

ΜΑΘΗΜΑ: ΟΡΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ
Εξάμηνο: 8ο, Σχολή Ηλεκτρολόγων Μηχ. & Μηχ. Υπολογιστών ΕΜΠ
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2
Εκτίμηση Οπτικής Ροής (Optical Flow), Εξαγωγή Χαρακτηριστικών σε Βίντεο για Αναγνώριση Δράσεων, Συνένωση Εικόνων (Image Stitching)

Μέρος 1: Παρακολούθηση Προσώπου και Χεριών με Χρήση της Μεθόδου Οπτικής Ροής των Lucas-Kanade

Στο πρώτο μέρος της εργαστηριακής άσκησης έχουμε ως στόχο την υλοποίηση ενός συστήματος Face Tracking σε ένα βίντεο νοηματικής γλώσσας, ανιχνεύοντας στο πρώτο frame του βίντεο την περιοχή του προσώπου με πιθανοτικό ανιχνευτή ανθρώπινου δέρματος και ύστερα θα παρακολουθούμε το πλαίσιο αυτό μέσω του διανυσματικού πεδίου οπτικής ροής (αλγόριθμος Lucas-Kanade).

1.1 Ανίχνευση Δέρματος Προσώπου και Χεριών

Στο πρώτο ερώτημα θέλουμε να ανιχνεύσουμε θέλουμε να ανιχνεύσουμε την περιοχή του προσώπου στο πρώτο frame του βίντεο. Για να το επιτύχουμε αυτό αρχικά μεταφέρουμε την εικόνα στον YCbCr χρωματικό χώρο και αδιαφορούμε για την πληροφορία της φωτεινότητας που βρίσκεται στο Y κανάλι και επικεντρωνόμαστε στα κανάλια Cb και Cr που περιγράφουν την ταυτότητα του χρώματος. Το χρώμα του δέρματος μοντελοποιείται με δισδιάστατη Γκαουσιανή κατανομή ως εξής:

$$P(c = \text{skin}) = \frac{1}{\sqrt{|\Sigma|(2\pi)^2}} e^{-\frac{1}{2}(c-\mu)^T \Sigma^{-1} (c-\mu)} \quad (1)$$

Όπου:

c το 1x2 διάνυσμα τιμών Cb και Cr για καθε pixel

Σ ο 2x2 πίνακας συνδιακύμανσης των τιμών Cb και Cr

μ το 1x2 διάνυσμα των μέσων τιμών Cb και Cr

Για την εκπαίδευση της γκαουσιανής κατανομής χρησιμοποιήσαμε δείγματα δέρματος που δίνονται στο αρχείο skinSamplesRGB.mat.

Με βάση τα παραπάνω και με χρήση της python και των βιβλιοθηκών της opencv, matplotlib και numpy καταφέρνουμε να διαβάσουμε τα δείγματα δέρματος και έπειτα να υπολογίσουμε τη μέση τιμή και την συνδιακύμανση.

Οι μετρήσεις που παρθηκαν εδωσαν τα εξής αποτελέσματα:

$\mu = [103.27048260381594, 157.0460157126824]$

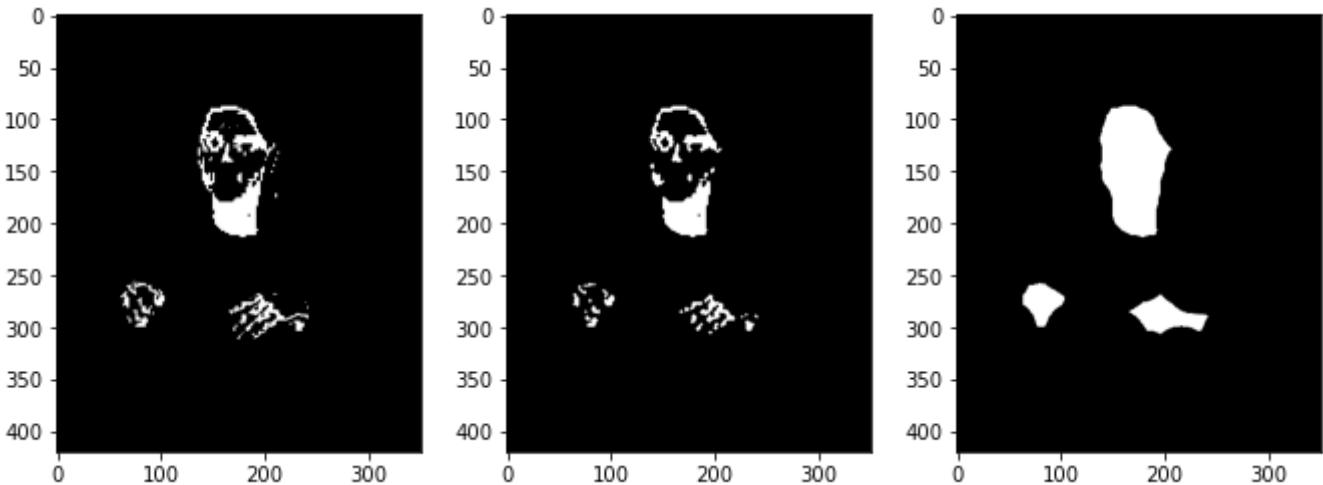
$\Sigma = [[44.19103128 -11.9310385] [-11.9310385 11.19574811]]$

Επειτα, διαβάσαμε το πρώτο frame του βίντεο που ζητείται να αναλύσουμε και υπολογίσαμε την πιθανότητα που έχει κάθε πίξελ να είναι αποτελεί δέρμα. Στη συνέχεια κανονικοποιήσαμε την εικόνα πολλαπλασιάζοντας με την μέγιστη τιμή πιθανότητας που θα μπορούσε να προκύψει. Η τιμή αυτή υπολογίζεται αν υποθέσουμε ότι ο εκθέτης του ε μηδενίζεται.

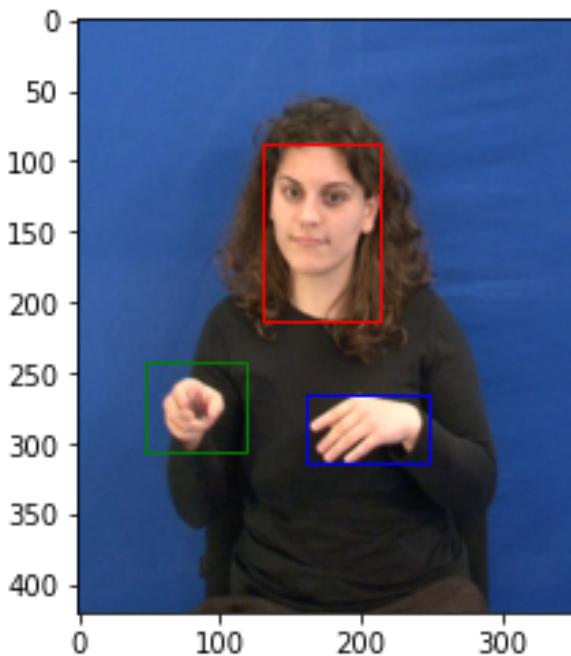
Τέλος, προκύπτει η δυαδική εικόνα με χρήση κατωφλιοποίησης. Για την κατωφλιοποίηση χρησιμοποιήσαμε την τιμή 0.05 την οποία θεωρήσαμε και ιδανική για την καλύτερη εξαγωγή των 3 περιοχών. Όμως, για την ακριβή ανίχνευση των χεριών και του προσώπου πρέπει να

δημιουργήσουμε 3 ενιαίες ανεξάρτητες περιοχές. Για τον σκοπό αυτό απαιτείται μια μορφολογική επεξεργασία της δυαδικής εικόνας. Αρχικά, θα κάνουμε opening με ένα μικρό δομικό στοιχείο ώστε να εξαλειφθούν οι πολύ μικρές μεμονωμένες περιοχές και έπειτα closing με ένα μεγάλο δομικό στοιχείο με σκοπό να καλύψουμε τις τρύπες και να αποκτήσουν συνοχή οι προκύπτουσες περιοχές των χεριών και του προσώπου.

Παρακάτω φαίνονται τα ενδιάμεσα στάδια της επεξεργασίας μέχρι την τελική επιλογή των περιοχών του προσώπου και των χεριών.



Τελικά, έχοντας πια 3 συνεκτικές συνιστώσες χρησιμοποιώντας την συνάρτηση `label` καταφέρνουμε να τις κατηγοριοποιήσουμε και με τη συνάρτηση `where` να βρούμε τις συντεταγμένες των περιοχών αυτών. Λαμβάνοντας υπόψιν πλέον την αρχική εικόνα εφαρμόζουμε πάνω της τα 3 bounding boxes που βρήκαμε με συντεταγμένες: `head = [130, 88, 84, 125]`, `left = [48, 242, 71, 63]`, `right = [162, 265, 87, 49]` όπως βλέπουμε παρακάτω:



1.2.1 Υλοποίηση του Αλγορίθμου των Lucas-Kanade

Σε αυτό το σημείο καλούμαστε να υλοποιήσουμε τον αλγόριθμο των Lucas-Kanade, ο οποίος δεδομένων δύο διαδοχικών εικόνων, υπολογίζει το πεδίο οπτικής ροής $d = (dx, dy)$ φέρνοντας τες

σε αντιστοιχία. Προκύπτει ότι η λύση ελαχίστων τετραγώνων (με ανάπτυξη κατά Taylor) για την βελτίωση του πεδίου οπτικής ροής d σε κάθε σημείο είναι:

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} (G_\rho * A_1^2)(x) + \varepsilon & (G_\rho * (A_1 A_2))(x) \\ (G_\rho * (A_1 A_2))(x) & (G_\rho * A_2^2)(x) + \varepsilon \end{bmatrix}^{-1} \cdot \begin{bmatrix} (G_\rho * (A_1 E))(x) \\ (G_\rho * (A_2 E))(x) \end{bmatrix} \quad (2)$$

όπου:

$$A(x) = \begin{bmatrix} A_1(x) & A_2(x) \end{bmatrix} = \begin{bmatrix} \frac{\partial I_{n-1}(\mathbf{x} + \mathbf{d}_i)}{\partial x} & \frac{\partial I_{n-1}(\mathbf{x} + \mathbf{d}_i)}{\partial y} \end{bmatrix} \quad (3)$$

και

$$E(x) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_i) \quad (4)$$

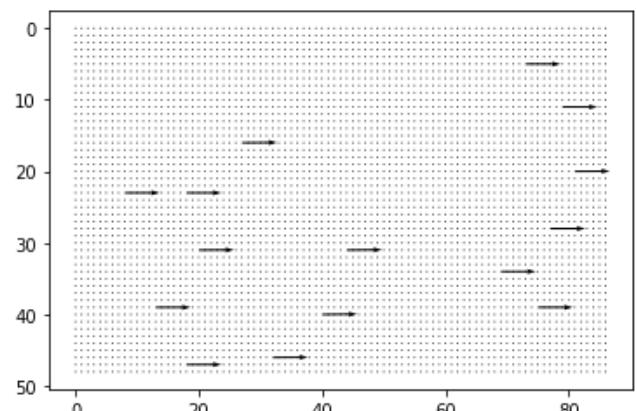
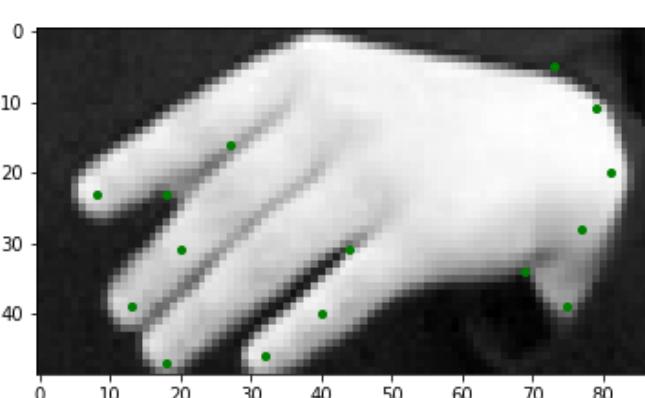
Η μικρή θετική σταθερά ε όπως θα συμπεράνουμε και παρακάτω πειραματικά βελτιώνει το αποτέλεσμα σε επίπεδες περιοχές με μειωμένη υφή, δηλαδή, μειωμένη πληροφορία για τον υπολογισμό της οπτικής ροής. Η ανανέωση του διανύσματος οπτικής ροής επαναλαμβάνεται αρκετές φορές έως ότου συγκλίνει σε κάποια τιμή.

Αρχικά ορίζουμε τις δύο εικόνες οι οποίες θα χρησιμοποιηθούν για την εύρεση της οπτικής ροής. Επιπλέον, υπολογίζουμε την πρώτη παράγωγο ως προς x και ως προς y της πρώτης εικόνας και την διδιάστατη γκαουσιανή που θα χρησιμοποιήσουμε στην συνέχεια για τον υπολογισμό της βελτίωσης u . Έπειτα, υπολογίζουμε τα σημεία ενδιαφέροντος της 2ης εικόνας. Για τον σκοπό αυτό χρησιμοποιούμε την συνάρτηση `goodFeaturesToTrack` η οποία υλοποιεί την μέθοδο των Shi και Tomashi. Για κάθε σημείο ενδιαφέροντος που προκύπτει από την παραπάνω συνάρτηση υλοποιούμε τον αλγόριθμο Lucas-Kanade. Συγκεκριμένα, απομονώνουμε μια μικρή περιοχή γύρω από το σημείο ενδιαφέροντος με μέγεθος ίσο με αυτό του γκαουσιανού πυρήνα.

Ένα σημαντικό σημείο στην υλοποίηση που πρέπει να προσέξουμε είναι ότι κατά τη διάρκεια των επαναλήψεων πολλές φορές θα χρειαστούμε τιμές της εικόνας I_{n-1} και των μερικών παραγώγων της σε ενδιάμεσα σημεία του πλέγματος αφού οι προβλέψεις μας και η βελτίωση που προκύπτει δεν έχουν ακέραιες τιμές. Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε την `map_coordinates`. Επομένως, με βάση τις παραπάνω σχέσεις και την διαδικασία που περιγράψαμε υπολογίζουμε τα A_1 , A_2 και E .

Έπειτα κάνοντας χρήση της γνωστής σχέσης του αντίστροφου πίνακα από την γραμμική άλγεβρα υπολογίζουμε το πρώτο μέρος του $u(x)$. Τέλος, απομένει ένας απλός πολλαπλασιασμός πινάκων που θα μας δώσει το τελικό αποτέλεσμα σε μία επανάληψη το οποίο προσθέτουμε στο προηγούμενο διάνυσμα οπτικής ροής d_{i-1} και προχωράμε στην επόμενη επανάληψη.

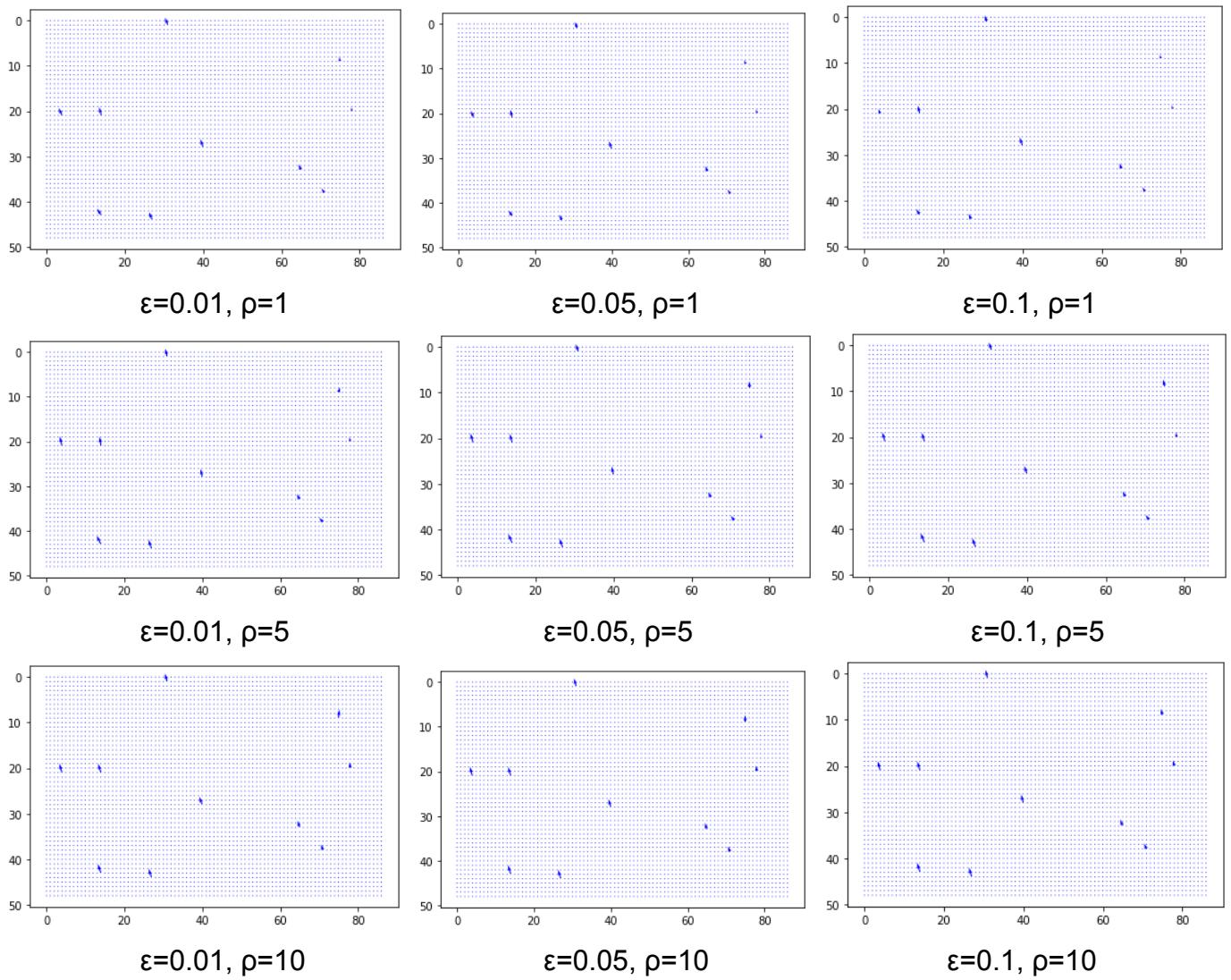
Αρχικά δοκιμάσαμε τον αλγόριθμό μας για μια μετακίνηση κατά 3 pixels στον άξονα x ώστε να επιβεβαιώσουμε ότι λειτουργεί σωστά. Πήραμε τα παρακάτω αποτελέσματα για το πεδίο οπτικής ροής και το bounding box για το δεξί χέρι.



Features detected for left hand

Optical flow for a 3 pixel displacement

Παρακάτω παρουσιάζουμε τα αποτελέσματα για το πεδίο οπτικής ροής μεταξύ του πρώτου και του δεύτερου frame της ακολουθίας GreekSignLanguage για διάφορες τιμές των παραμέτρων ρ και ϵ ώστε να κατανοήσουμε την επίδραση τους στην ροή.

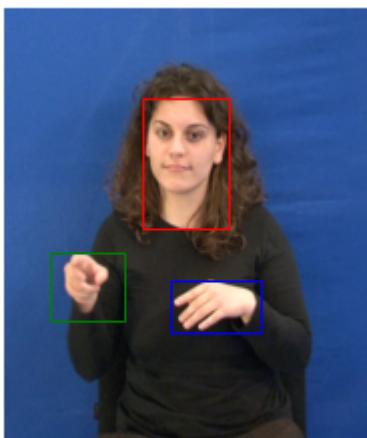


Αυτό που παρατηρούμε είναι ότι για μεγαλύτερες τιμές του ρ το πεδίο οπτικής εμφανίζει μεγαλύτερη ομοιομορφία και ομαλότητα και αυτό φαίνεται και από τις κατευθύνσεις των διανυσμάτων. Πειραματιστήκαμε και με μεγαλύτερες τιμές του ρ και παρατηρήσαμε ότι οι αλλαγές ύστερα είναι πολύ μικρές. Γι' αυτό παρατηρούμε όλα τα σημεία ενδιαφέροντος να έχουν σχεδόν την ίδια μετατόπιση. Η μικρή σταθερά ϵ γνωρίζουμε ότι βελτιώνει το αποτέλεσμα του υπολογισμού του πεδίου οπτικής ροής σε επίπεδες περιοχές με μειωμένη υφή. Όμως, διατηρώντας σταθερή την παράμετρο ρ βλέπουμε ότι με την αύξηση του ϵ το διανυσματικό πεδίο οπτικής ροής φαίνεται να δίνει μικρότερες μετατοπίσεις. Αυτό πιθανώς συμβαίνει γιατί ο αλγόριθμός μας δεν προλαβαίνει να συγκλίνει και τα υπολογισμένα διανύσματα ροής έχουν τελικά μικρότερες τιμές. Για μεγαλύτερο ϵ χρειάζονται περισσότερες επαναλήψεις για να φτάσουμε μια μικρή τιμή κατωφλίου του κριτηρίου συγκλισης.

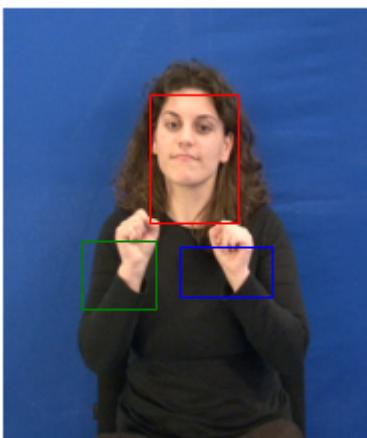
1.2.2 Υπολογισμός της Μετατόπισης των Παραθύρων από τα Διανύσματα Οπτικής

Έχοντας υπολογίσει την οπτική ροή της εικόνας ή στα σημεία ενδιαφέροντος σκοπός μας είναι να βρούμε το συνολικό διάνυσμα μετατόπισης του bounding box ώστε να μπορούμε να παρακολουθήσουμε την κίνηση των χεριών και του προσώπου. Για να το πετύχουμε αυτό θα υπολογίσουμε τον μέσο όρο των διανυσμάτων μετατόπισης των σημείων ενδιαφέροντος. Για να μειώσουμε το σφάλμα του απλού μέσου όρου, να πετύχουμε μεγαλύτερη ακρίβεια και να απορρίψουμε outliers μπορούμε να χρησιμοποιήσουμε την ενέργεια διανύσματος ταχύτητας, να ορίσουμε ένα κατώφλι ενέργειας κάτω από το οποίο δεν θα συμπεριλαμβάνουμε στο μέσο όρο το διάνυσμα οπτικής ροής.

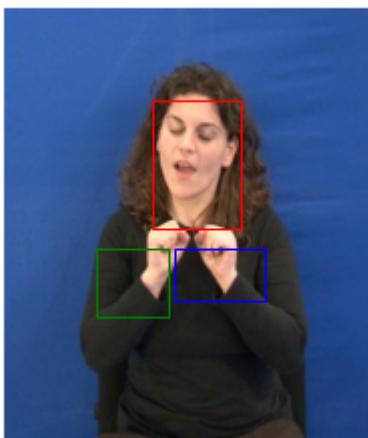
Τελικά, για διάφορες τιμές των ρ , ε έχουμε τις εξής μετατοπίσεις των bounding boxes:



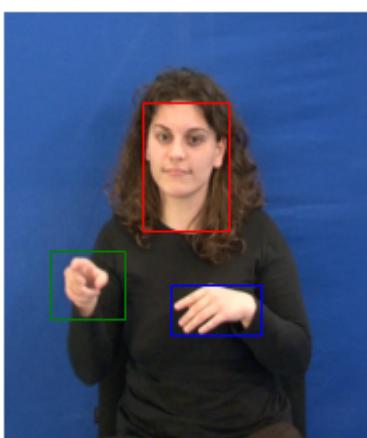
($\varepsilon=0.1, \rho=1$) 3



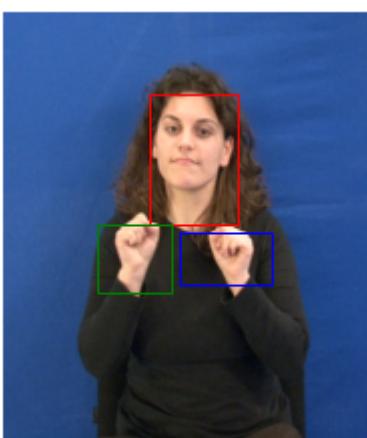
15



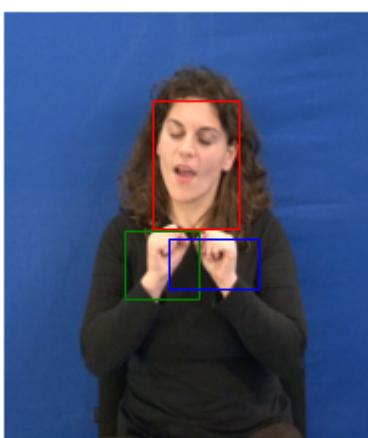
19



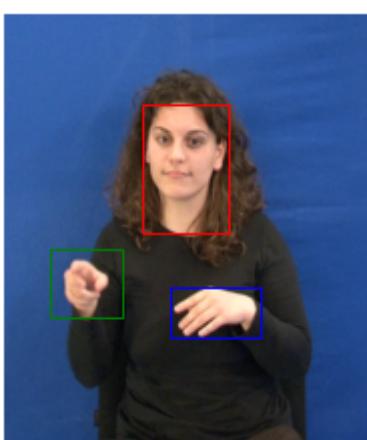
($\varepsilon=0.05, \rho=3$) 3



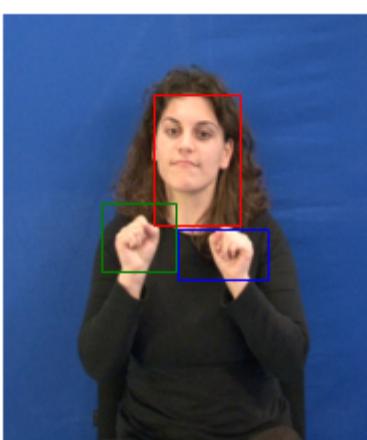
15



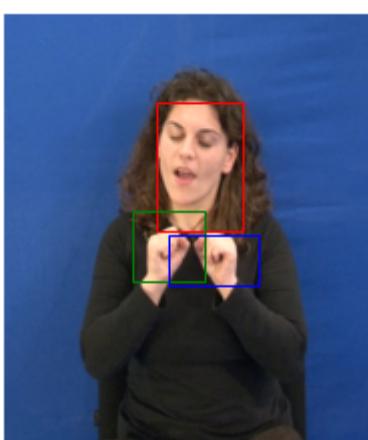
19



($\varepsilon=0.01, \rho=7$) 3



15



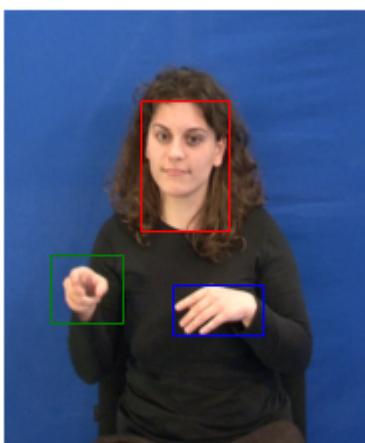
19

Οπως ειχαμε παρατηρήσει και στα διαγράμματα οπτικής ροής έτσι και εδώ βλέπουμε ότι όσο μεγαλύτερο είναι το ρ τόσο πιο ομαλές κινήσεις έχουμε. Γι' αυτό στην τελευταία γραμμή στο frame 19 το πράσινο box συνεχίζει να κινείται προς τα πάνω ενώ έπρεπε να κινηθεί και περισσότερο δεξιά. Παρόλα αυτά έχει προσεγγίσει αρκετά καλά την κίνηση του χεριού στον άξονα των y. Τέλος, είναι φανερό πως τη χειρότερη επίδοση έχουμε για $\rho = 1$.

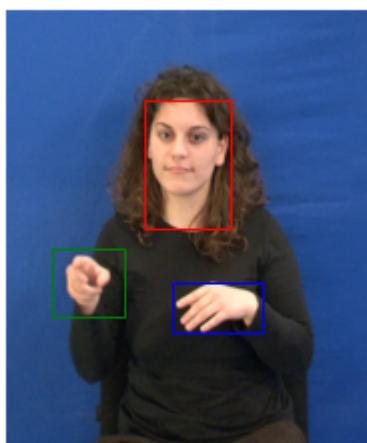
1.2.3 Πολυ-Κλιμακωτός Υπολογισμός Οπτικής Ροής

Στο ερώτημα αυτό θα υλοποιήσουμε την πολυκλιμακωτή εκδοχή του αλγορίθμου Lucas – Kanade. Ο αλγόριθμος αρχικά αναλύει τα δύο διαδοχικά frames σε Γκαουσιανές πυραμίδες, πραγματοποιώντας διαδοχικές υποδειγματοληψίες του αρχικού. Για τη μετάβαση από μεγάλες σε μικρές κλίμακες κατά την κατασκευή της Γκαουσιανής πυραμίδας φιλτράρουμε την εικόνα με ένα βαθυπερατό φίλτρο (χρησιμοποιήθηκε γκαουσιανή τυπικής απόκλισης 3 pixel) προκειμένου να μειωθεί η φασματική αναδίπλωση (aliasing) της εικόνας. Επίσης, κατά τη μετάβαση από τις μικρές στις μεγάλες κλίμακες το διάνυσμα d διπλασιάζεται, καθώς μετατόπιση 1 pixel σε μία κλίμακα συνεπάγεται μετατόπιση 2 pixel στην αμέσως μεγαλύτερη λόγω downsampling κατά 2.

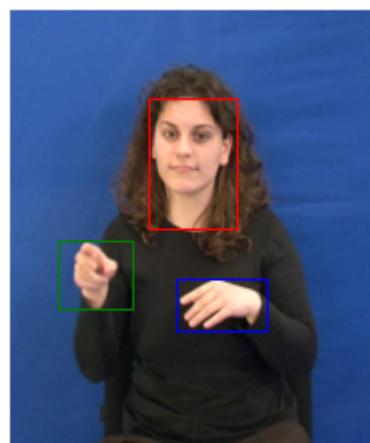
Κάτι που πρέπει να προσέξουμε στην υλοποιήση είναι ότι καθώς υποδειγματοληπτείται η εικόνα τα σημεία ενδιαφέροντος μειώνονται αισθητά συνεπώς πρέπει να κάνουμε πιο επιεική τα κριτήρια/παραμέτρους εύρεσης σημείων ενδιαφέροντος στην συνάρτηση goodFeaturesToTrack. Παρακάτω παρουσιάζουμε τα αποτελέσματα για τον πολυκλιμακωτό Lucas-Kanade πρώτα για παραμέτρους $\epsilon=0.05$ και $\rho=3$ και έπειτα για $\epsilon=0.01$ και $\rho=7$.



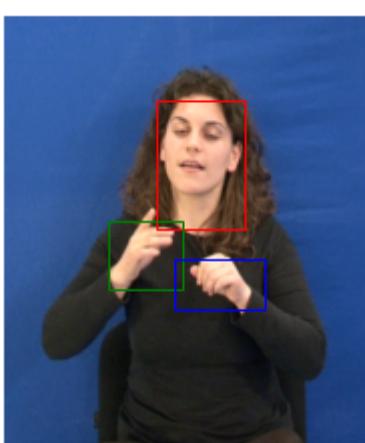
Πλαίσιο 2



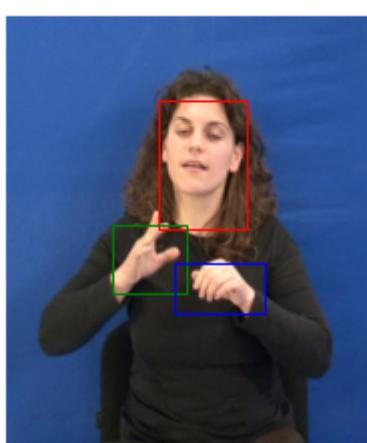
Πλαίσιο 3



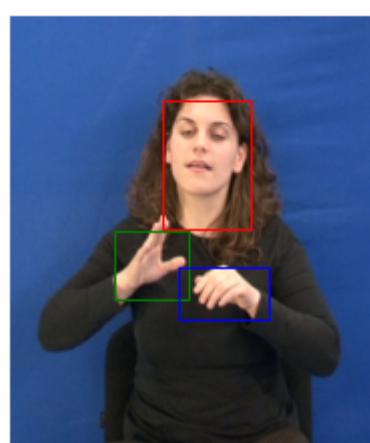
Πλαίσιο 4



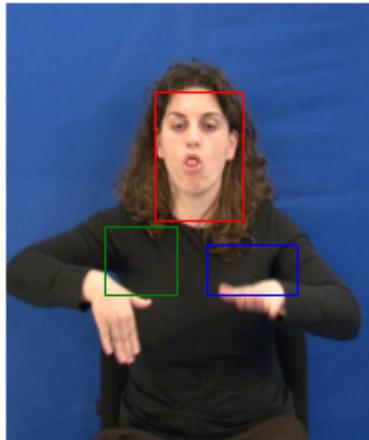
Πλαίσιο 27



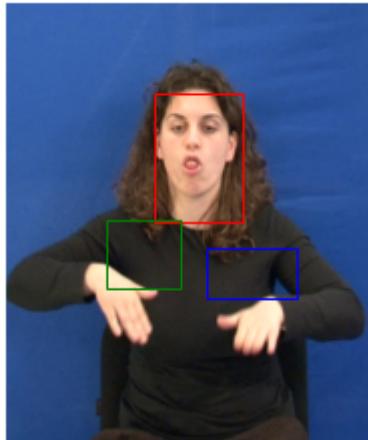
Πλαίσιο 28



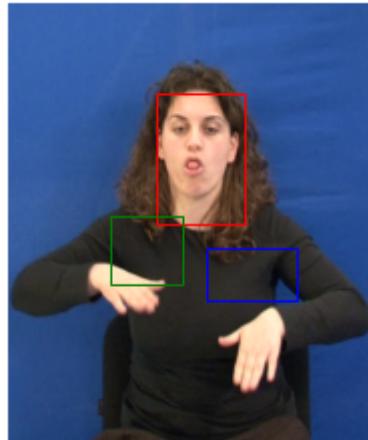
Πλαίσιο 29



Πλαίσιο 63

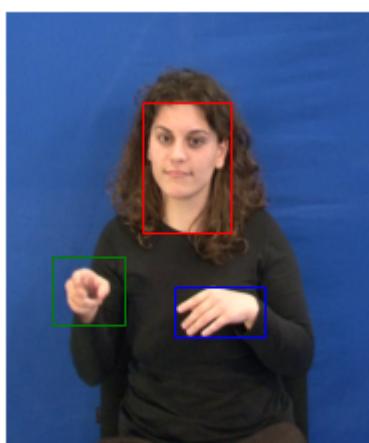


Πλαίσιο 64

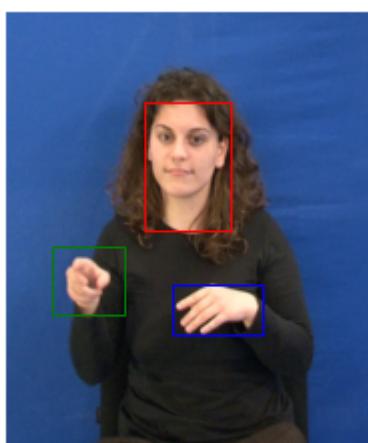


Πλαίσιο 65

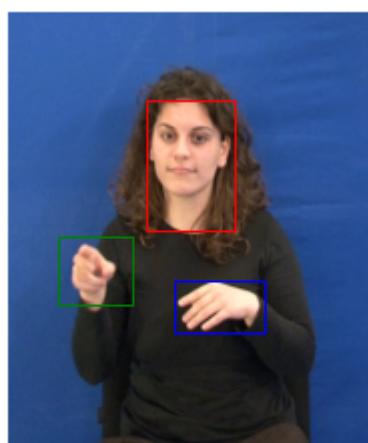
Πλαίσια από την αρχή, την μέση και το τέλος του βίντεο για $\epsilon=0.05$ και $\rho=3$



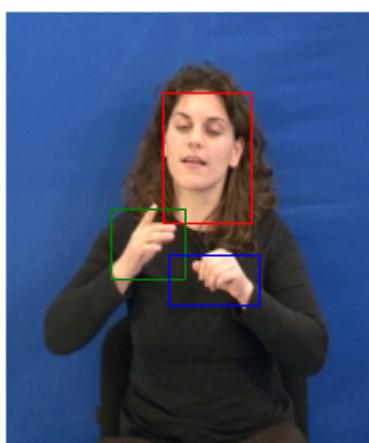
Πλαίσιο 2



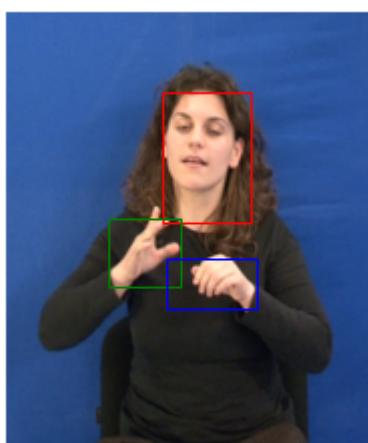
Πλαίσιο 3



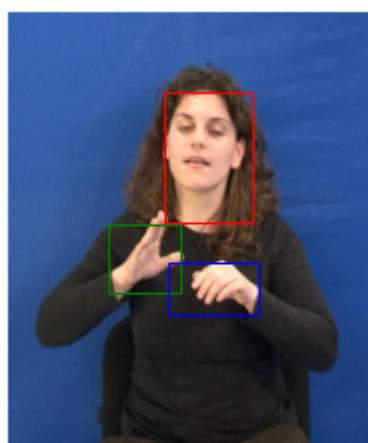
Πλαίσιο 4



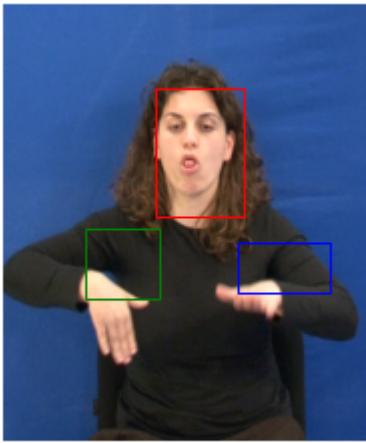
Πλαίσιο 27



Πλαίσιο 28

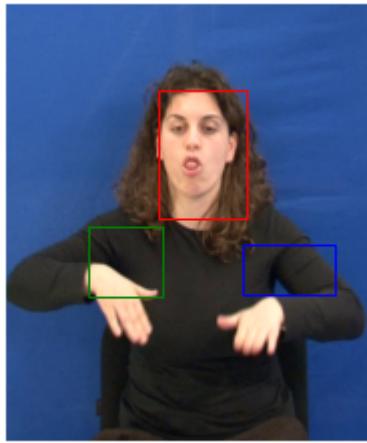


Πλαίσιο 29

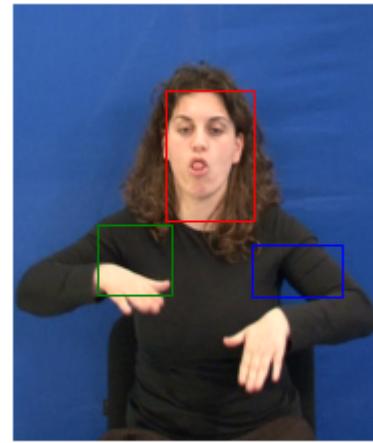


Πλαίσιο 63

Πλαίσια από την αρχή, την μέση και το τέλος του βίντεο για $\varepsilon=0.01$ και $\rho=7$



Πλαίσιο 64



Πλαίσιο 65

Παρατηρούμε πως για μεγαλύτερο ρέχουμε καλύτερη παρακολούθηση των περιοχών, καθώς στα τελευταία frames του βίντεο για $\rho=3$ φαίνεται πως χάνουμε το δεξί χέρι, ενώ για $\rho=7$ αν και αυτό έχει βγεί μερικώς εκτός του bounding box δεν έχει χαθεί από το σύστημα. Αυτό οφείλεται στο γεγονός πως για μεγαλύτερα ρέχουμε μεγαλύτερες περιοχές συνέλιξης και συνεπώς ακολουθούμε πιο μεγάλες περιοχές, σε αντίθεση με μικρότερα ρέπου περιορίζονται γύρω από τα σημεία ενδιαφέροντος. Έτσι είμαστε ικανοί να ακολουθούμε πιο απότομες κινήσεις.

Μέρος 2: Εντοπισμός Χωρο-χρονικών Σημείων Ενδιαφέροντος και Εξαγωγή Χαρακτηριστικών σε Βίντεο Ανθρωπίνων Δράσεων

Στο δεύτερο μέρος, διαθέτουμε τρεις κλάσεις δράσεων (boxing, running, walking), τις οποίες χρησιμοποιούμε για να κατηγοριοποιήσουμε τις ανθρώπινες δράσεις, χρησιμοποιώντας χωροχρονικά χαρακτηριστικά μέσω των ανιχνευτών Harris Stephens και Gabor. Τα χαρακτηριστικά αυτά είναι απαραίτητα για την εξαγωγή των χωροχρονικών περιγραφητών.

2.1 Χωρο-χρονικά Σημεία Ενδιαφέροντος

Αρχικά, υλοποιούμε τον Harris Stephens 3D detector, σχηματίζοντας έναν πίνακα M . Ο M προκύπτει προσθέτοντας το 3D γκαουσιανό πυρήνα ομαλοποίησης στις χωροχρονικές παραγώγους ως προς s και t . Αυτό το επιτυγχάνουμε φιλτράροντας δύο φορές τα στοιχεία του πίνακα των παραγώγων μεσω της συνάρτηση $cv2.filter2D$ με τη γκαουσιανή 2D που είχαμε φτιάξει στην 1η εργαστηριακή, εφαρμόζοντας συνέλιξη ως προς κάθε διάσταση. Υπολογίζοντας την ορίζουσα του M , εύκολα προκύπτει το 3D κριτήριο γωνιότητας.

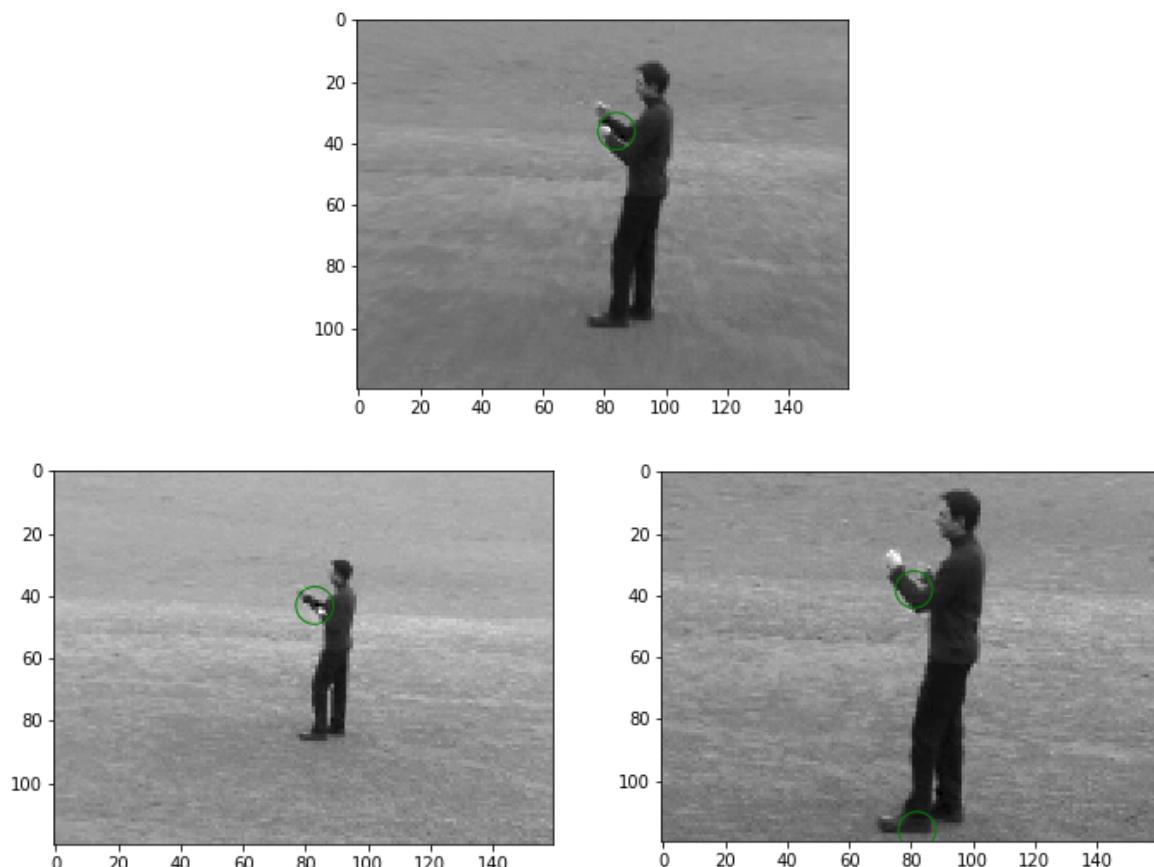
Στη συνέχεια, υλοποιούμε το Gabor detector. Στην αρχή συνελίσσουμε το βίντεο με το 2D γκαουσιανό πυρήνα, με στόχο την εξομάλυνσή του στις χωρικές διαστάσεις. Σχηματίζουμε το περιπτό και το άρτιο gabor φίλτρο, όπως ορίζονται οι συναρτήσεις και τα όριά τους από την εκφώνηση και καταλήγουμε στον υπολογισμό του 3D κριτηρίου γωνιότητας.

Αφού έχουμε ορίσει τον υπολογισμό των σημείων ενδιαφέροντος για τους δύο ανιχνευτές, προχωράμε στη δημιουργία ενός path, όπου αποθηκεύονται 100 frames για καθένα από τα 3 βίντεο που επιλέξαμε από τις κλάσεις δράσεων. Σε αυτά τα frames αποτυπώνονται και τα σημεία

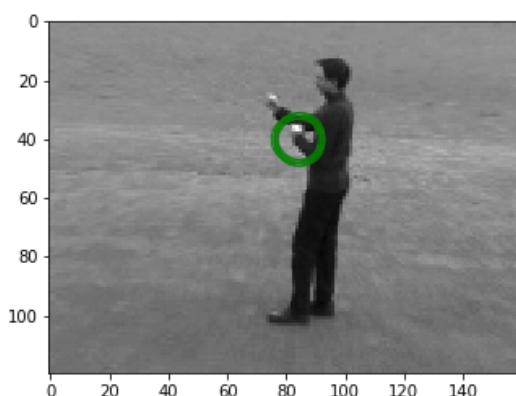
ενδιαφέροντος. Παρακάτω παρουσιάζουμε μερικά παραδείγματα frames με τα σημεία ενδιαφέροντος ώστε να συγκρίνουμε τη λειτουργία των δύο ανιχνευτών.

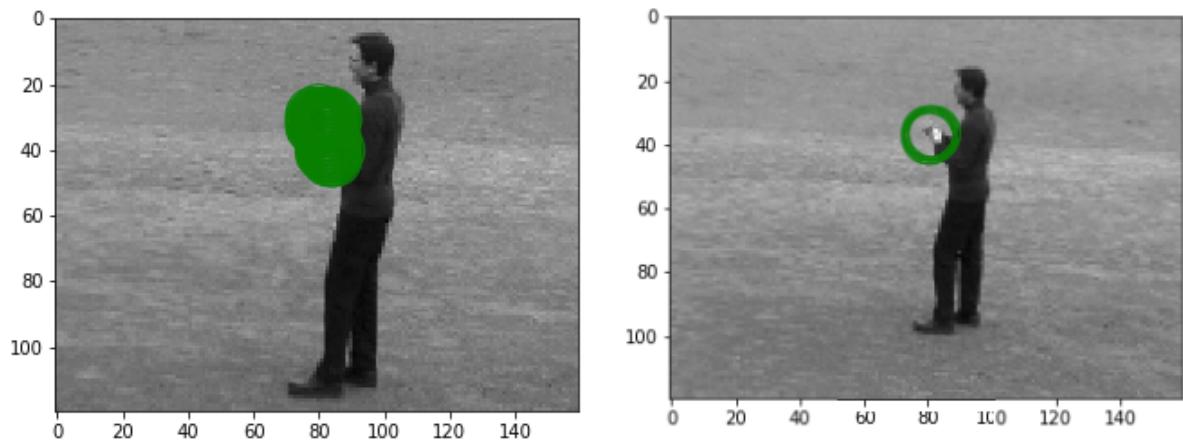
Παρατηρούμε ότι και οι δύο ανιχνευτές καταφέρνουν να εντοπίσουν κίνηση στα σημεία ενδιαφέροντος, δηλαδή στα άκρα του ανθρώπου, ιδίως όταν πρόκειται για μεγάλες μετατοπίσεις. Σημειώνεται, όμως, ότι ο Gabor detector διαθέτει περισσότερα πλαίσια στα σημεία ενδιαφέροντος, ενώ ο Harris detector πολλές φορές δεν τα εντοπίζει. Επίσης, ο Harris εντοπίζει ενίοτε σημεία που δε σχετίζονται με τα άκρα, ενώ τα πλαίσια του Gabor είναι πάντα πολλά και συγκεντρωμένα στα σημεία ενδιαφέροντος. Για αυτό το λόγο, ο Gabor detector είναι πιο ακριβής σε σχέση με το Harris.

Video: person01_boxing_d2_uncomp.avi



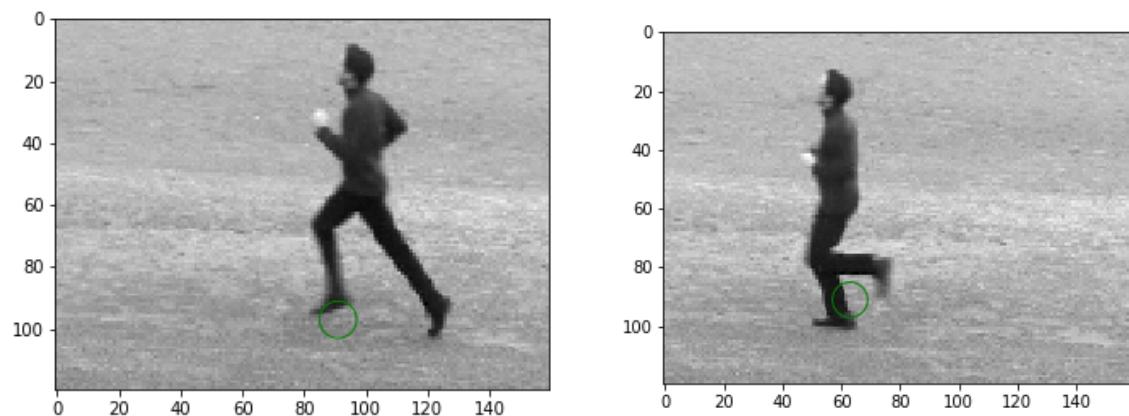
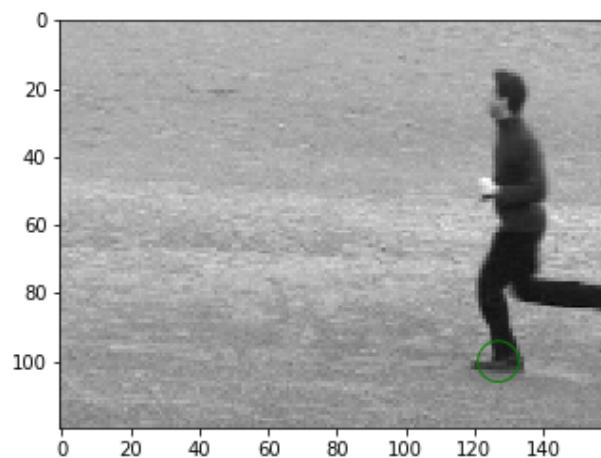
Harris Detector



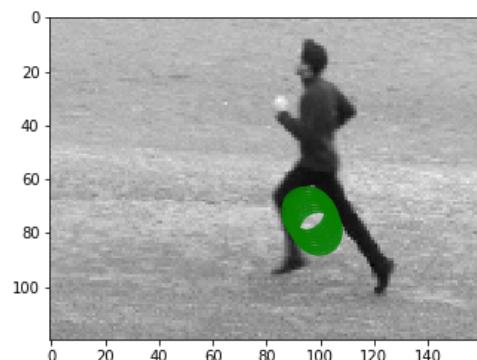


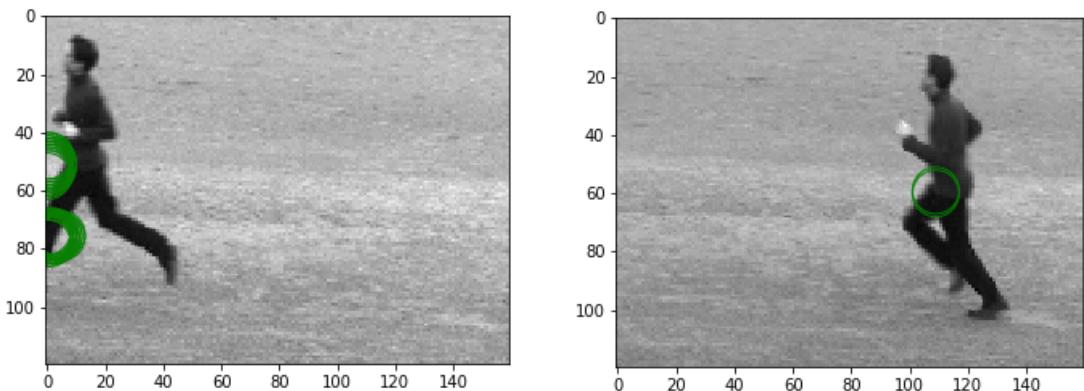
Gabor Detector

Video: person01_running_d1_uncomp.avi



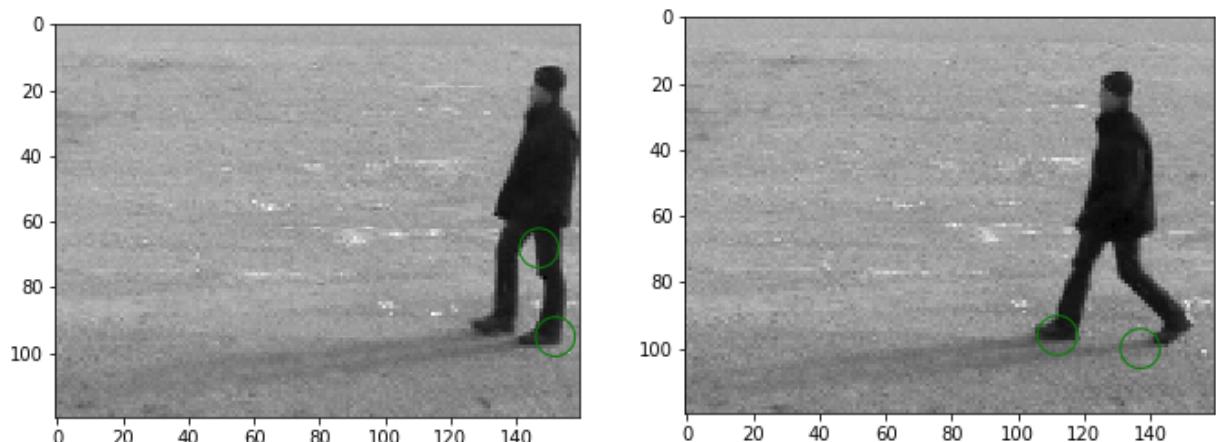
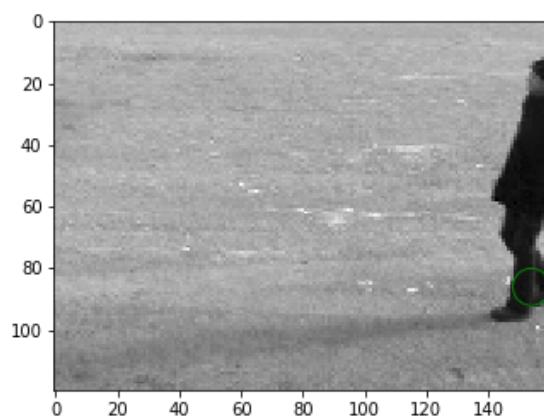
Harris Detector



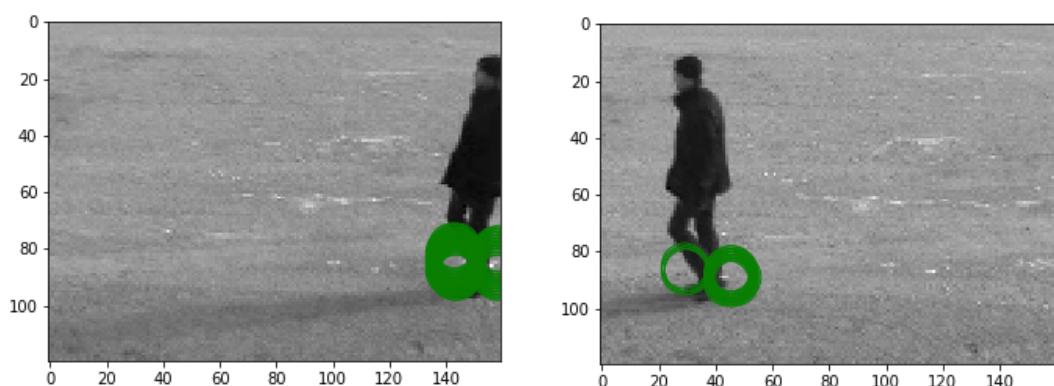


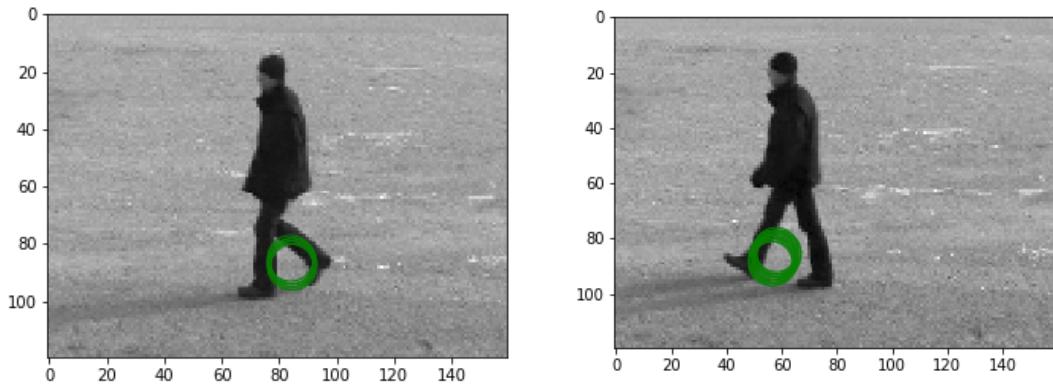
Gabor Detector

Video: person04_walking_d1_uncomp.avi



Harris Detector





Gabor Detector

2.2 Χωρο-χρονικοί Ιστογραφικοί Περιγραφητές

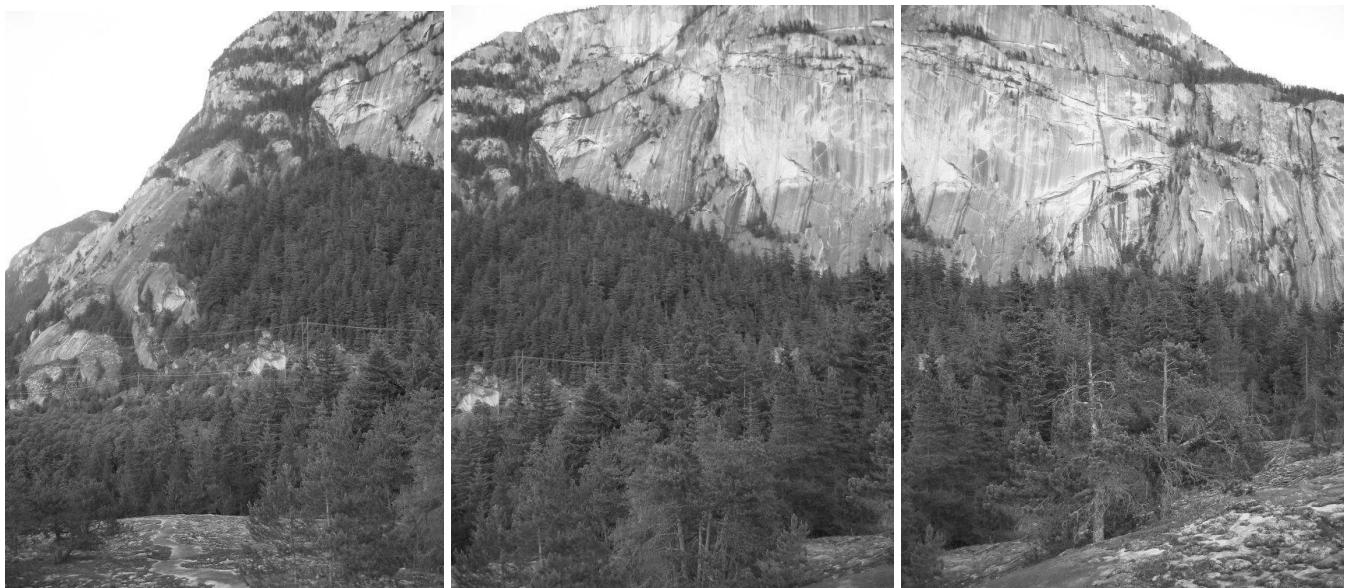
Έχοντας υπολογίσει τα σημεία ενδιαφέροντος, μπορούμε να υλοποιήσουμε τους χωροχρονικούς περιγραφητές. Η συνάρτηση `desc=orientation_histogram(Gx,Gy,nbins,np.array([n,m]))` υπολογίζει τους περιγραφητές και καλείται αφού έχουμε υπολογίσει τα ορίσματα του διανυσματικού πεδίου. Αν ο descriptor είναι ο HOG, υπολογίζεται το διάνυσμα κλίσης, αν είναι ο HOF υπολογίζεται η οπτική ροή για κάθε pixel, ενώ στην περίπτωση του HOG/HOF συνενώνουμε τους άλλους δύο περιγραφητές.

2.3: Κατασκευή Bag of Visual Words και χρήση Support Vector Machines για την ταξινόμηση δράσεων

Αρχικά, χωρίζουμε τα δεδομένα σε train και test, αποθηκεύοντας παράλληλα τα ονόματα των δεδομένων αυτών.

Μέρος 3: Συνένωση Εικόνων (Image Stitching) για Δημιουργία Πανοράματος

Για να επιτύχουμε πιο ακριβή αποτελέσματα, θα φορτώσουμε τις δύο εικόνες μας ως grayscale. Κάνουμε χρήση της συνάρτησης της cv2, της `cv2.SIFT_create(nfeatures=2000)` με argument 2000 ως ο μέγιστος αριθμός χαρακτηριστικών που θέλουμε να ανιχνευτούν. Για τον εντοπισμό των χαρακτηριστικών ενδιαφέροντος και την εξαγωγή των descriptors χρησιμοποιούμε την συνάρτηση `detectAndCompute`, ενώ για την αναπαράσταση αυτών κάνουμε χρήση της συνάρτησης `drawKeypoints`.



Βήμα 0





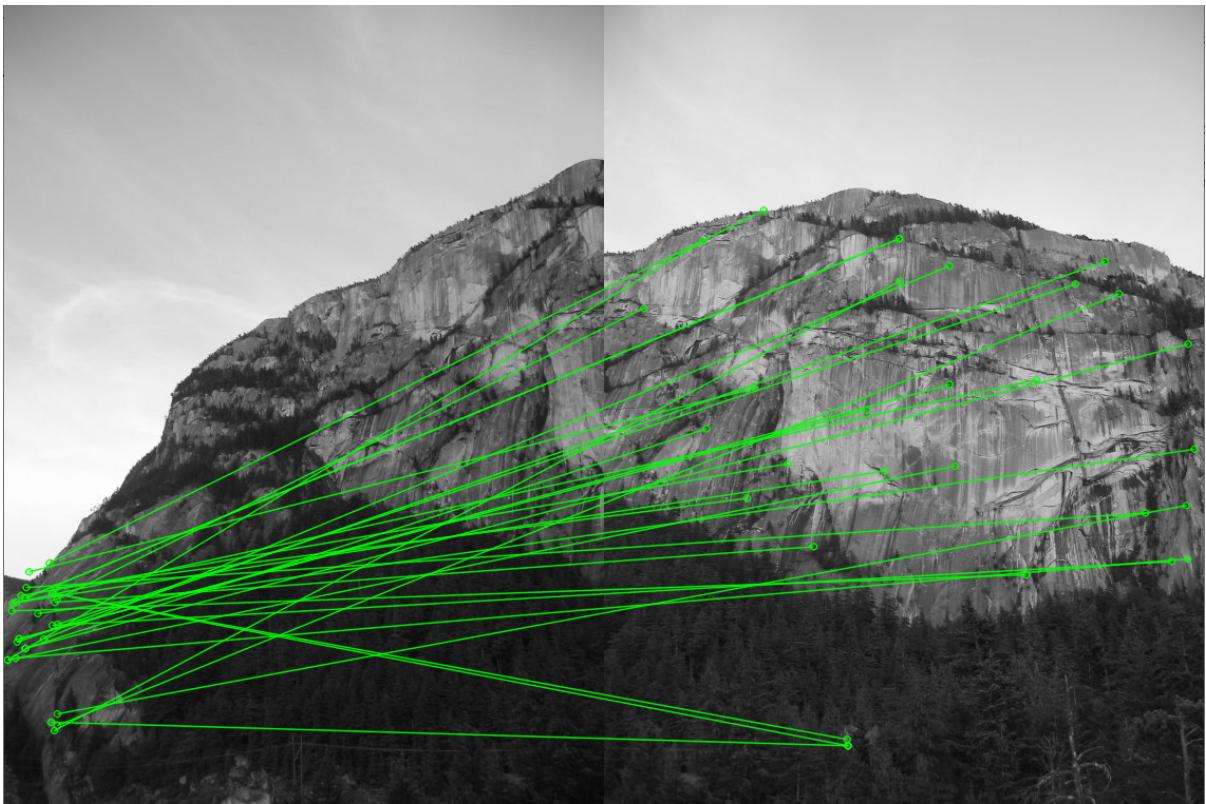
Βήμα 1

Στη συνέχεια, στόχος μας είναι να αντιστοιχίσουμε αυτά τα χαρακτηριστικά μεταξύ των εικόνων με βάση το κριτήριο εγγύτητας το οποίο επιστρέφει την απόσταση μεταξύ των αντίστοιχων descriptors, δηλαδή συγκρίνουμε descriptors από την πρώτη εικόνα με αυτούς της δεύτερης για κάθε ζεύγος εικόνων. Επιλέγουμε τη μέθοδο FLANN για την εύρεση των προσεγγιστικών κοντινότερων γειτόνων, δημιουργώντας το object FlannBasedMatcher μέσω της συνάρτησης cv2.FlannBasedMatcher(index_params, search_params). Ως ορίσματα περνάμε τα dictionaries, που ορίζουν τον αλγόριθμο τον οποίο θα χρησιμοποιήσουμε, και τον αριθμό που πρέπει να διατρέξουμε αναδρομικά τα δέντρα. Όσο μεγαλύτερος ο αριθμός αυτός, τόσο καλύτερη ακρίβεια, για αυτό κάνουμε 50 checks. Έπειτα, καλούμε τη συνάρτηση knnMatch(descriptors1, descriptors2, k=2), η οποία επιστρέφει για κάθε περιγραφητή τις k=2 καλύτερες αντιστοιχίσεις. Έτσι, παίρνουμε πληροφορίες για το συγκεκριμένο χαρακτηριστικό ενδιαφέροντος, όπως τις συντεταγμένες, το size και τους αντίστοιχους descriptors. Παρακάτω παραθέτουμε ενδεικτικά ένα παραδείγμα για τις εικόνες 1 και 2:

```
Image 1
(x,y)= (531.0, 308.0)
Keypoint size is 31.0
Descriptor of this keypoint is: [ 84  89 185 224   8  68  91 128 170 184 130   3 149 151  65  90 224 213
125 208  74   1 219  80 217 164 175  61 144  50  37  58]
Image 2
(x,y)= (295.0, 328.0)
Keypoint size is 31.0
Descriptor of this keypoint is: [ 11 177 229 179  62 243  24 137  31 241   9 163 253 124 216 226 151  23
189 149 151  72 127 213 143  95 223 230 180 134  21 127]
```

Βήμα 2

Στο τρίτο βήμα, καλούμε τη συνάρτηση cv2.drawMatches, για να σχεδιάσουμε τα matching points στις εκάστοτε φωτογραφίες. Θέτουμε argument all_matches[:30], ώστε να αναπαραστήσουμε 30 αντιστοιχίσεις. Προκειμένου να κρατήσουμε μόνο τα αξιόλογα matches, ελέγχουμε αν ο λόγος της απόστασης μεταξύ των δύο ζευγών περιγραφητών είναι μεγαλύτερος από το κατώφλι 0.8.



Bήμα 3: Drawing matches between two pictures

Για να κάνουμε stitching δύο εικόνες μεταξύ τους πρέπει να εφαρμόσουμε αρχικά τον αλγόριθμο RANSAC για την αξιολόγηση του πίνακα της ομογραφίας. Ο πίνακας αυτός χρησιμοποιείται για να μετασχηματίσουμε τη δεύτερη εικόνα, ώστε να έχει ίδια προοπτική όπως η πρώτη εικόνα, η οποία αποτελεί πλαίσιο αναφοράς. Έτσι, θα λαβουμε πληροφορίες για το μετασχηματισμό της δεύτερης εικόνας, τις οποίες θα χρησιμοποιήσουμε για να στοιχίσουμε τη δεύτερη εικόνα με την πρώτη. Για να βρούμε αυτόν τον πίνακα μετασχηματισμού, πρέπει να εξάγουμε τις συντεταγμένες τουλάχιστον 4 σημείων στην πρώτη εικόνα και των αντίστοιχων 4 σημείων στη δεύτερη εικόνα. Στη συνέχεια, σχηματίζουμε δύο λίστες με αυτά τα σημεία. Η πρώτη λίστα που ονομάζεται `list_of_points_1` αφορά τις συντεταγμένες της εικόνας αναφοράς και η δεύτερη λίστα που ονομάζεται `temp_points` αφορά τις συντεταγμένες της δεύτερης εικόνας που θέλουμε να μετασχηματίσουμε. Στη συνέχεια, θα εφαρμόσουμε τη συνάρτηση `cv2.perspectiveTransform(temp_points, H)`, την οποία χρησιμοποιούμε για τον υπολογισμό του πίνακα μετασχηματισμού (ομογραφία). Η συνάρτηση αυτή απαιτεί δύο ορίσματα, μια λίστα σημείων της δεύτερης εικόνας και τον πίνακα `H`. Τέλος, μπορούμε να αλλάξουμε την προοπτική της δεύτερης εικόνας χρησιμοποιώντας τη συνάρτηση `cv2.warpPerspective()`.

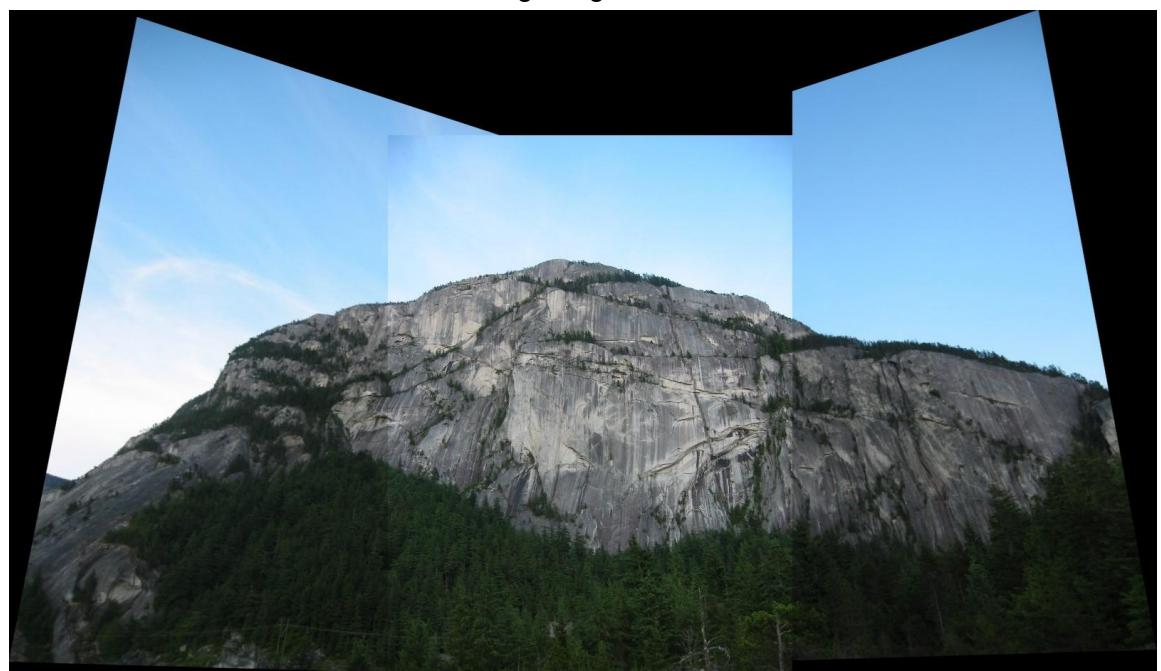
Το δεύτερο και τελικό βήμα για το επιτυχές stitching είναι να χρησιμοποιήσουμε τον πίνακα που βρήκαμε για να υπολογίσουμε το warping transformation με βάση τα αντιστοιχισμένα χαρακτηριστικά.

Θέτουμε ελάχιστο αριθμό αντιστοιχίσεων 10 για να βρούμε το object. Εάν βρεθούν αρκετές αντιστοιχίες, εξάγουμε τις θέσεις των αντιστοιχισμένων χαρακτηριστικών ενδιαφέροντος και στις δύο εικόνες. Στη συνέχεια, μετατρέπουμε τα χαρακτηριστικά ενδιαφέροντος σε τύπο np.float32 επειδή θα τα χρησιμοποιήσουμε ως παράμετρο της συνάρτησης `cv2.findHomography()`. Αφού έχουμε υπολογίσει τον πίνακα ομογραφίας, μπορούμε τελικά να συνενώσουμε τις εικόνες μεταξύ τους χρησιμοποιώντας τη συνάρτηση `cv2.warpImages()`. Είναι καλό να θυμόμαστε ότι η

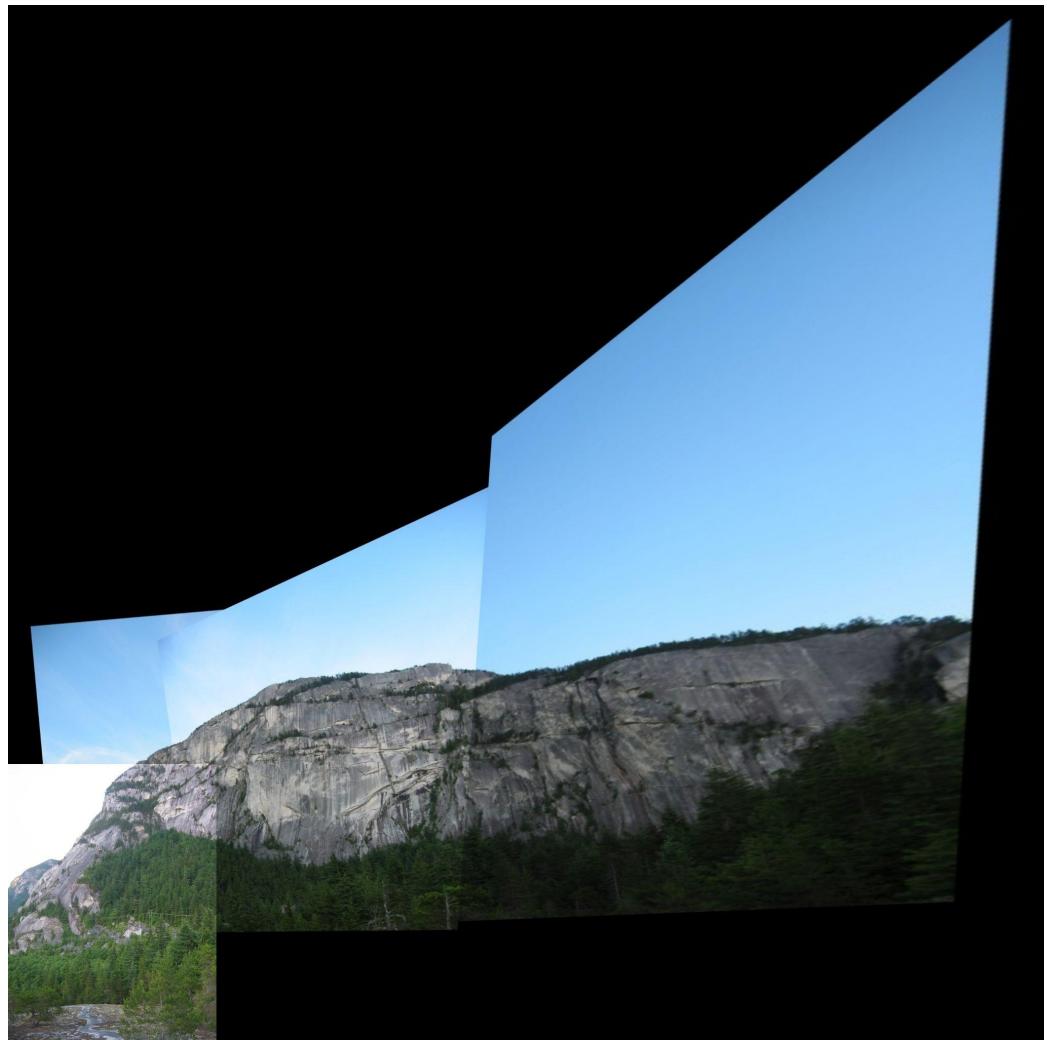
αντιστοίχιση χαρακτηριστικών δεν παράγει πάντα 100% ακριβείς αντιστοιχίες. Η αντιστοίχιση ενδέχεται να έχει λάθη σε πολλές περιπτώσεις και γι' αυτό η συνάρτηση cv2.findHomography() χρησιμοποιεί τη διαδικασία RANSAC, η οποία καθιστά τη συνάρτηση “ανθεκτική” στις ακραίες τιμές. Χρησιμοποιώντας αυτή τη μέθοδο μπορούμε να λάβουμε ακριβή αποτελέσματα ακόμη και να γνωρίζουμε αν έχουμε υψηλό ποσοστό λανθασμένων αντιστοιχίσεων.



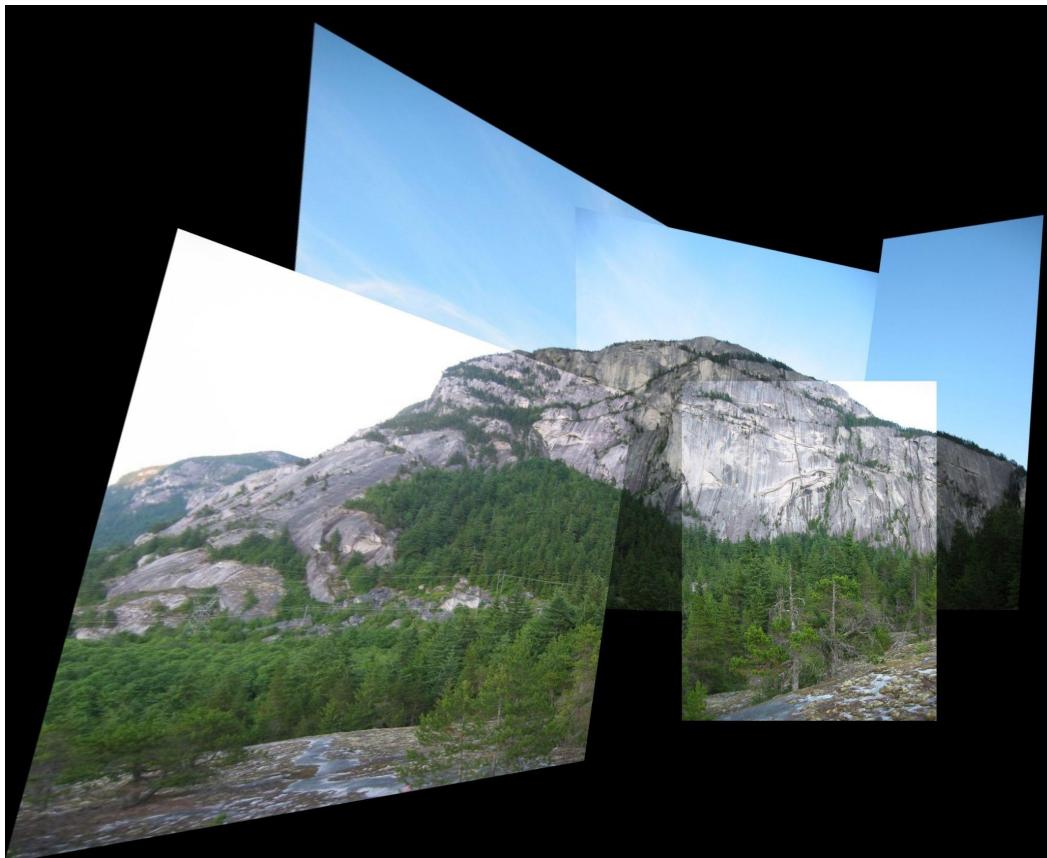
Stitching images 1 and 2



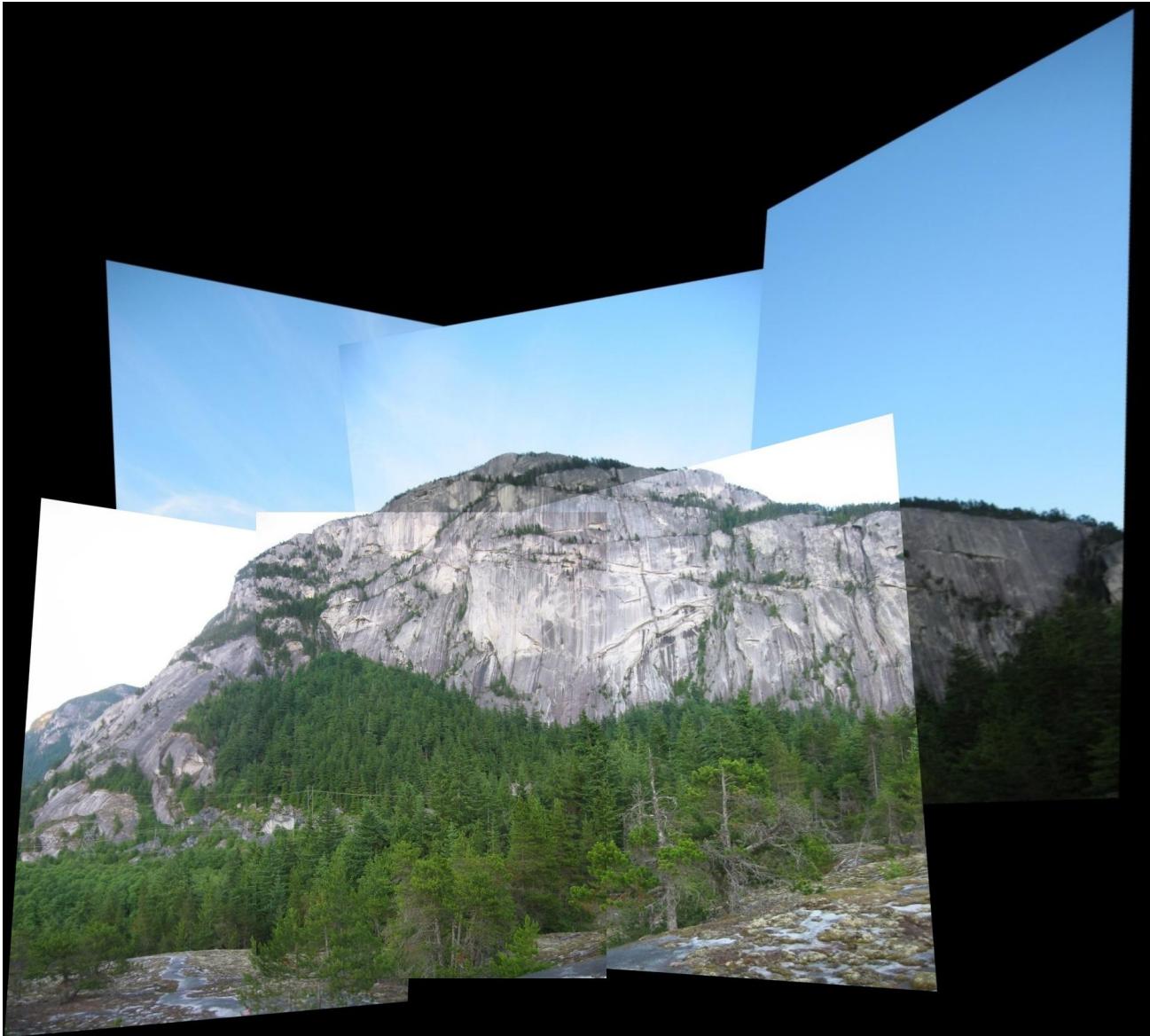
Stitching images 1,2 and 3



Stitching images 1,2, 3 and 4



Stitching images 1,2, 3, 4 and 6



Stitching images 1,2, 3, 4, 6 and 5