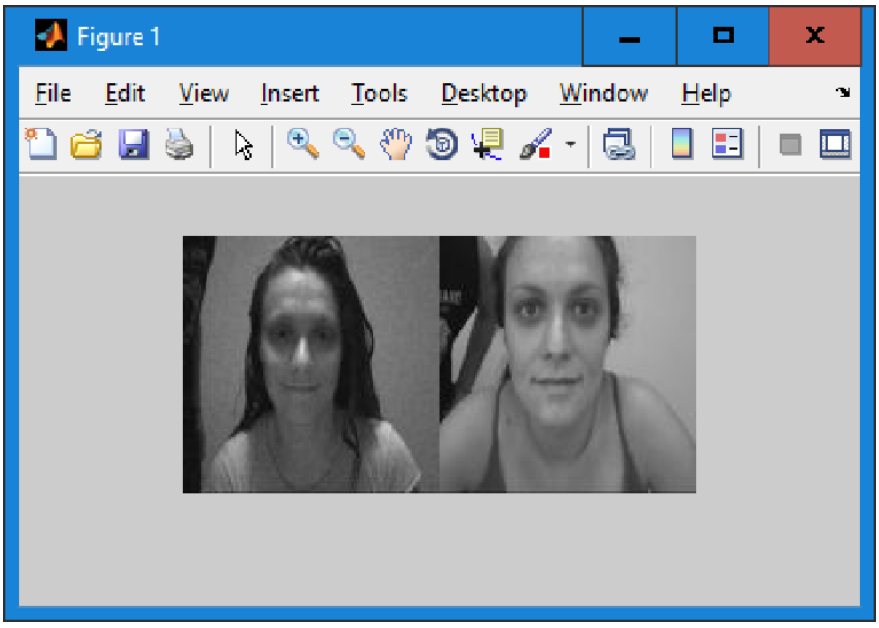
3) Structural similarity

Με αυτή τη σειρά είναι και οι επιλογές που έχει τη δυνατότητα να επιλέξει ο χρήστης στο drop down menu στο gui μας.

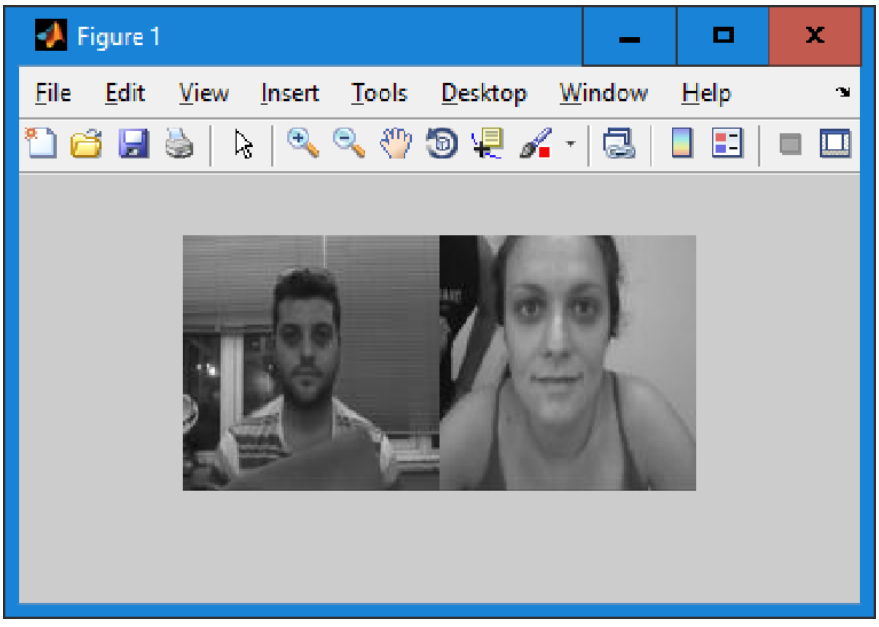
Όταν ο χρήστης πατήσει το κουμπί Recognize Face εκτελούνται οι εξής ενέργειες:

1) Συγκρίνει το πρόσωπο της εικόνας που βρίσκεται στο παράθυρο Loaded Image στο gui με το πρόσωπο της κάθε εικόνας που βρίσκεται στο φάκελο FACES που είναι στην επιφάνεια εργασίας. Ο φάκελος FACES είναι για εμάς η database μας.

2) Σύμφωνα με τη μέθοδο που έχουμε επιλέξει, γίνεται η ταυτοποίηση προσώπων και εμφανίζεται ένα νέο παράθυρο με τα 2 πρόσωπα που έκανε το matching ο αλγόριθμος.



Εικόνα 4.3.1. Παράδειγμα επιτυχημένης ταυτοποίησης προσώπου



Εικόνα 4.3.2 Παράδειγμα αποτυχημένης ταυτοποίησης προσώπου

Παραπάνω στην εικόνα 4.3.1 έχουμε μια επιτυχημένη προσπάθεια ταυτοποίησης προσώπου. Αυτό όμως δεν σημαίνει ότι οι αλγόριθμοι μας έχουν πάντα επιτυχία. Αυτό θα εξετάσουμε παρακάτω κάνοντας συγκρίσεις στις 3 μεθόδους που χρησιμοποιήσαμε εξάγοντας χρήσιμα συμπεράσματα για την κάθε μέθοδο.

4.4 Πειραματικό Μέρος

4.4.1 Γενικά

Όταν καλείται το πρόγραμμά μας να κάνει face recognition, ανάλογα με τη μέθοδο που έχουμε επιλέξει, καλείται και η αντίστοιχη συνάρτηση. Πιο συγκεκριμένα αν επιλέξουμε τη μέθοδο 2-d correlation coefficient καλείται η συνάρτηση recog_corr2(), αν επιλέξουμε τη μέθοδο histogram καλείται η συνάρτηση recog_histogram() ενώ αν επιλέξουμε τη μέθοδο structural similarity index method καλείται η συνάρτηση recog_ssim. Προκειμένου να γίνει η ταυτοποίηση προσώπου γίνονται συγκρίσεις προσώπων. Για να το επιτύχουμε αυτό το πρόγραμμα μας μπαίνει σε ένα loop και μετά από κάθε σύγκριση προσώπου επιστρέφεται μια τιμή η οποία μας δείχνει πόσο όμοια είναι τα 2 πρόσωπα. Όταν τα πρόσωπα είναι τελείως όμοια η τιμή αυτή παίρνει τα ακόλουθα αποτελέσματα:

Μέθοδος corr2: Όταν τα πρόσωπα είναι όμοια -> h=1

Μέθοδος histogram: Όταν τα πρόσωπα είναι όμοια -> h=0

Μέθοδος ssim: Όταν τα πρόσωπα είναι όμοια -> h=1

Και στις 3 παραπάνω περιπτώσεις υπάρχει πλήρη ταύτιση προσώπων.

Μετά από πειραματικές δοκιμές έχουμε σετάρει την μεταβλητή h σε μια συγκεκριμένη τιμή για κάθε μέθοδο ώστε να μας δίνει πιο πετυχημένα αποτελέσματα αλλιώς μας εμφανίζει το μήνυμα ("Face Not Detected"). Εγώ έχω δώσει στην μεταβλητή μου στις ακόλουθες τιμές:

Για τη μέθοδο corr2 εμφανίζει αποτελέσματα αν (h > 0.4)

Για τη μέθοδο histogram εμφανίζει αποτελέσματα αν (h < 0.07)

Για τη μέθοδο ssim εμφανίζει αποτελέσματα αν (h > 0.4) && (h < 1.1)

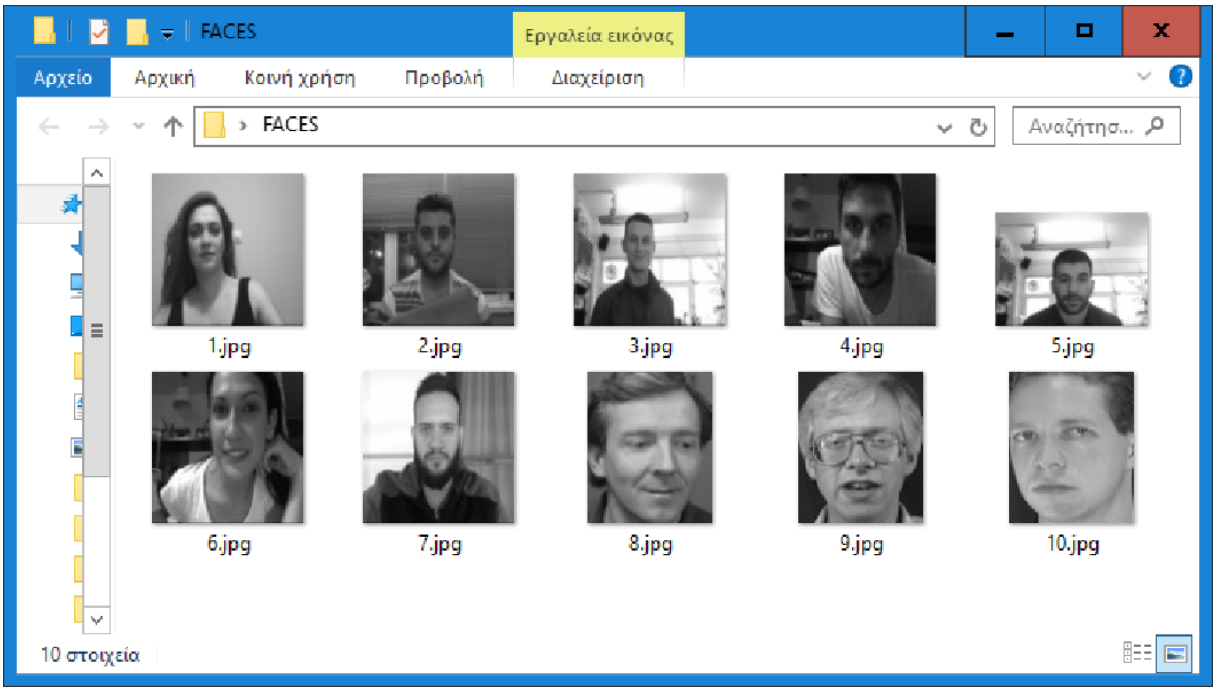
Οι τιμές αυτές μπορούν φυσικά να αλλάξουν και να προσαρμοστούν ανάλογα με τις απαιτήσεις μας.

4.4.2 Πειραματικές Δοκιμές

Τώρα ήρθε η ώρα να δούμε πως ανταποκρίνεται ο αλγόριθμός μας στην πράξη.

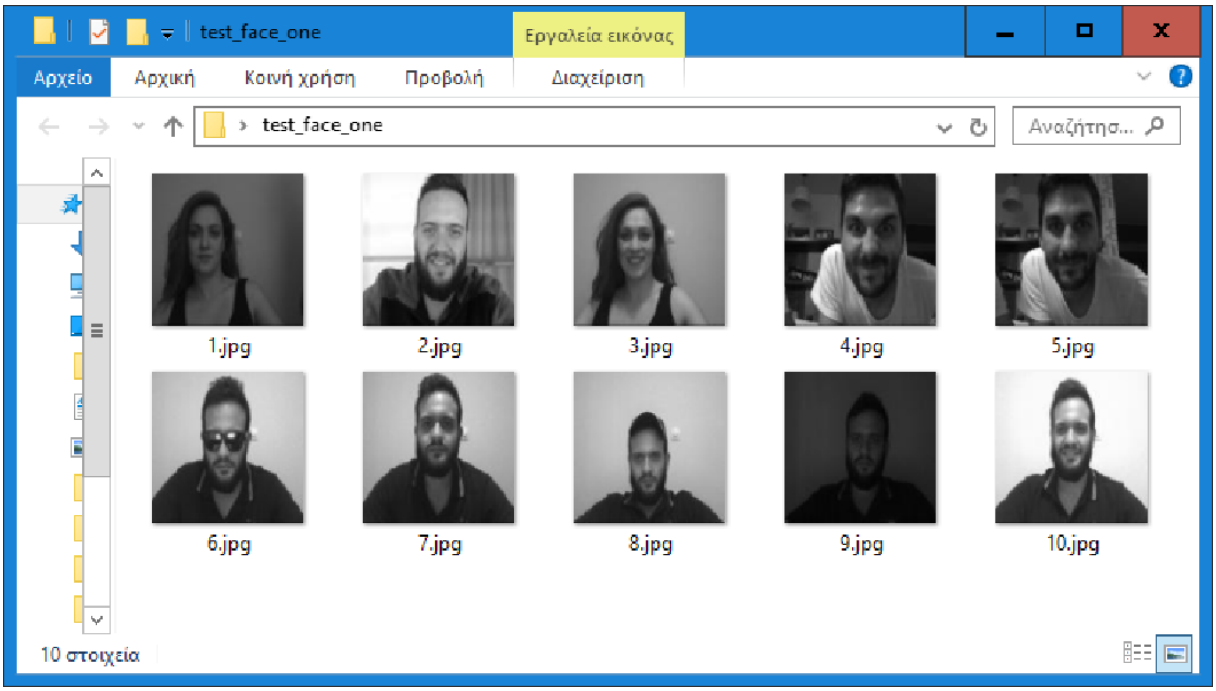
Δοκιμή 1.

Βάλαμε σε σχόλιο τον περιορισμό που βάλαμε στην τιμή h έτσι ώστε να μας δίνει πάντα αποτέλεσμα και να μην βγάζει το μήνυμα "face not detected". Έτσι θα δούμε πόσο επιτυχημένη σύγκριση κάνει στα πρόσωπα το πρόγραμμά μας. Βασική προϋπόθεση όμως είναι στη database μας να υπάρχει το πρόσωπο που ψάχνουμε να κάνουμε ταυτοποίηση, αλλιώς θα κάνουμε πάντα matching με λανθασμένο πρόσωπο. Για την πρώτη δοκιμή επιλέχτηκε η παρακάτω database.



database 1

Παρακάτω φαίνονται 10 φωτογραφίες με τις οποίες θα προσπαθήσουμε να κάνουμε ταυτοποίηση προσώπου με τα πρόσωπα που βρίσκονται στην παραπάνω database.



test_face_one

Κάναμε τη δοκιμή και με τις 3 μεθόδους και καταγράψαμε τα ποσοστά επιτυχίας ταυτοποίησης προσώπου:

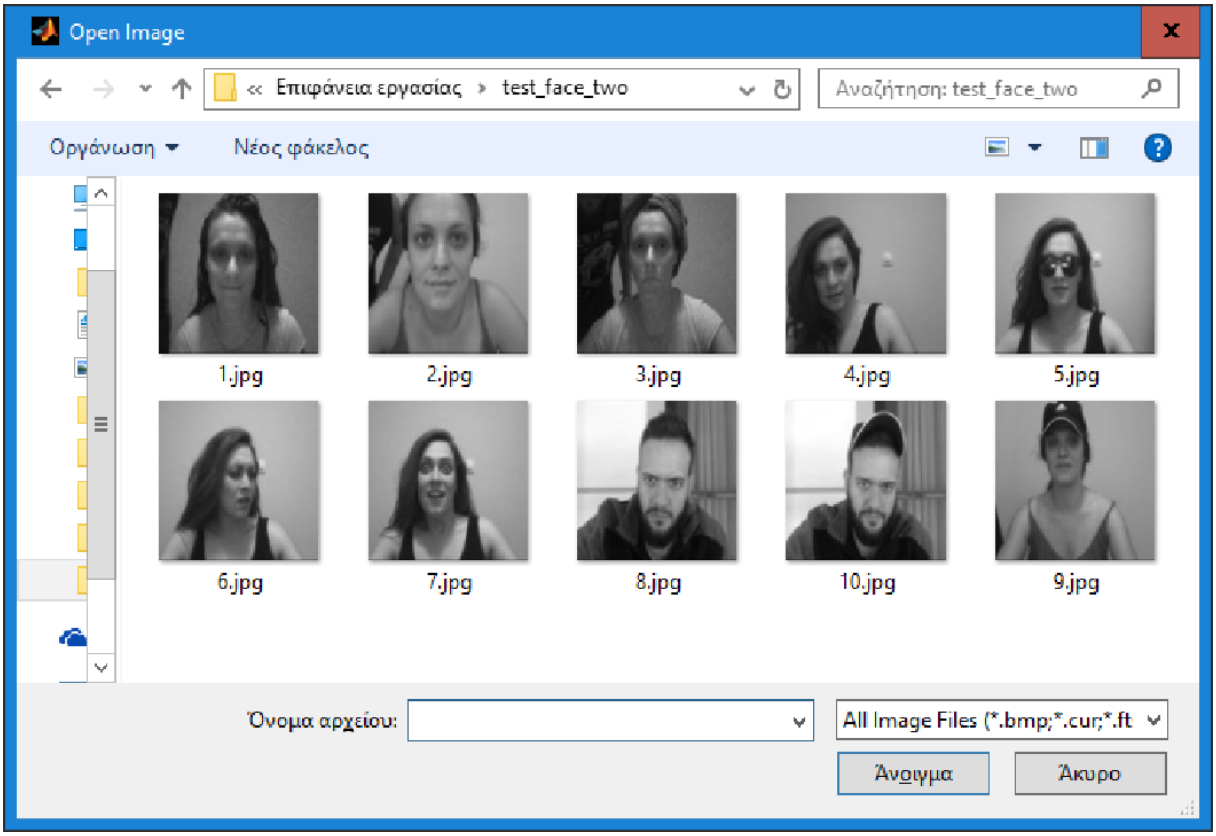
Μέθοδος corr2: 70% επιτυχία

Μέθοδος histogram: 30% επιτυχία

Μέθοδος ssim: 40% επιτυχία

Δοκιμή 2

Με την ίδια database θα κάνουμε ξανά δοκιμή. Παρακάτω φαίνονται 10 φωτογραφίες με τις οποίες θα προσπαθήσουμε να κάνουμε ταυτοποίηση προσώπου.



test_face_two

Κάναμε τη δοκιμή και με τις 3 μεθόδους και καταγράψαμε τα ποσοστά επιτυχίας ταυτοποίησης προσώπου:

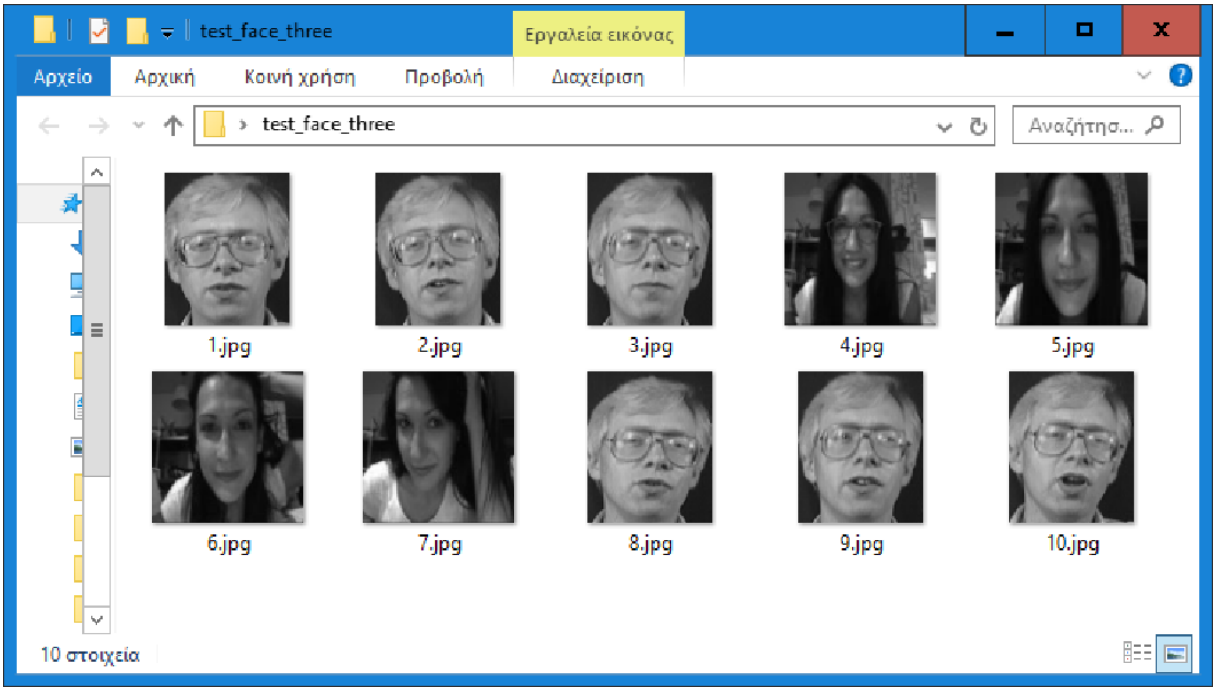
Μέθοδος corr2: 80% επιτυχία

Μέθοδος histogram: 30% επιτυχία

Μέθοδος ssim: 50% επιτυχία

Δοκιμή 3

Με την ίδια database θα κάνουμε ξανά δοκιμή. Παρακάτω φαίνονται 10 φωτογραφίες με τις οποίες θα προσπαθήσουμε να κάνουμε ταυτοποίηση προσώπου.



test_face_three

Κάναμε τη δοκιμή και με τις 3 μεθόδους και καταγράψαμε τα ποσοστά επιτυχίας ταυτοποίησης προσώπου:

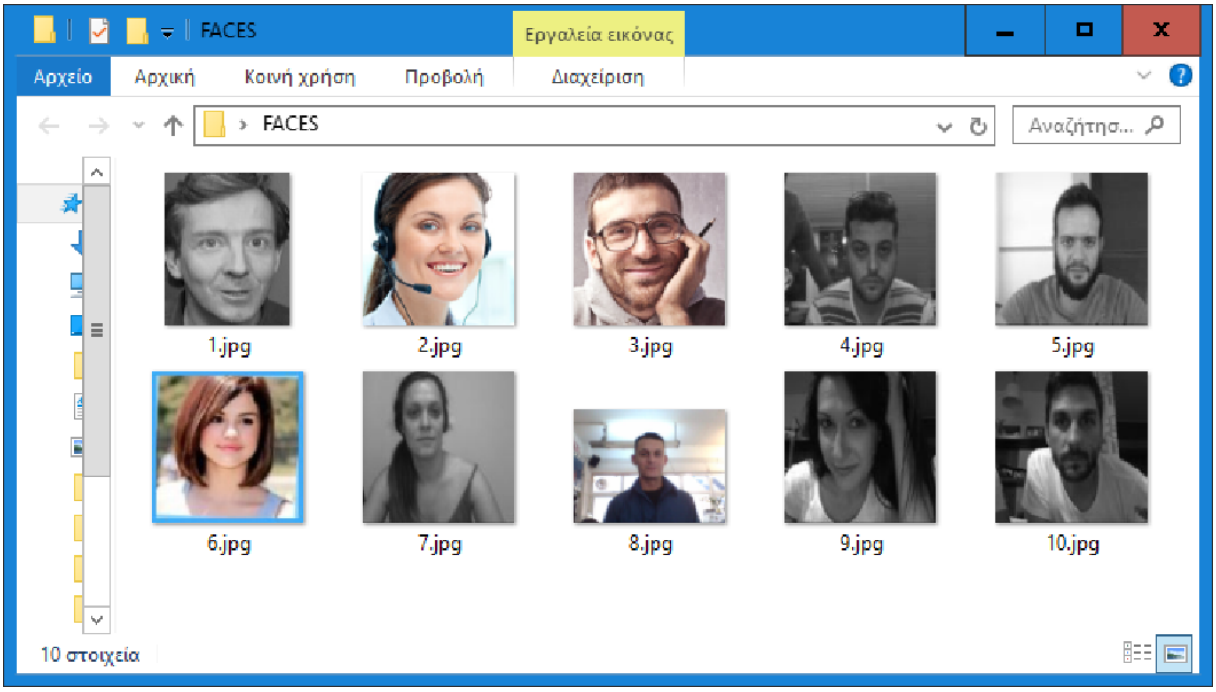
Μέθοδος corr2: 80% επιτυχία

Μέθοδος histogram: 100% επιτυχία

Μέθοδος ssim: 100% επιτυχία

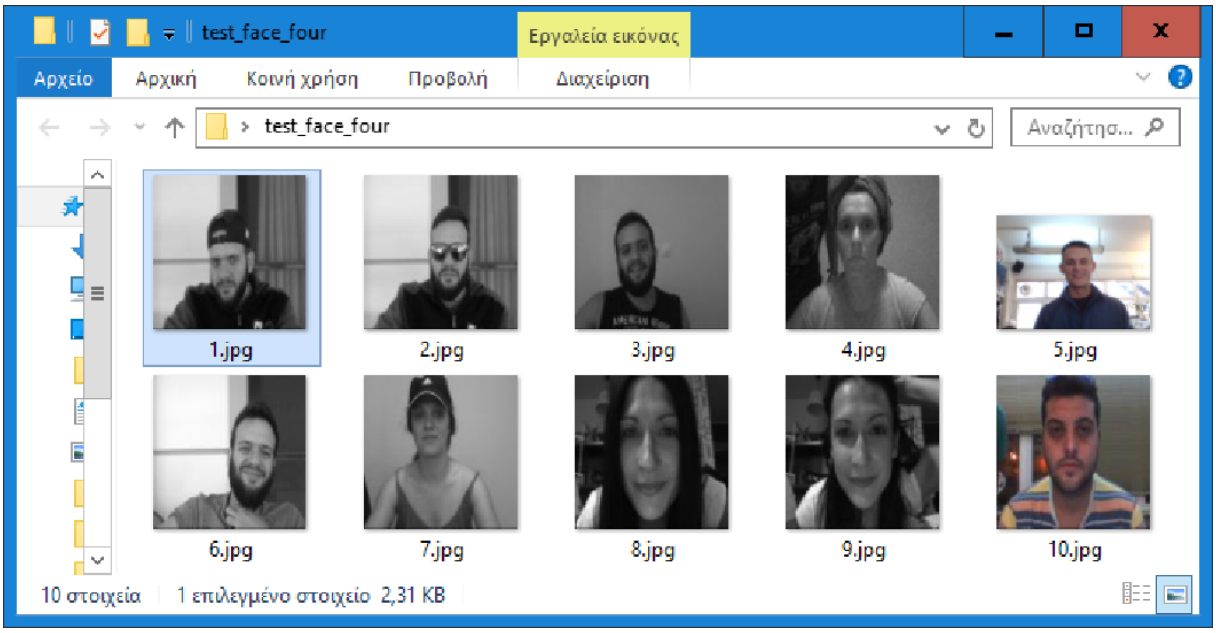
Δοκιμή 4

Τώρα θα αλλάξουμε τη database μας. Παρακάτω φαίνεται η καινούρια database:



database_2

Παρακάτω φαίνονται 10 φωτογραφίες με τις οποίες θα προσπαθήσουμε να κάνουμε ταυτοποίηση προσώπου στη database_2:



test_face_four

Κάναμε τη δοκιμή και με τις 3 μεθόδους και καταγράψαμε τα ποσοστά επιτυχίας ταυτοποίησης προσώπου:

Μέθοδος corr2: 60% επιτυχία

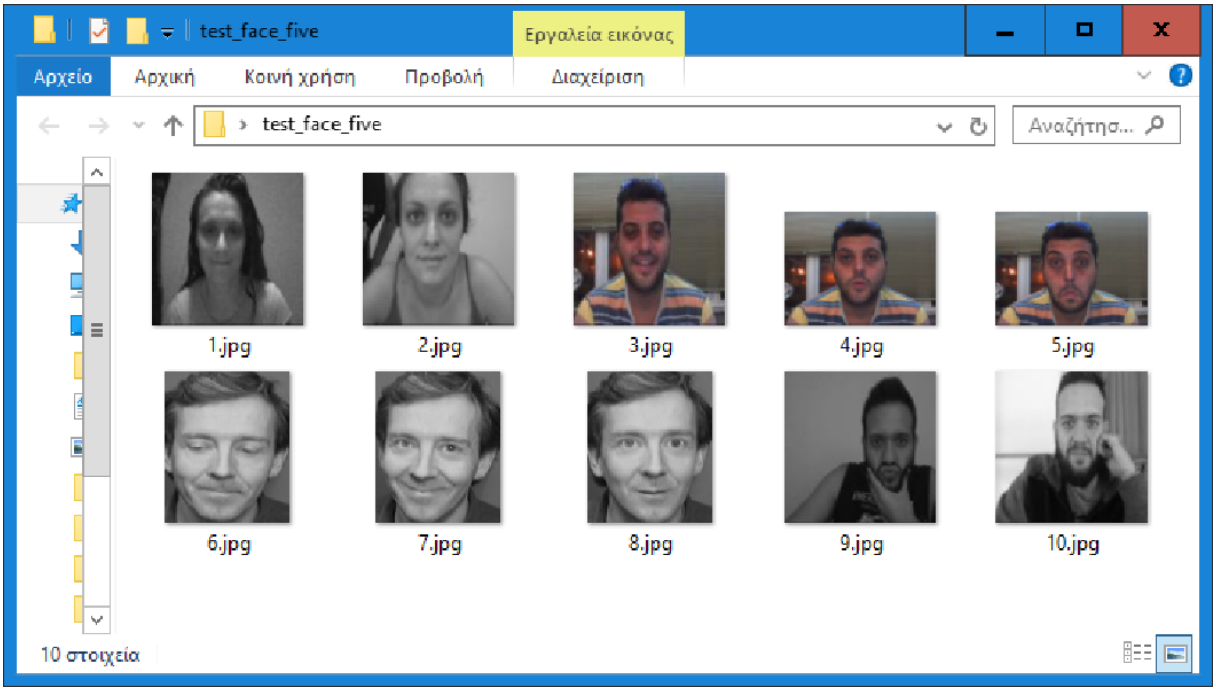
Μέθοδος histogram: 40% επιτυχία

Μέθοδος ssim: 70% επιτυχία

Δοκιμή 5

Με τη database_2 θα κάνουμε μια τελευταία δοκιμή.

Παρακάτω φαίνονται 10 φωτογραφίες με τις οποίες θα προσπαθήσουμε να κάνουμε ταυτοποίηση προσώπου στη database_2:



test_face_five

Κάναμε τη δοκιμή και με τις 3 μεθόδους και καταγράψαμε τα ποσοστά επιτυχίας ταυτοποίησης προσώπου:

Μέθοδος corr2: 70% επιτυχία

Μέθοδος histogram: 80% επιτυχία

Μέθοδος ssim: 70% επιτυχία

4.5 Συμπερασματα - Αποτελέσματα

Τελικά αποτελέσματα επιτυχίας ταυτοποίησης προσώπου:

Μέθοδος corr2: 72% επιτυχία

Μέθοδος histogram: 56% επιτυχία

Μέθοδος ssim: 66% επιτυχία

Σύμφωνα με τις πειραματικές δοκιμές που έκανα ο αλγόριθμος που ανταποκρίνεται καλύτερα είναι η μέθοδος 2-d correlation coefficient με ποσοστό επιτυχίας 72%. Βέβαια αυτά τα ποσοστά σίγουρα θα αλλάξουν αν μεταβάλλουμε συνθήκες όπως ο φωτισμός, οι σκιές, οι μορφασμοί των προσώπων ή αν φοράμε διάφορα αντικείμενα όπως γυαλιά ή καπέλο. Εμείς προσπαθήσαμε να κάνουμε διάφορες δοκιμές σε διαφορετικές συνθήκες και πήραμε τα παραπάνω αποτελέσματα. Ο αλγόριθμος histogram φαίνεται να μην ανταποκρίνεται και τόσο καλά ειδικά όταν τα δυο πρόσωπα που συγκρίνονται είναι σε διαφορετικό φωτισμό ή όταν το πρόσωπο κάνει κάποιο μορφασμό. Το καλύτερο αποτέλεσμα που κατάφερε να κάνει είναι να κάνει σωστή ταυτοποίηση σε περίπου ένα στα δυο πρόσωπα. Ένας λόγος που συμβαίνει αυτό είναι στο ότι όλη η διπλωματική έχει στηθεί σε εικόνες δυο διαστάσεων, δηλαδή σε grayscale εικόνες και τα δεδομένα μας είναι λιγότερα από ότι σε σε μια έγχρωμη εικόνα.

4.6 Τροποποιήσεις στα codes και προσωπική συμβολή στην εργασία

Όλη η εργασία μου δημιουργήθηκε στο προγραμματιστικό περιβάλλον της matlab. Προκειμένου να εχω ένα ευχάριστο περιβάλλον χρησιμοποίησα το gui της matlab και δημιούργησα ένα δικό μου γραφικό περιβάλλον στο οποίο ο χρήστης εχει τη δυνατότητα να να κάνει face detection και face recognition και να βλέπει τα αποτελέσματα κατευθείαν στην οθόνη του. Για να υλοποιήσω το face detection χρησιμοποιήθηκε η μέθοδος viola janes και το παρακάτω link: https://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-camshift.html

Εκει παρέχει πληροφορίες για κάποια έτοιμα εργαλεία του matlab που χρησιμοποιήθηκαν.

```
% Create a cascade detector object.
faceDetector = vision.CascadeObjectDetector();

% Read a video frame and run the detector.
videoFileReader = vision.VideoFileReader('visionface.avi');
videoFrame      = step(videoFileReader);
bbox            = step(faceDetector, videoFrame);

% Draw the returned bounding box around the detected face.
videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'Face');
figure, imshow(videoOut), title('Detected face');
```

Στο δεύτερο μέρος της εργασίας μου γίνεται το face recognition. Δημιούργησα δικό μου κώδικα ώστε να κάνω σύγκριση μιας reference εικόνας με την κάθε εικόνα μιας database με πολλές εικόνες με σκοπό να κανω αναγνώριση και ταυτοποίηση προσώπου. Ουσιαστικά γίνονται αλλεπάλληλες συγκρίσεις δυσδιάστατων πινάκων (εικόνες) με σκοπό να βρω ποτε έχω το μικρότερο σφάλμα. Δηλαδή ποιες εικόνες έχουν τις λιγότερες διαφορές ώστε να κάνω το σωστό matching προσώπων και να κάνω όσο το δυνατόν πιο επιτυχημένο face recognition. Για να το επιτύχω αυτό χρησιμοποίησα 3 διαφορετικές μεθόδους. 1) Έκανα συγκρίσεις εικόνων με τη μέθοδο 2D correlation coefficient. Για να το επιτύχω αυτό χρησιμοποίησα το έτοιμο εργαλείο που μου παρέχει το matlab: η συνάρτηση corr2(). Παρακάτω ακολουθεί το link: https://www.mathworks.com/help/images/ref/corr2.html?s_tid=srchtitle

Το αποτέλεσμα μετά από κέθε σύγκριση είναι μια τιμή η οποία θα είναι για εμάς ένας δείκτης ομοιότητας των 2 εικόνων. Όσο μεγαλύτερη είναι αυτή η τιμή τόσο μεγαλύτερη είναι η ομοιότητα των 2 προσώπων.

2) Η δεύτερη μέθοδος που χρησιμοποίησα είναι η μέθοδος histogram. Ουσιαστικά σύγκρινα το ιστόγραμμα της μιας εικόνας με το ιστόγραμμα της άλλης. Για να το επιτύχω αυτό χρησιμοποίησα το παρακάτω link που είναι στο community της mathworks: https://www.mathworks.com/matlabcentral/answers/85447-comparison-of-two-histograms-using-pdist2

Επίσης χρησιμοποιήθηκαν τα έτοιμα εργαλεία που μας προσφέρει η matlab: οι συναρτήσεις imhist( ) και pdist2( ). Ακολουθούν τα link των συναρτήσεων:

https://www.mathworks.com/help/images/ref/imhist.html?searchHighlight=imhist

https://www.mathworks.com/help/stats/pdist2.html

Το αποτέλεσμα μετά από κέθε σύγκριση είναι μια τιμή η οποία θα είναι για εμάς ένας δείκτης ομοιότητας των 2 εικόνων. Όσο μικρότερη είναι αυτή η τιμή τόσο μεγαλύτερη είναι η ομοιότητα των 2 προσώπων.

3) Η τρίτη μέθοδος που χρησιμοποίησα είναι η μέθοδος ssim. Για να το επιτύχω αυτό χρησιμοποίησα το έτοιμο εργαλείο που μου παρέχει η matlab: η συνάρτηση ssim(). Ουσιαστικά είναι ενας δείκτης δομημένης ομοιότητας που χρησιμοποιείται σε συγκρίσεις εικόνων. Ακολουθεί το link της συνάρτησης: https://www.mathworks.com/help/images/ref/ssim.html

Το αποτέλεσμα μετά από κέθε σύγκριση είναι μια τιμή η οποία θα είναι για εμάς ένας δείκτης ομοιότητας των 2 εικόνων. Όσο μεγαλύτερη είναι αυτή η τιμή τόσο μεγαλύτερη είναι η ομοιότητα των 2 προσώπων.

Αυτό που κατάφερα να κάνω σε αυτή τη διπλωματική είναι με τη βοήθεια των εργαλείων που μας προσφέρει το matlab να δημιουργήσω ένα σύστημα ανίχνευσης, αναγνώρισης και ταυτοποίησης προσώπων μέσα από ένα ευχάριστο γραφικό περιβάλλον. Χρησιμοποίησα 1 μέθοδο για ανίχνευση προσώπου και 3 διαφορετικές μεθόδους αναγνώρισης προσώπων ρυθμίζοντας κατάλληλα τις παραμέτρους της κάθε μεθόδου ώστε να μας επιφέρει καλύτερα αποτελέσματα (η μεταβλητή h στο πρόγραμμά μας). Μετά από πειραματικές δοκιμές εξήγαγα τα αποτελέσματα των συγκρίσεων της κάθε μεθόδου και το ποσοστό επιτυχίας τους. Το συνολικό όφελος της εργασίας είναι που καταφέραμε και παντρέψαμε τη μέθοδο viola jones, και τις μεθόδους structural similarity, 2d dorrelation coefficient και histogram με σκοπό τη δημιουργία ενός ολοκληρωμένου συστήματος ανίχνευσης και αναγνώρισης προσώπων. Ρυθμίσαμε τις παραμέτρους ώστε να έχουμε τα καλύτερα αποτελέσματα και καταγράψαμε τα αποτελέσματα μετά από πειραματικές δοκιμές.

Κεφάλαιο 5

5.1 Μελλοντική επέκταση του προγράμματος

Το πρόγραμμά μας έχει υλοποιηθεί έτσι ώστε πολύ εύκολα να μπορέσουμε να προσθέσουμε και άλλες μεθόδους ανίχνευσης προσώπου και να εμφανίσουμε τα αποτελέσματα στο ήδη υπάρχον γραφικό περιβάλλον. Επίσης μπορούμε να εισάγουμε άλλες μεθόδους αναγνώρισης προσώπου όπως 3-D εικόνες ή ακόμα και μεθόδους αναγνώρισης της ίριδας (iris detection). Σίγουρα αυτές οι μέθοδοι είναι πιο ακριβείς στις συγκρίσεις τους και μπορούν να μας φέρουν πιο επιτυχημένα αποτελέσματα. Τέλος το πρόγραμμά μας θα μπορούσε να εφαρμοστεί σε σύστημα αναγνώρισης προσώπου από κινούμενη εικόνα (video).

5.2 Ολόκληρο το πρόγραμμά μας που υλοποιήθηκε στο προγραμματιστικό περιβάλλον της matlab

function gui.m :

```
function varargout = gui(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
   'gui_Singleton',  gui_Singleton, ...
   'gui_OpeningFcn', @gui_OpeningFcn, ...
   'gui_OutputFcn',  @gui_OutputFcn, ...
   'gui_LayoutFcn',  [] , ...
   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

global photo
photo=[];
% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

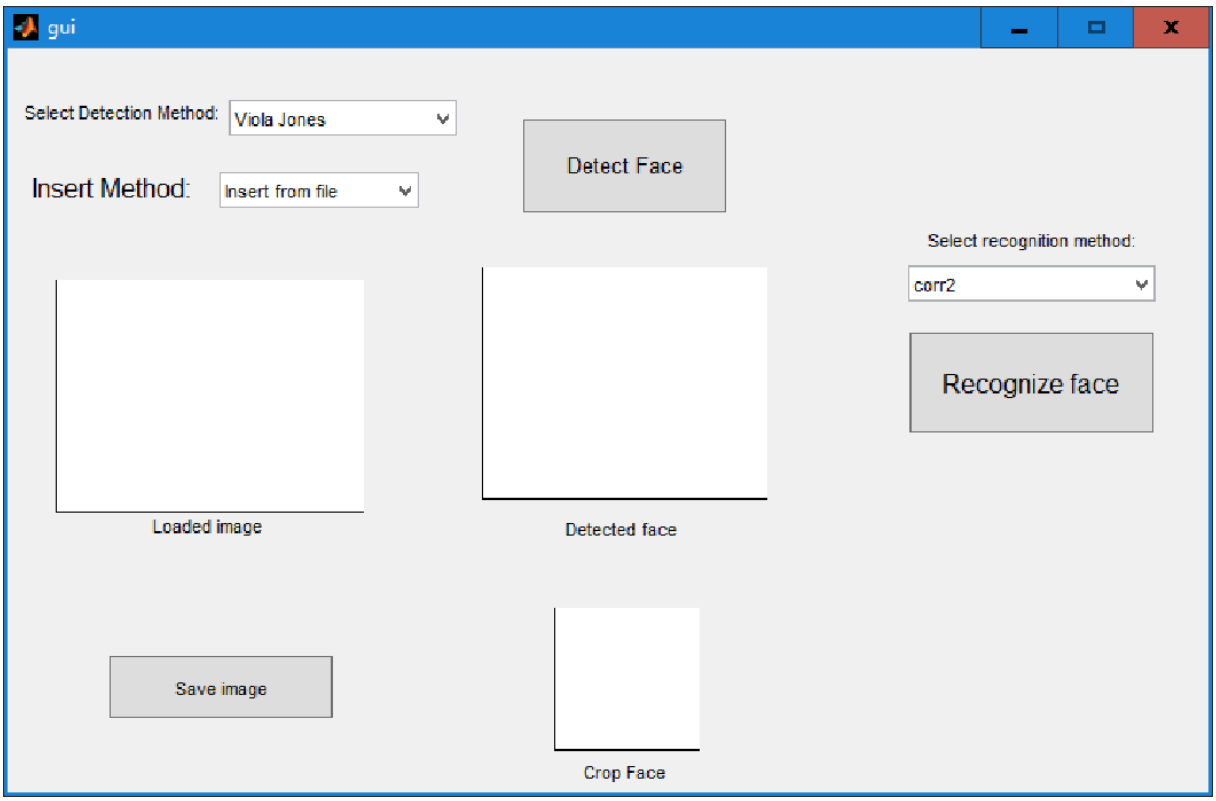
```
% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in InsertMethod.
function InsertMethod_Callback(hObject, eventdata, handles)
% hObject    handle to InsertMethod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns InsertMethod contents as cell array
%        contents{get(hObject,'Value')} returns selected item from InsertMethod


% --- Executes during object creation, after setting all properties.
function InsertMethod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InsertMethod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in RecognitionMethod.
function RecognitionMethod_Callback(hObject, eventdata, handles)
% hObject    handle to RecognitionMethod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns RecognitionMethod contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from RecognitionMethod



% --- Executes during object creation, after setting all properties.
function RecognitionMethod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RecognitionMethod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




% --- Executes on button press in DetectFace.
function DetectFace_Callback(hObject, eventdata, handles)
val = get(handles.InsertMethod,'Value');
global photo out
switch val
    case 1  %Insert from file

        [path,user_canceled]=imgetfile();
        if user_canceled
            msgbox(sprintf('You didnt select an image'),'Error','Error');
            return
        end

        photo=imread(path);

        axes(handles.LoadedImage);
```

```
        imshow(photo);

        faceDetector = vision.CascadeObjectDetector;
        bbox = step(faceDetector, photo);
        out3=imcrop(photo,bbox);
        out = insertObjectAnnotation(photo,'rectangle',bbox,'Face');
        axes(handles.detectedFace);
        imshow(out);

        axes(handles.CropFace);
        imshow(out3);



    case 2  %Insert from camera

        vid = videoinput('winvideo', 1,'MJPG_640x480');
        set(vid, 'ReturnedColorSpace');
        photo = getsnapshot(vid);
        axes(handles.LoadedImage);
        imshow(photo);

        faceDetector = vision.CascadeObjectDetector;
        bbox = step(faceDetector, photo);
        out3=imcrop(photo,bbox);
        out = insertObjectAnnotation(photo,'rectangle',bbox,'Face');
        axes(handles.detectedFace);
        imshow(out);

        axes(handles.CropFace);
        imshow(out3);


end


% --- Executes on button press in SaveImage.
function SaveImage_Callback(hObject, eventdata, handles)
global photo
if isempty(photo)~=0
    msgbox(sprintf('Please load image'),'Error','Error');
else
    if(size(photo, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
        photo=rgb2gray(photo);
    end

    photo=imresize(photo,[128,128]);
    [FILENAME, PATHNAME, user_canceled] = uiputfile('*.jpg;*.png','save image');
    if (user_canceled)==1
```

```
        imwrite(photo,[PATHNAME FILENAME],'jpg');
    else
        msgbox(sprintf('Photo didnt save!'),'Error','Error');
    end
end


% --- Executes on button press in RecognizeFace.
function RecognizeFace_Callback(hObject, eventdata, handles)
val = get(handles.RecognitionMethod,'Value');
global photo;
if isempty(photo)~=0
    msgbox(sprintf('Please load image'),'Error','Error');
else

    switch val

        case 1    %corr2 method
            recog_corr2(photo);

        case 2     %histogram method
            recog_histogram(photo);

        case 3     %ssim method
            recog_ssim(photo);
    end

end

%--- Executes on selection change in InputMethod.
function InputMethod_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function InputMethod_CreateFcn(hObject, eventdata, handles)
% hObject    handle to InputMethod (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

το γραφικό περιβάλλον που χρησιμοποιήσαμε:
gui.fig



function ssim.m :

```
function [mssim, ssim_map] = ssim(img1, img2, K, window, L)

%========================================================================
%SSIM Index, Version 1.0
%Copyright(c) 2003 Zhou Wang
%All Rights Reserved.
%
%This is an implementation of the algorithm for calculating the
%Structural SIMilarity (SSIM) index between two images. Please refer
%to the following paper:
%
%Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image
%quality assessment: From error visibility to structural similarity"
%IEEE Transactios on Image Processing, vol. 13, no. 4, pp.600-612,
%Apr. 2004.
%
```

```
%Kindly report any suggestions or corrections to zhouwang@ieee.org
%
%----------------------------------------------------------------------
%
%Input : (1) img1: the first image being compared
%       (2) img2: the second image being compared
%       (3) K: constants in the SSIM index formula (see the above
%           reference). defualt value: K = [0.01 0.03]
%       (4) window: local window for statistics (see the above
%           reference). default widnow is Gaussian given by
%           window = fspecial('gaussian', 11, 1.5);
%       (5) L: dynamic range of the images. default: L = 255
%
%Output: (1) mssim: the mean SSIM index value between 2 images.
%           If one of the images being compared is regarded as
%           perfect quality, then mssim can be considered as the
%           quality measure of the other image.
%           If img1 = img2, then mssim = 1.
%       (2) ssim_map: the SSIM index map of the test image. The map
%           has a smaller size than the input images. The actual size:
%           size(img1) - size(window) + 1.
%
%Default Usage:
%   Given 2 test images img1 and img2, whose dynamic range is 0-255
%
%   [mssim ssim_map] = ssim_index(img1, img2);
%
%Advanced Usage:
%   User defined parameters. For example
%
%   K = [0.05 0.05];
%   window = ones(8);
%   L = 100;
%   [mssim ssim_map] = ssim_index(img1, img2, K, window, L);
%
%See the results:
%
%   mssim                        %Gives the mssim value
%   imshow(max(0, ssim_map).^4)  %Shows the SSIM index map
%
%======================================================================


if (nargin < 2 | nargin > 5)
   ssim_index = -Inf;
   ssim_map = -Inf;
   return;
end

if (size(img1) ~= size(img2))
   ssim_index = -Inf;
```

```
  ssim_map = -Inf;
  return;
end

[M N] = size(img1);

if (nargin == 2)
  if ((M < 11) | (N < 11))   % ?????
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  window = fspecial('gaussian', 11, 1.5);       % ???1.511*11????
  K(1) = 0.01;                                                  % default settings
  K(2) = 0.03;                                                   %
  L = 255;                                  %
end

if (nargin == 3)
  if ((M < 11) | (N < 11))
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  window = fspecial('gaussian', 11, 1.5);
  L = 255;
  if (length(K) == 2)
    if (K(1) < 0 | K(2) < 0)
              ssim_index = -Inf;
              ssim_map = -Inf;
              return;
    end
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end

if (nargin == 4)
  [H W] = size(window);
  if ((H*W) < 4 | (H > M) | (W > N))
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  L = 255;
  if (length(K) == 2)
    if (K(1) < 0 | K(2) < 0)
              ssim_index = -Inf;
              ssim_map = -Inf;
```

```
            return;
    end
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end

if (nargin == 5)
  [H W] = size(window);
  if ((H*W) < 4 | (H > M) | (W > N))
        ssim_index = -Inf;
        ssim_map = -Inf;
    return
  end
  if (length(K) == 2)
    if (K(1) < 0 | K(2) < 0)
            ssim_index = -Inf;
            ssim_map = -Inf;
            return;
    end
  else
        ssim_index = -Inf;
        ssim_map = -Inf;
        return;
  end
end
%%
C1 = (K(1)*L)^2;    % C1Lxy?
C2 = (K(2)*L)^2;    % C2??Cxy?
window = window/sum(sum(window)); %??
img1 = double(img1);
img2 = double(img2);

mu1   = filter2(window, img1, 'valid');  % ????
mu2   = filter2(window, img2, 'valid');  % ????

mu1_sq = mu1.*mu1;     % Ux??
mu2_sq = mu2.*mu2;     % Uy??
mu1_mu2 = mu1.*mu2;    % Ux*Uy?

sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;  % sigmax ??
sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;  % sigmay ??
sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;   % sigmaxy??

if (C1 > 0 & C2 > 0)
  ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq + C1).*(sigma1_sq +
sigma2_sq + C2));
else
  numerator1 = 2*mu1_mu2 + C1;
```

```
    numerator2 = 2*sigma12 + C2;
    denominator1 = mu1_sq + mu2_sq + C1;
    denominator2 = sigma1_sq + sigma2_sq + C2;
    ssim_map = ones(size(mu1));
    index = (denominator1.*denominator2 > 0);
    ssim_map(index) =
(numerator1(index).*numerator2(index))./(denominator1(index).*denominator2(index));
    index = (denominator1 ~= 0) & (denominator2 == 0);
    ssim_map(index) = numerator1(index)./denominator1(index);
end

mssim = mean2(ssim_map);

return
```

function simil.m :

```
function [ msim ] = simil( Im1,Im2 )

faceDetector = vision.CascadeObjectDetector;

bboxes = step(faceDetector, Im1);
out1=imcrop(Im1,bboxes);
out1=imresize(out1,[128,128]);
boxes = step(faceDetector, Im2);
out2=imcrop(Im2,boxes);
out2=imresize(out2,[128,128]);

K = [0.05 0.05];
window = ones(8);
L = 100;
[msim, sim_map] = ssim(out1, out2, K, window, L);

fprintf('error=%f\n\n',msim);

end
```

function sim_hist.m :

```
function [ cal ] = sim_hist( Im1,Im2 )

faceDetector = vision.CascadeObjectDetector;

bboxes = step(faceDetector, Im1);
out1=imcrop(Im1,bboxes);
out1=imresize(out1,[128,128]);

boxes = step(faceDetector, Im2);
out2=imcrop(Im2,boxes);
out2=imresize(out2,[128,128]);

cal=err(out1,out2);
fprintf('error=%f\n\n',cal);

end
```

function recog_ssim.m :

```
function recog_ssim( in )   %structural similarity (ssim)

fprintf('\n*****************************\nssim method\n');

if(size(in, 3)) == 3   %edw tsekarw an eikona mou einai grayscale or coloured
   fprintf('arxikh eikona pros anazhthsh = coloured\n');
   in=rgb2gray(in);
else
   fprintf('arxikh eikona pros anazhthsh = grayscale\n');
end

in=imresize(in,[128,128]);
[row, column, ss]=size(in);
n=count_images(  );
fprintf('\nno of images:%d\n\n',n);

if n==0
   msgbox(' Database is empty ');
   return;
else

   fi='C:\Users\Ioannis\Desktop\FACES\';
   la='.jpg';
```

```
for i=1:n  % edw mpainei se ena loop opou psaxnw na kanw tautopoihsh me ta proswpa
   reference=[fi int2str(i) la]; %pou vriskontai sto fakelo FACES pou vrisketai
   reference=imread(reference); %sthn epifaneia ergasias

   if(size(reference, 3)) == 3    %edw tsekarw an eikona mou einai grayscale or coloured
      fprintf('eikona %d = coloured\n',i);
      reference=rgb2gray(reference);
   else
      fprintf('eikona %d = grayscale\n',i);
   end
   reference=imresize(reference,[128,128]);
   [row2 ,column2]=size(reference);
   if row ~= row2 || column ~= column2
      sprintf('epanalipsi:%d\n',i)
   end
   k(i)=simil(in,reference); %Ginetai sugrish twn 2 eikonwn kai mesw
   %ths sunarthshs sim_hist epistrefetai ston pinaka k mia timh
end
h=max(k);
for i=1:n  %Sugrinoume tis times tou k wste na paroume afth me to mikrotero sfalma
   if (k(i)==h)
      mid=i;
      break
   end
end
fprintf('\nMatchinng....with photo %d: \nh=%f\n',i, h);
mid=int2str(mid);
reference=[fi mid la];
reference=imread(reference);

[row2 ,column2]=size(reference);
if row ~= row2 || column ~= column2
   fprintf('mphke sto last elegxo h=%f\n',h);

   in = imresize(in, [128 ,128]);
   reference = imresize(reference, [128 ,128]);

   if(size(in, 3)) == 3   %edw tsekarw an eikona mou einai grayscale or coloured
      fprintf('telikh  in eikona pros anazhthsh = coloured\n');
      in=rgb2gray(in);
   else
      fprintf('telikh   in eikona pros anazhthsh = grayscale\n');
   end

   if(size(reference, 3)) == 3   %edw tsekarw an eikona mou einai grayscale or coloured
      fprintf('telikh  reference eikona pros anazhthsh = coloured\n');
      reference=rgb2gray(reference);

   else
      fprintf('telikh   reference eikona pros anazhthsh = grayscale\n');
```

```
        end

    end

    if (h>0.4 && h<1.1)
    pairOfImages = [reference in];
    figure,     imshow(pairOfImages);

    else
        msgbox(' FACE NOT DETECTED ');
        fprintf('!!!! FACE NOT DETECTED !!!!!\n')
    end

end
```

function recog_histogram.m :

```
function recog_histogram( in )    %histogram method

fprintf('\n**************************\nhistogram method\n');

if(size(in, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
    fprintf('arxikh eikona pros anazhthsh = coloured\n');
    in=rgb2gray(in);
else
    fprintf('arxikh eikona pros anazhthsh = grayscale\n');
end

in=imresize(in,[128,128]);
[row, column, ss]=size(in);
n=count_images(  );
fprintf('\nno of images:%d\n\n',n);

if n==0
    msgbox(' Database is empty ');
    return;
else

    fi='C:\Users\Ioannis\Desktop\FACES\';
    la='.jpg';
    for i=1:n  % edw mpainei se ena loop opou psaxnw na kanw tautopoihsh me ta proswpa
        reference=[fi int2str(i) la]; %pou vriskontai sto fakelo FACES pou vrisketai
        reference=imread(reference); %sthn epifaneia ergasias

        if(size(reference, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
```

```
      fprintf('eikona %d = coloured\n',i);
      reference=rgb2gray(reference);
   else
      fprintf('eikona %d = grayscale\n',i);
   end

   reference=imresize(reference,[128,128]);

   [row2 ,column2]=size(reference);
   if row ~= row2 || column ~= column2
      sprintf('epanalipsi:%d\n',i)
   end
   k(i)=sim_hist(in,reference); %Ginetai sugrish twn 2 eikonwn kai mesw
   %ths sunarthshs sim_hist epistrefetai ston pinaka k mia timh
end
h=min(k);
for i=1:n  %Sugrinoume tis times tou k wste na paroume afth me to mikrotero sfalma
   if (k(i)==h)
      mid=i;
      break
   end
end
fprintf('\nMatchinng....with photo %d: \nh=%f\n',i, h);
mid=int2str(mid);
reference=[fi mid la];
reference=imread(reference);

[row2 ,column2]=size(reference);
if row ~= row2 || column ~= column2
   fprintf('mphke sto last elegxo h=%f\n',h);
   in = imresize(in, [128 ,128]);
   reference = imresize(reference, [128 ,128]);

   if(size(in, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
      fprintf('telikh  in eikona pros anazhthsh = coloured\n');
      in=rgb2gray(in);
   else
      fprintf('telikh   in eikona pros anazhthsh = grayscale\n');
   end

   if(size(reference, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
      fprintf('telikh  reference eikona pros anazhthsh = coloured\n');
      reference=rgb2gray(reference);

   else
      fprintf('telikh   reference eikona pros anazhthsh = grayscale\n');
   end

end
```

```
    if (h<0.07)
    pairOfImages = [reference in];
    figure,     imshow(pairOfImages);
    else
       msgbox(' FACE NOT DETECTED ');
       fprintf('!!!! FACE NOT DETECTED !!!!!\n')
    end

end
```

function recog_corr2.m :

```
function recog_corr2( in )   %2-d correlation coefficient method

fprintf('\n*****************************\n2-d correlation coefficient method\n');

% h eikona in einai h eikona pou psaxnw na kanw tautopoihsh

if(size(in, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
    fprintf('arxikh eikona pros anazhthsh = coloured\n');
    in=rgb2gray(in);
else
    fprintf('arxikh eikona pros anazhthsh = grayscale\n');
end

in=imresize(in,[128,128]);
[row, column, ss]=size(in);
n=count_images(  );
fprintf('\nno of images:%d\n\n',n);

if n==0
    msgbox(' Database is empty ');
    return;
else
    fi='C:\Users\Ioannis\Desktop\FACES\';
    la='.jpg';
    for i=1:n      % edw mpainei se ena loop opou psaxnw na kanw tautopoihsh me ta proswpa
       reference=[fi int2str(i) la];  %pou vriskontai sto fakelo FACES pou vrisketai
       reference=imread(reference);   %sthn epifaneia ergasias

       if(size(reference, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
          fprintf('eikona %d = coloured\n',i);
          reference=rgb2gray(reference);
       else
```

```
      fprintf('eikona %d = grayscale\n',i);
   end

   reference=imresize(reference,[128,128]);
   [row2 ,column2]=size(reference);
   if row ~= row2 || column ~= column2
      fprintf('den einai idies oi diastaseis twn eikonwn sthn epanalhpsh:%d\n',i);
   end
   k(i)=corr_error(in,reference); %Ginetai sugrish twn 2 eikonwn kai mesw
   %ths sunarthshs sim_hist epistrefetai ston pinaka k mia timh
end
h=max(k);
for i=1:n  %Sugrinoume tis times tou k wste na paroume afth me to mikrotero sfalma
   if (k(i)==h)
      mid=i;
      break
   end
end
fprintf('\nMatchinng....with photo %d: \nh=%f\n',i, h);
mid=int2str(mid);
reference=[fi mid la];
reference=imread(reference);
[row2 ,column2]=size(reference);

if row ~= row2 || column ~= column2
   fprintf('mphke sto last elegxo h=%f\n',h);
   in = imresize(in, [128 ,128]);
   reference = imresize(reference, [128 ,128]);

   if(size(in, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
      fprintf('telikh  in eikona pros anazhthsh = coloured\n');
      in=rgb2gray(in);
   else
      fprintf('telikh   in eikona pros anazhthsh = grayscale\n');
   end

   if(size(reference, 3)) == 3   %elegxw an h eikona mou einai grayscale or coloured
      fprintf('telikh  reference eikona pros anazhthsh = coloured\n');
      reference=rgb2gray(reference);

   else
      fprintf('telikh   reference eikona pros anazhthsh = grayscale\n');
   end

end

if (h>0.4)
pairOfImages = [reference in];
figure,     imshow(pairOfImages);
else
   msgbox(' FACE NOT DETECTED ');
```

```
      fprintf('!!!! FACE NOT DETECTED !!!!!\n')
   end
end
```

function err.m :

```
function [ d ] = err(i1,i2)

i1=i1(:,:,1);
[c1,n]=imhist(i1);
c1=c1/size(i1,1)/size(i1,2);

i2=i2(:,:,1);
[c2,n2]=imhist(i2);
c2=c2/size(i2,1)/size(i2,2);
d = pdist2(c1',c2');


end
```

function count_images.m :

```
function no=count_images(  )
ff=dir('C:\Users\Ioannis\Desktop\FACES');
n=length(ff)-2;
no=n;

end
```

function corr_error.m:

```
function [ cal ] = corr_error( Im1,Im2 )

faceDetector = vision.CascadeObjectDetector;

bboxes = step(faceDetector, Im1);
out1=imcrop(Im1,bboxes);
out1=imresize(out1,[128,128]);

boxes = step(faceDetector, Im2);
out2=imcrop(Im2,boxes);
out2=imresize(out2,[128,128]);

cal=corr2(out1,out2);
fprintf('error=%f\n\n',cal);

end
```

Βιβλιογραφία / Αναφορές

Γ. Πασσαλής. *Μέθοδοι Γραφικών και Τεχνητής Όρασης για την Ανακατασκευή, Αναπαράστασηκαι Ανάκτηση Τρισδιάστατων Αντικειμένων με Εφαρμογή στην Βιομετρία.* Διδακτορική Διατριβή, ΕΚΠΑ 2007.

G. Bradski, A. Kaehler and V. Pisarevsky. Learning-Based Computer Vision with Intel's Open Source Computer Vision Library. In *Intel Technology Journal,* Vol. 09-Iss.02, May 2005

Otsu , A threshold selection method from Gray – Level Histograms , IEEE Transactions on Systems, Man and Cybernetics 1979

D. Cristinacce and T. Cootes. Facial Feature Detection using Adaboost with Shape Constraints. In Proc. 14th British Machine Vision Conference, 2003.

K. Derpanis. The Harris Corner Detector. Technical Report, York University, 2004.

I. Dryden and K. Mardia. Statistical Shape Analysis, Wiley 1998.

P.E Danielsson, Euclidean Distance Mapping ,Computer Graphics and Image Processing 1980

P. Viola , M. Jones. Robust real-time face detection , International Journal of Computer Viion May 2004

Pantic, M. and Rothkrantz L. J. M., *"Expert system for automatic analysis of facial expressions"*, Image Vis. Comput., vol. 18, no. 11, pp. 881905, Aug.2000.

Rowley, H.A. and Baluja, S. and Kanade, T., *"Neural networkbased face detection"*, Pattern Analysis and Machine Intelligence, IEEE Transactions on Vol. 20, pages 2338, 1998.

S. Theodoridis, K. Koutroumbas. *Pattern Recognition*, 3d Ed. Academic Press 2006.

P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features.
In *Proceedings of IEEE Computer Sociefy Conference on Computer Vision and Pattern Recognition,*
2001.

H. Yang, D. Kriegman, and N. Ahuja. Detecting Faces in Images: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2002.

F. Zuo and P. With. Fast Facial Feature Extraction Using a Deformable Shape Model with Haar-Wavelet Based Local Texture Attributes. In *Int. Proc. on Image Processing,* 2004.

https://www.mathworks.com/help/vision/examples/face-detection-and-tracking-using-camshift.html

https://www.mathworks.com/help/images/ref/ssim.html

https://www.mathworks.com/help/images/ref/corr2.html?s_tid=srchtitle

https://www.mathworks.com/help/images/ref/imhist.html?searchHighlight=imhist

https://www.mathworks.com/help/stats/pdist2.html

https://www.mathworks.com/matlabcentral/answers/85447-comparison-of-two-histograms-using-pdist2