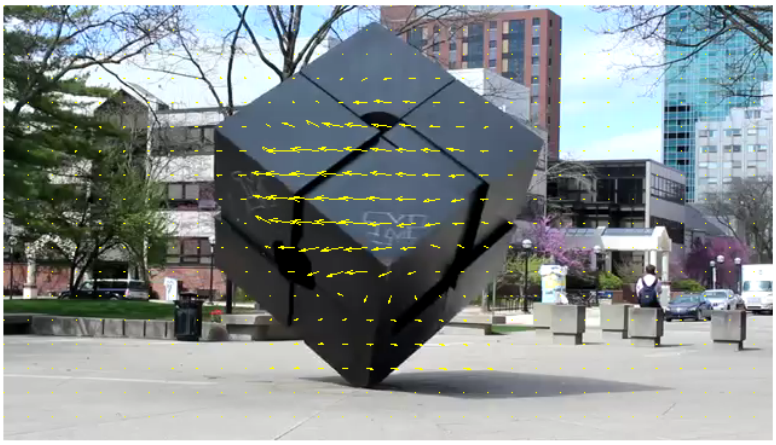
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

2η Εργαστηριακή Άσκηση

Παρακολούθηση Χεριού με Χρήση του Πεδίου Οπτικής Ροής(Optical Flow) με τη Μέθοδο των Lucas-Kanade

Μάθημα : Όραση Υπολογιστών



Ροή Σ

Συνεργάτες :

- Βαβουλιώτης Γεώργιος (**Α.Μ. : 03112083**)
- Σταυρακάκης Δημήτριος (**Α.Μ. : 03112017**)

Εισαγωγή

Σκοπός της δεύτερης εργαστηριακής άσκησης είναι η υλοποίηση ενος συστήματος Παρακολούθησης Χεριού (*Hand Tracking*) σε μια ακολουθία βίντεο με gestures. Το σύστημα το οποιό καλούμαστε να υλοποιήσουμε αρχικά θα πρέπει να ανιχνεύει για το πρώτο frame μόνο την περιοχή που βρίσκετε ο χρήστης, χρησιμοποιώντας την μάσκα χρήστη η οποία προκύπτει απο την επεξεργασία της πληροφορίας βάθους που προσφέρει ο αισθητήρας kinect. Θα πρέπει να τονίσουμε οτι δεν υλοποιήσαμε δική μας μάσκα αλλά χρησιμοποιήσαμε τα δεδομένα βάθους του kinect που μας δίνονται στο αρχείο **Chalearn-Depth.zip**. Αυτό που πρέπει να γίνεται στη συνέχεια είναι να ανιχνεύεται το δεξί χέρι του νοηματιστή με χρήση ενός πιθανοτικού ανιχνευτή ανθρωπίνου δέρματος(Μέρος 1). Τέλος η παρακολούθηση της κίνησης θα γίνει με χρήση του διανυσματικού πεδίου οπτικής ροής, υπολογισμένο με την μέθοδο ***Lucas-Kanade***(Μέρος 2).

Μέρος 1 : Ανίχνευση Δέρματος Χρήστη

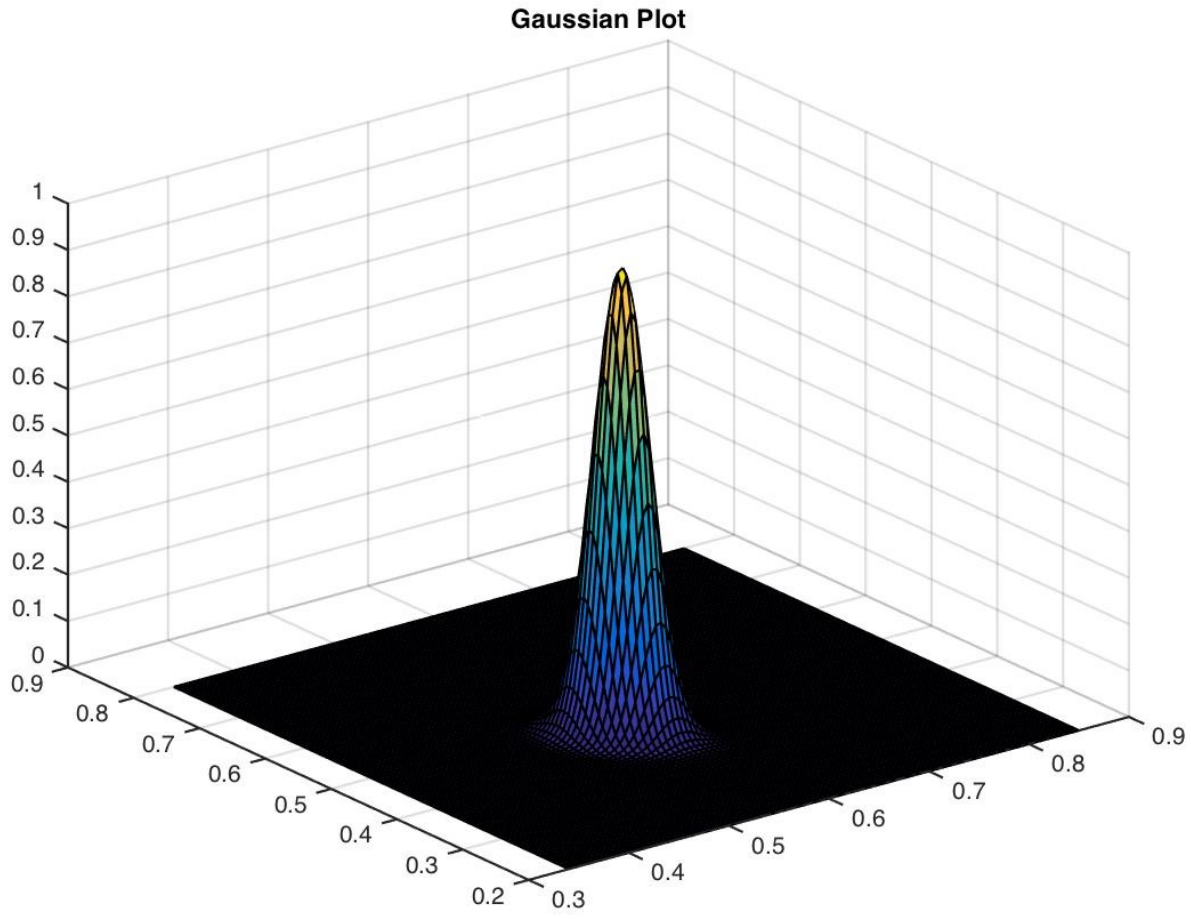
Όπως ανέφερα και στην εισαγωγή της άσκησης, αυτό που θα κάνουμε είναι να χρησιμοποιήσουμε μια πιθανοτική διαδικασία για να ανιχνεύσουμε το δεξί-πρωτεύον χέρι του νοηματιστή. Αρχικά για να ανιχνεύσουμε τα σημεία του δέρματος θα χρησιμοποιήσουμε το χρωματικό χώρο **YCbCr** αλλά θα κρατήσουμε μόνο τα κανάλια **Cb** και **Cr** διότι μόνο αυτά έχουν την πληροφορία του χρώματος της εικόνας. Θα πρέπει να αναφέρω εκ νέου οτι αυτό το κάνουμε για το πρώτο frame αφού μόνο σ'αυτό μας ενδιαφέρει να ανιχνεύσουμε το δεξί χέρι του νοηματιστή όπως προείπα. Απο την εκφώνηση της άσκησης μας δίνεται οτι το χρώμα του δέρματος μοντελοποιείται με την παρακάτω Γκαουσιανή Κατανομή:

$$P(\mathbf{c} = \text{skin}) = \frac{1}{\sqrt{|\Sigma|\,(2\pi)^2}} e^{-\frac{1}{2}(\mathbf{c}-\boldsymbol{\mu})\Sigma^{-1}(\mathbf{c}-\boldsymbol{\mu})'}$$

όπου c είναι το διάνυσμα τιμών Cb και Cr για κάθε pixel της εικόνας μας.

Το διάνυσμα μέσης τιμής μ και ο πίνακας συνδιακύμανσης Σ που χρησιμοποιούνται στην παραπάνω κατανομή προκύπτουν από τα δοσμένα **δείγματα δέρματος**, τα οποία παραθέτω παρακάτω :

Με την βοήθεια της συνάρτησης *surf()* του Matlab μπορώ να σας παρουσιάσω το γράφημα της παραπάνω Γκαουσιανής κατανομής, το οποίο φαίνεται παρακάτω :

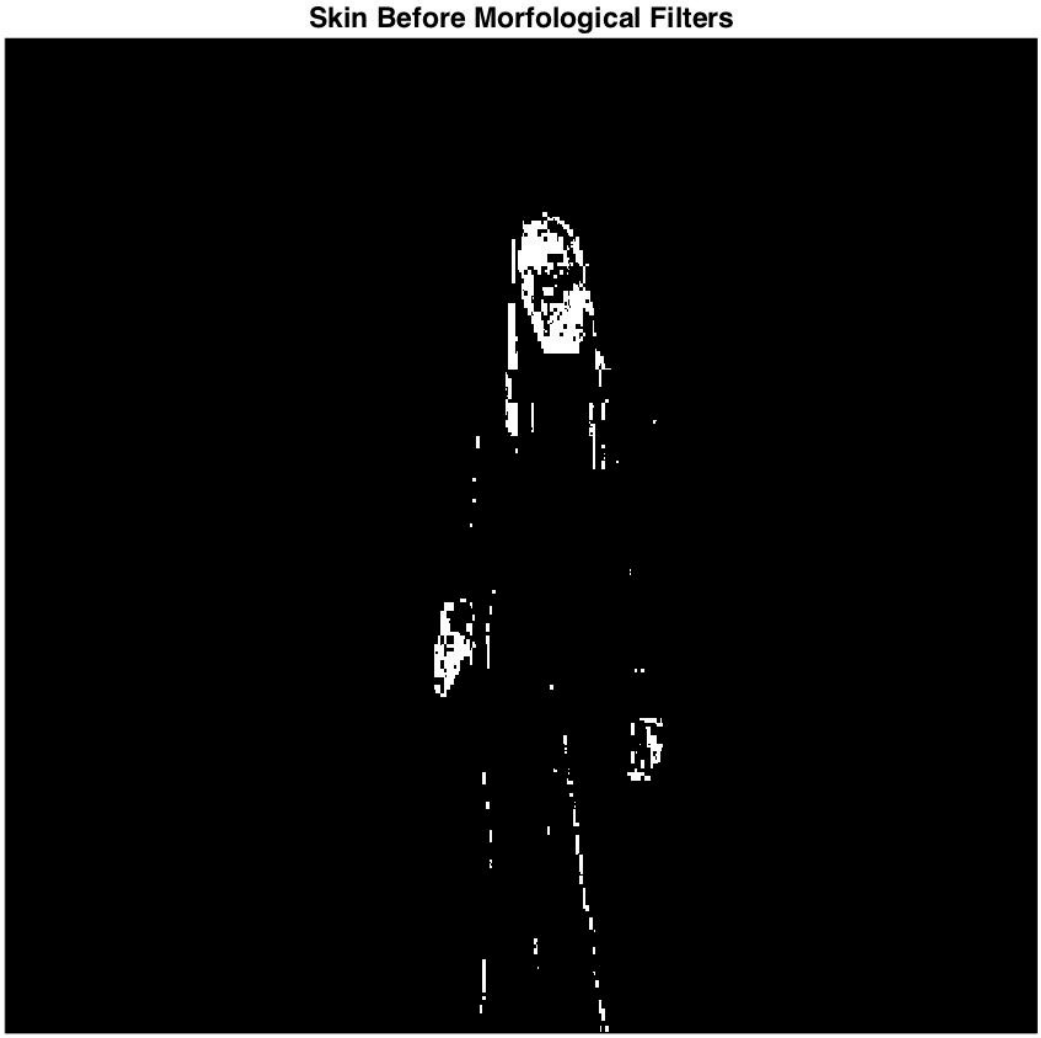


Απο την γραφική παράσταση της Gaussian που φαίνεται παραπάνω είναι προφανές οτι έχει την μορφή που αναμέναμε να έχει, γεγονός το οποίο είναι λογικό αφού ακολουθήσαμε ακριβώς την εξίσωση που μας δόθηκε στην εκφώνηση της άσκησης και την υλοποίησαμε στο Matlab. Κάτι άλλο το οποίο θα πρέπει να παρατηρήσουμε είναι οτι η παραπάνω Gaussian είναι κανονικοποιημένη στο διάστημα [0,1] το οποίο είναι απαραίτητο για να έχω ορθά αποτελέσματα όπως θα φανεί παρακάτω.

Στη συνέχεια αυτό που κάνουμε είναι να εφαρμόσουμε μια κατωφλιοποίηση ώστε να μην πάρουμε λάθος σημεία. Διαλέξαμε απο τις ενδεικτικές τιμές κατωφλίου την τιμή 0.3 για threshold δηλαδή επιλέψαμε να κρατήσουμε τα σημεία εκείνα με πιθανότητα μεγαλύτερη ή ίση του 0.3. Παρακάτω σας παρουσιάζω την ανίχνευση δέρματος όπως προκύπτει πριν την εφαρμογή της μάσκας kinect :

Απο την παραπάνω εικόνα είναι προφανές οτι η ανίχνευση δέρματος δεν μας έχει δώσει και πολύ αξιόπιστα αποτελέσματα, αφού βλέπουμε οτι έχουν ανιχνευτεί σαν περιοχές δέρματος και περιοχές η οποίες στην πραγματικότητα δεν είναι περιοχές που αντιστοιχούν σε δέρμα.
Στο σημείο αυτό για να μην πάρω λάθος σημεία σαν δέρμα και η τελική μου ανίχνευση είναι λανθασμένη θα πολλαπλάσιάσω τον πίνακα του δέρματος με την μάσκα βάθους που μας υπάρχει στο **ChalearnDepth.zip** για το πρώτο frame. Πλέον έχει μείνει μόνο να κάνω μια μορφολογική επεξεργασία της δυαδικής εικόνας ώστε να καταφέρουν να αποκτήσουν συνοχή οι διάφορες περιοχές δέρματος μεταξύ τους. Αρχικά εφαρμόζω ενα opening με μικρό δομικό στοιχείο για να καλύψω τις τρύπες και στη συνέχεια ενα closing με μεγάλο δομικό στοιχείο για να εξαλείψω τις μικρές περιοχές και τελικά η δυαδική εικόνα να αποκτήσει την ζητούμενη συνοχή όσο αφορά τις περιοχές δέρματος.

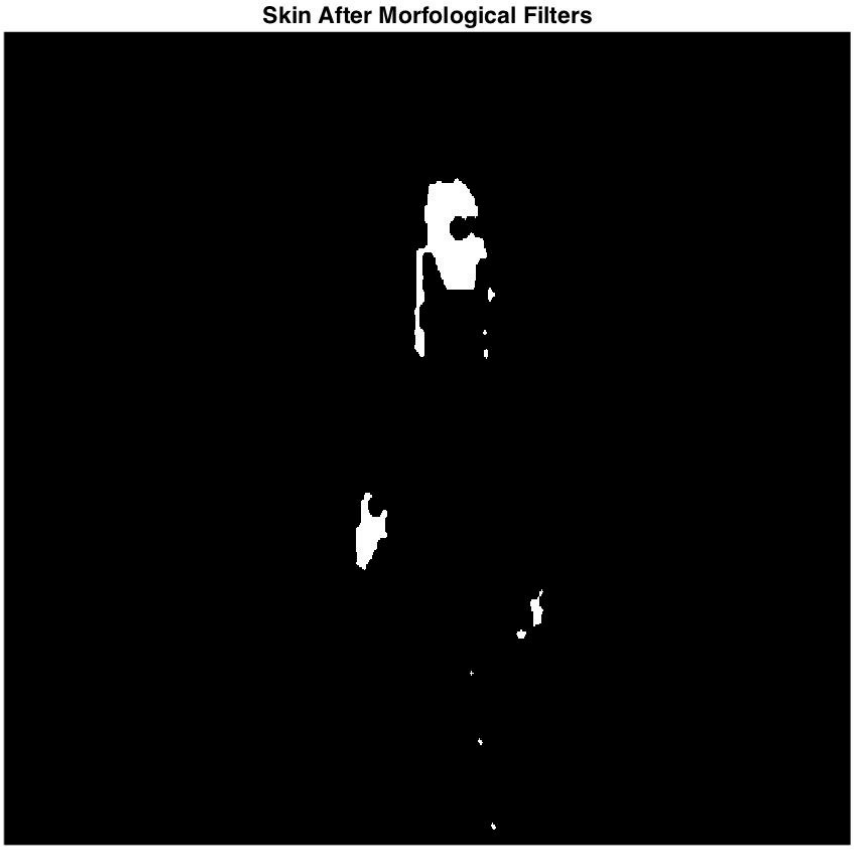
Στη συνέχεια σας παραθέτω την δυαδική εικόνα δέρματος **πρίν την μορφολογική επεξεργασία**(η μάσκα kinect έχει εφαρμοστεί):



Η δυαδική εικόνα δέρματος **μετά την μορφολογική επεξεργασία(open,close)** φαίνεται παρακάτω :

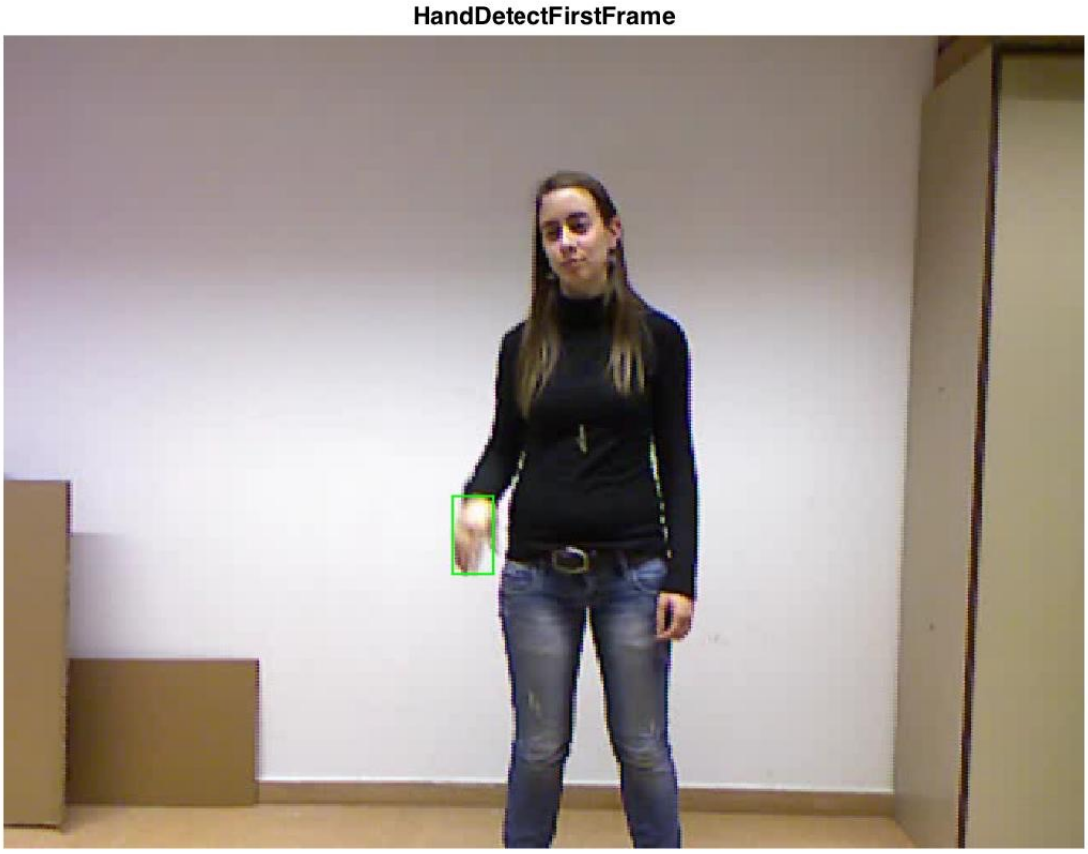
Αρχικά παρουσιάζω το αποτέλεσμα που επιφέρει το opening με μικρό πυρήνα :

Μετά την εφαρμογή και του closing παίρνω την τελική ανίχνευση δέρματος, η οποία φαίνεται παρακάτω:



Σχολιασμός : Παρατηρώ οτι όλα όσα ισχυρίστηκα οτι θα κάνουν τα μορφολογικά φίλτρα το έκαναν(κάλυψη τρυπών, εξάλειψη μικρών περιοχών), ωστόσο παρατηρούνται ακόμα κάποιες μικρές περιοχές που δεν έχουν εξαλειφθεί ακόμα. Αυτό μπορεί να επιτευχθεί αν γίνει μια ακόμα μορφολογική επεξεργασία αλλά δεν το έκανα αφού το αποτέλεσμα που θα έδινε είναι προφανές και δεν επηρεάζει την τελική ανίχνευση του πρωτεύον χεριού του νοηματιστή.

Αφού πλέον έχω κάνει όλα τα παραπάνω και έχω ανιχνεύσει σωστά τις διάφορες περιοχές δέρματος πλέον θα πρέπει να επιλέξω εκείνη την περιοχή στην οποία αντιστοιχεί το δεξί χέρι του νοηματιστή. Η επιλογή αυτή μπορεί να γίνει με διάφορους τρόπους και εμείς την υλοποιήσαμε με δυο διαφορετικούς οι οποίοι όμως μας δίνουν ακριβώς το ίδιο αποτέλεσμα(και οι δυο είναι στον κώδικα **boundingBox.m** αλλά ο δεύτερος τρόπος είναι σε σχόλια). Ο πρώτος τρόπος χρησιμοποιεί την εντολή state = regionprops(label, 'BoundingBox'). Η παράμετρος 'BoundingBox' καθορίζει τη μορφή του αποτελέσματος που θα γυρίσει η regionprops. Αυτή επιστρέφει ένα struct που περιέχει τις συντεταγμένες της πάνω αριστερά γωνίας των Bounding Boxes (x,y) για όλες τις περιοχές δέρματος που βρίσκει καθώς και το μέγεθός αυτών (width,height). Εμείς κάνουμε το struct αυτό αρχικά cell array και μετά απλό array και επιλέγουμε τη γραμμή αυτή που έχει τη μικρότερη τιμή στο x, και επομένως αντιστοιχεί στην αριστερότερη περιοχή, δηλαδή το χέρι που επιζητούμε. Ο δεύτερος τρόπος, ο οποίος είναι σε σχόλια ουστικά δίνει labels-ονόματα στις διάφορες περιοχές δέρματος και τελικά επιλέγει αυτή που αντιστοιχεί στο δεξί χέρι του νοηματιστή.Το αποτέλεσμα της μεθόδου αυτής, δηλαδή η τελική ανίχνευση του χεριού φαίνεται παρακάτω όπως προέκυψε ακολουθώντας τα παραπάνω βήματα:



Σχολιασμός : Το παραπάνω αποτέσμα μπορεί πολύ εύκολα να επηρεαστεί απο την επιλογή της παραμέτρου κατωφλίου αφού αν η παράμετρος αυτή επιλεγεί αρκετά μικρή θα επιλεγούν επιπλέον περιοχές οι οποίες τελικά μπορεί να επηρεάσουν την τελική ανίχνευση αφού μπορεί να επιλεγεί κάποιο πλαίσιο το οποίο εκτός απο το δεξί χέρι του νοηματιστή να περιέχει και κάτι αλλό το οποίο δεν το επιθυμούμε. Επίσης αν αυξήσουμε αρκετά το κατώφλι υπάρχει ο κίνδυνος να μην επιλεγεί η περιοχή του δεξιού χεριού και τελικά να έχω λάθος ανίχνευση. Συμπερασματικά θα πρέπει να προσεχθεί η τιμή που θα πάρουμε σαν κατώφλι.

Στα αρχεία **main_script.m** και **boundingBox.m** που σας επισυνάπτω στο zip αρχείο υπάρχουν όλα όσα σας περιέγραψα παραπάνω σε μορφή κώδικα, ο οποίος μας έδωσε τα αποτελέσματα που σας παράθεσα παραπάνω.

Μέρος 2 : Παρακολούθηση Χεριού

2.1 Υλοποίηση του Αλγορίθμου Lucas-Kanade

Στο μέρος αυτό υλοποιούμε τον αλγόριθμο Lucas-Kanade, ο οποίος αυτό που κάνει είναι να υπολογίζει την οπτική ροή σε κάθε σημείο της εικόνας με την βοήθεια της μεθόδου των ελαχίστων τετραγώνων, θεωρώντας οτι το πεδίο οπτικής ροής d είναι σταθερό σε ενα μικρό παράθυρο γύρω απο το κάθε σημείο, ελαχιστοποιώντας το τετραγωνικό σφάλμα το οποίο δίνεται απο τον παρακάτω τύπο :

$$J_{\mathbf{x}}(\mathbf{d}) = \int_{\mathbf{x}' \in \mathbb{R}^2} G_\rho(\mathbf{x} - \mathbf{x}') \left[I_n(\mathbf{x}') - I_{n-1}(\mathbf{x}' + \mathbf{d}) \right]^2 \mathrm{d}\mathbf{x}'$$

όπου :

- Gp είναι μια συνάρτηση παραθύρωσης(πχ Gaussian) .
- d είναι το πεδίο οπτικής ροής.
- το I(x) αποτελέι μια εικόνα.
- x είναι ενα τυχαίο σημείο της εικόνας.

Θα πρέπει να τονίσουμε επίσης οτι δυο εικόνες έρχονται σε αντιστοιχία με την βοήθεια του πεδίου οπτικής ροής d με την παρακάτω μαθηματική σχέση :

$$I_n(\mathbf{x}) \approx I_{n-1}(\mathbf{x}+\mathbf{d})$$

Για να μπορέσουμε να υλοποιήσουμε τον αλγόριθμο θεωρούμε οτι έχουμε μια εκτίμηση di για το d και προσπαθούμε να τη βελτιώσουμε κάθε φορά κατά ενα παράγοντα u, δηλαδή ισχύει ότι $d_{i+1}=d_i+u$.
Αν τώρα αναπτύξουμε σε σειρά Taylor την σχέση $I_{n-1}(\mathbf{x}+\mathbf{d}) = I_{n-1}(\mathbf{x}+\mathbf{d}_i+\mathbf{u})$

προκύπτει οτι :

$$I_{n-1}(\mathbf{x}+\mathbf{d}) \approx I_{n-1}(\mathbf{x}+\mathbf{d}_i) + \nabla I_{n-1}(\mathbf{x}+\mathbf{d}_i)^T\mathbf{u}$$

Η παραπάνω έκφραση που προέκυψε μετά το ανάπτυγμα Taylor μπορεί να αντικατασταθεί στην πρώτη σχέση και να προκύψει τελικά η λύση ελαχίστων τετραγώνων που βελτιώνει το d σε κάθε σημείο δίνεται απο την παρακάτω σχέση :

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} (G_\rho * A_1^2)(\mathbf{x}) + \epsilon & (G_\rho * (A_1A_2))(\mathbf{x}) \\ (G_\rho * (A_1A_2))(\mathbf{x}) & (G_\rho * A_2^2)(\mathbf{x}) + \epsilon \end{bmatrix}^{-1} \cdot \begin{bmatrix} (G_\rho * (A_1E))(\mathbf{x}) \\ (G_\rho * (A_2E))(\mathbf{x}) \end{bmatrix}$$

όπου

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

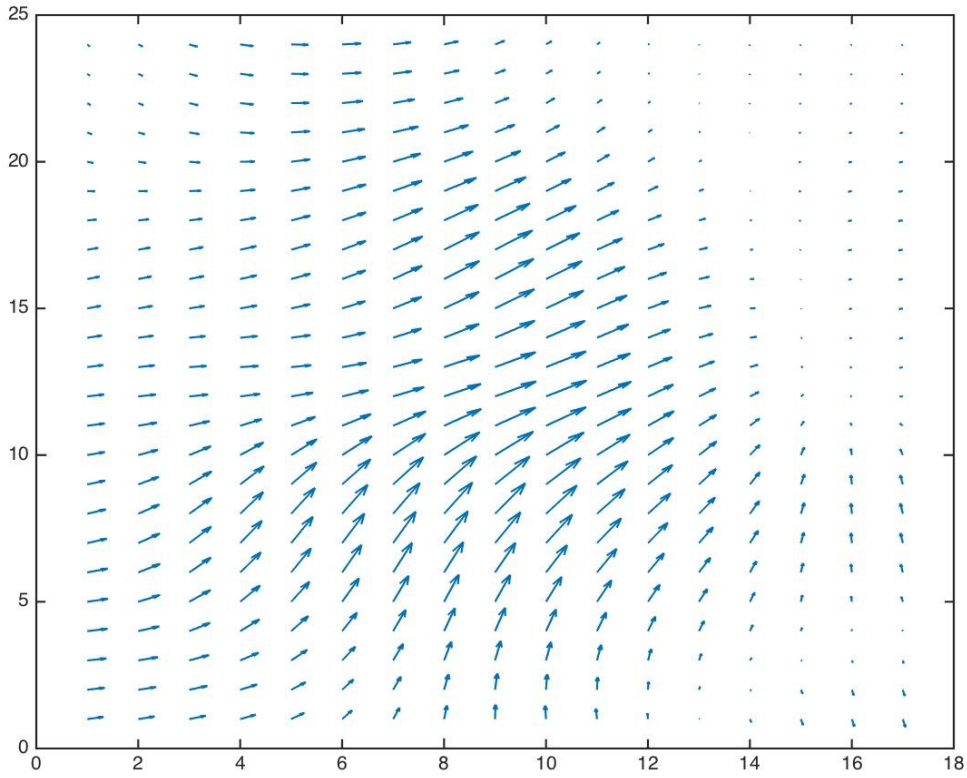
$$E(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x}+\mathbf{d}_i)$$

Ο αλγόριθμος Lucas-Kanade που υλοποήσαμε στο Matlab και είναι το αρχείο με όνομα **lk.m** που υπάρχει στο zip αρχείο δέχεται σαν είσοδο δυο παραθυροποιημένα διαδοχικά καρέ βίντεο με βάση το bounding box, με εύρος ρ του Gaussian παραθύρου, την θετική σταθερά ε και την αρχική εκτίμηση για το πεδίο οπτικής ροής(και στις δυο διαστάσεις) και τελικά επιστρέφει το d. Η παράμετρος ε είναι μια σταθερά την οποία χρησιμοποιούμε για να ενισχύσουμε τυχόν περιοχές που έχουν μειωμένη υφή ώστε να μπορέσουμε να εντοπίσουμε στη συνέχεια την μετατόπισή τους ανάμεσα στα δυο καρέ.
Αφού τρέξαμε τον κώδικα για πολλά ζευγάρια παραμέτρων **ρ,ε** καταλήξαμε οτι το καλύτερο αποτέλεσμα το πετυχαίνουμε για **ρ = 8.9** και **ε = 0.008** όπως φαίνεται και στο κώδικα του πηγαίου αρχείου που σας παραδίδω. Θα πρέπει να επισημάνω επίσης ότι για να επιτευχθεί η απαιτούμενη σύγκλιση έγιναν 80 επαναλήψεις, για να είμαστε σίγουροι οτι θα πάρουμε σωστό αποτέλεσμα τελικά, γεγονός το οποίο όμως δεν μας στοίχισε σε χρόνο πολύ αφού ο

κώδικας τρέχει σε λίγο χρόνο(20 επαναλήψεις αρκούσαν για να επιτευχθεί σύγκλιση ωστόσο κάναμε παραπάνω για σιγουριά).

Για τις παραπάνω τιμές για τις παραμέτρους ρ και ε σας παρουσιάζω ενδεικτικά τα αποτελέσματα τις οπτικής ροής απο το καρέ 1 στο καρέ 2.

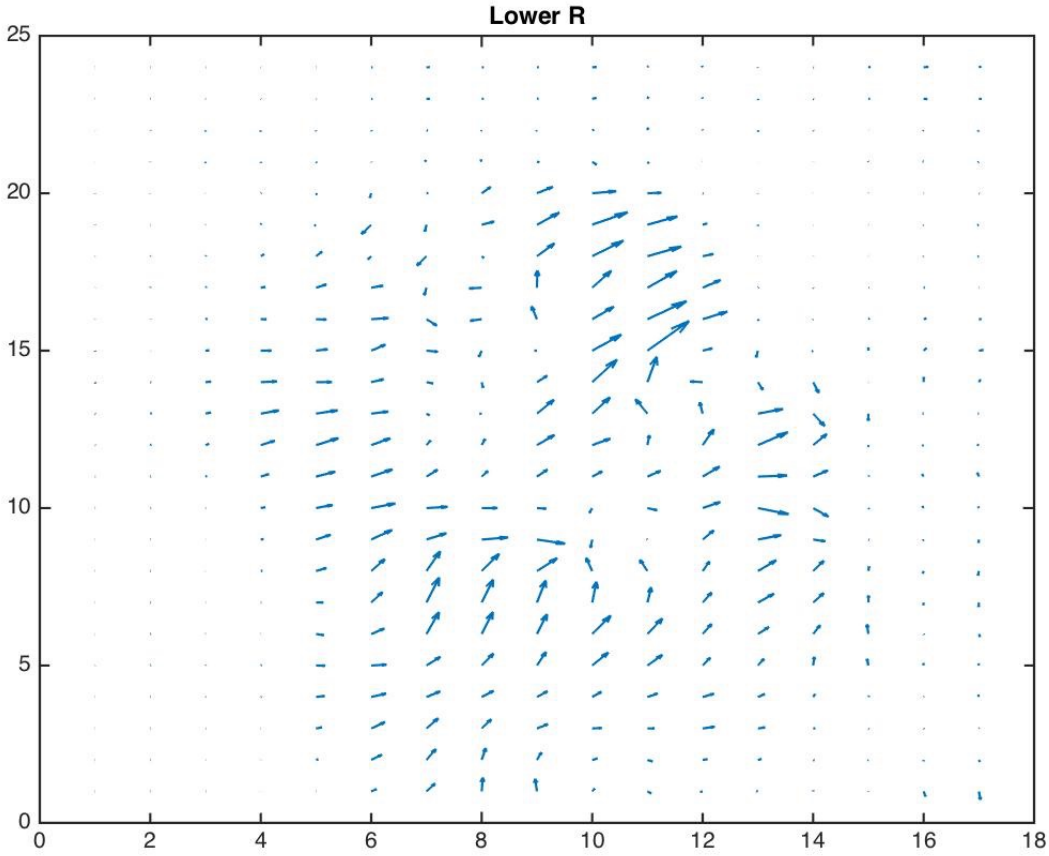
Οπτική Ροή (καρε 1 σε καρε 2) :



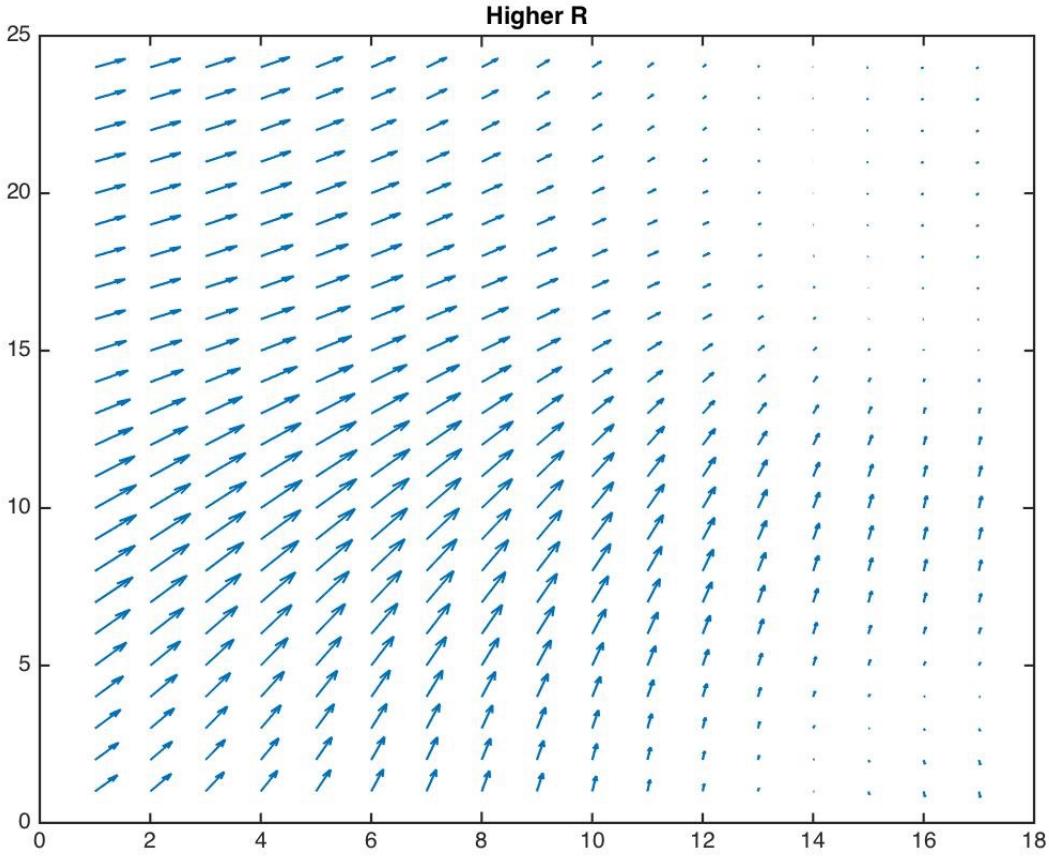
Σχολιασμός : Απο τα παραπάνω σχήματα οπτικής ροής μπορούμε να παρατηρήσουμε οτι υπάρχει μια ομαλότητα στο πεδίο της οπτικής ροής, γεγονός το οποίο μας επιβεβαιώνει την ορθότητα των αποτελεσμάτων μας, αφού η ομαλότητα αυτή θα έπρεπε να υπάρχει διότι αναφερόμαστε σε διαδοχικά καρέ ακολουθίας βίντεο και δεν θα μπορούσε να υπάρχει κάποια μεγάλη μεταβολή στα σημεία της εικόνας. Τα γραφήματα για την οπτική ροή των υπολοίπων καρέ είναι στο φάκελο με όνομα **results-opticalflow** και μπορείτε να τα δείτε εκεί όλα συγκεντρωμένα.

Στη συνέχεια θα εξετάσω την επίδραση που είχε στο αποτέλεσμα μια μεταβολή σε κάποια απο τις παραμέτρους ρ και ε για το **καρέ 1 στο καρέ 2** (οπτική ροή απο το καρέ1 στο καρέ2). Για τον σκοπό αυτό εκτελώ το αρχείο με όνομα **ParametersEffect.m** που σας επισυνάπτω στο αρχείο zip. Τα αποτελέσματα που παίρνω φαίνονται παρακάτω :

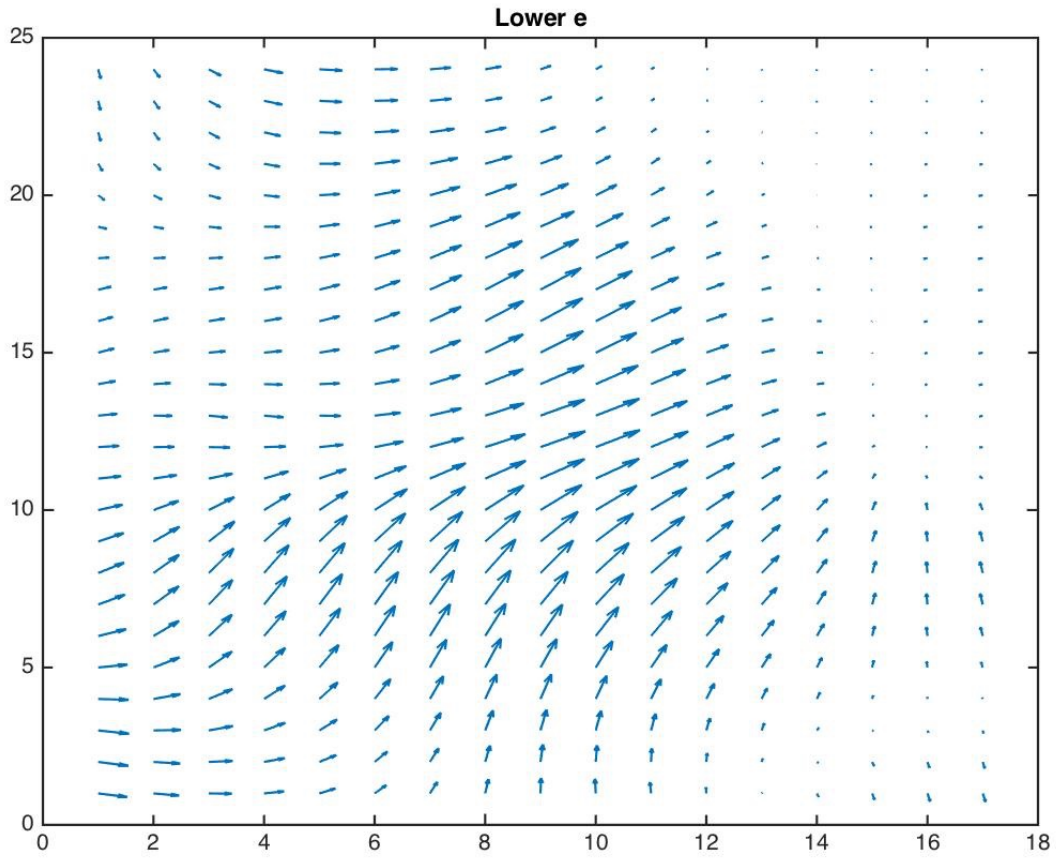
Χαμηλότερο(Lower) ρ :



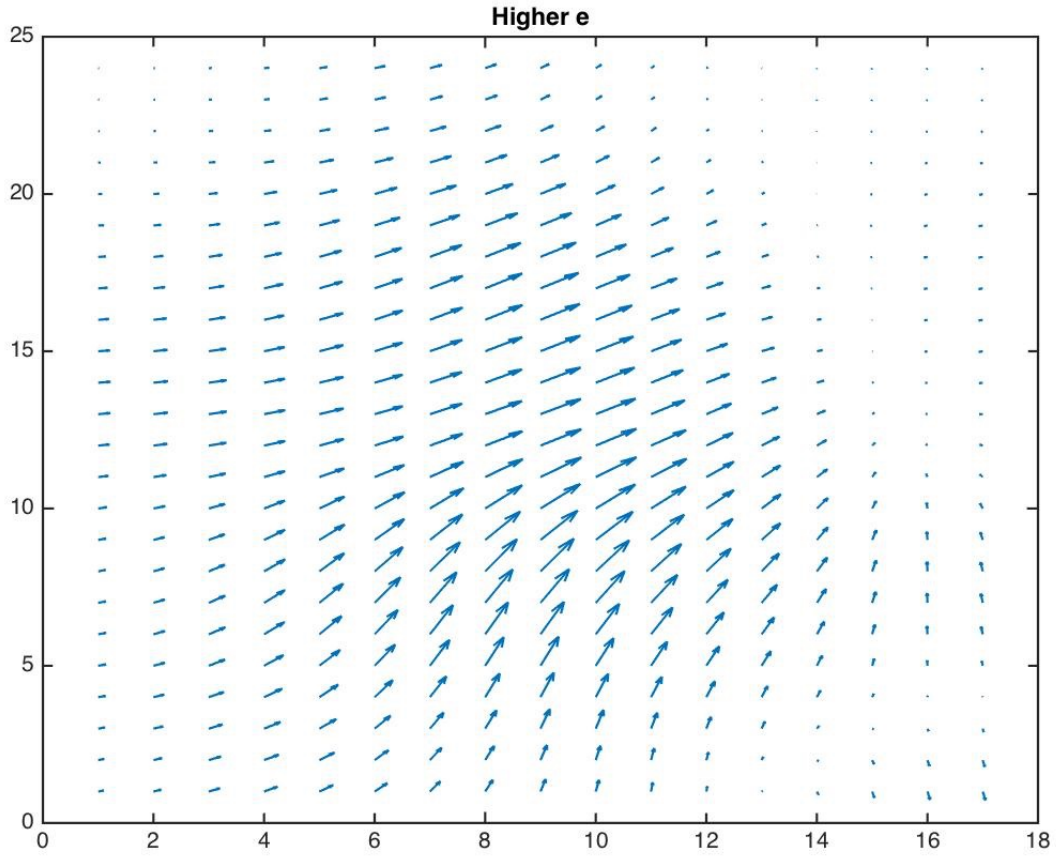
Υψηλότερο(Higher) ρ :



Χαμηλότερο(Lower) ε :



Υψηλότερο(Higher) ε:



Σχολιασμός :

- Απο το πρώτο γράφημα παρατηρώ οτι αν **μειώσω αισθητά το ρ** τότε θα προκύψουν προφανώς πολύ απότομες μεταβολές στην διεύθυνση των διανυσμάτων μετατόπισης που ανήκουν σε διαδοχικά pixels. Αύτο το φαινόμενο δεν είναι ορθό αφού περιμένουμε διαδοχικά pixel να μετατοπίζονται με παρόμοιο τρόπο. Ενα ακόμα φαινόμενο που παρατηρείται είναι η μείωση του μήκους των διανυσμάτων μετατόπισης αν τα συγκρίνουμε με αυτά που προέκυψαν απο την επιλογή παραμέτρων που εμείς βρήκαμε σαν ‘βέλτιστη’.

- Αν τώρα **αυξήσω το ρ** κατά κάποιο σημαντικό παράγοντα είναι προφανές οτι προκύπτει αύξηση του μήκους των διανυσμάτων μετατόπισης του πεδίου οπτικής ροής, όπως είναι προφανές αν δει κανείς το δέυτερο γράφημα. Η αύξηση του μήκους των διανυσμάτων σημαίνει στην πράξη οτι κάθε σημείο έκανε μεγαλύτερη μετατόπιση απο αυτή που έκανε στην πραγματικότητα.

- Μια **μείωση του ε** παρατηρούμε οτι οδηγεί στον εντοπισμό κίνησης σε περισσότερες περιοχές και τα διανύσματα μετατόπισης όπως είναι λογικό έχουν μικρότερο μέγεθος.

- Μια **αύξηση της παραμέτρου ε** οδηγεί σε εντοπίσμο κίνησης σε λιγότερες περιοχές, διότι πολλά pixel που είχαν κάποιο διάνυσμα μετατοπίσης πλέον δεν έχουν. Τέλος στα σημεία που διατηρείται η κίνηση είναι λίγο πιο απότομη όπως μπορεί να παρατηρήσει κανείς απο το τέταρτο σχήμα.

2.2 Πολυ-Κλιμακωτός Υπολογισμός Οπτικής Ροής

Στο ερώτημα αυτό καλούμαστε να υλοποιήσουμε την πολυκλιμακωτή μέθοδο του Lucas- Kanade. Ουσιαστικά, αυτό που κάνουμε εδώ είναι η ανάλυση της εικόνας σε gaussian πυραμίδες και ο υπολογισμός της οπτικής ροής από την πιο μικρή προς τη πιο μεγάλη κλίμακα διαδοχικά, έχοντας κάθε φορά ως αρχική συνθήκη του αλγορίθμου Lucas-Kanade που εφαρμόζεται για κάθε κλίμακα την οπτική ροή που υπολογίστηκε στην αμέσως προηγούμενη κλίμακα, σαφώς με κλιμάκωση μεγέθους και μέτρου. (Ετσι επιτυγχάνουμε και ταχύτερη σύγκλιση.) Η ανωτέρω διαδικασία υλοποιήθηκε σαν ξεχωριστή συνάρτηση(**lk_ms.m**) και καλεί την απλή μέθοδο Lucas-Kanade(**lk.m**) ως υπορουτίνα.

Λεπτομέρειες υλοποίησης :

i. Αρχικά υπολογίζουμε τις gaussian πυραμίδες για τις εικόνες που έχουμε σαν όρισμα στην κλήση της συνάρτησης, αφού πρώτα τις φιλτράρουμε με ένα βαθυπερατό φίλτρο (imgaussfilt(I,3),Gaussian με τυπική απόκλιση 3). Ο υπολογισμός των πυραμίδων γίνεται με τη συνάρτηση impyramid του matlab. (οι διαστάσεις της εξόδου της συνάρτησης impyramid είναι οι μισές από την αρχική γι'αυτό χρειαζόμαστε και την κλιμάκωση σε επόμενο στάδιο) Επιλέξαμε μετά απο δοκιμές ότι το να χρησιμοποιήσουμε 4 κλίμακες μας δίνει ικανοποιητικά αποτελέσματα. (η συνάρτησή μας δέχεται των αριθμό των κλιμάκων σαν όρισμα και έτσι παίξαμε με τις παραμέτρους και καταλήξαμε σε αυτό το συμπέρασμα) .

ii. Στη συνέχεια, ξεκινώντας ένα Loop από τη μικρότερη κλίμακα προς την μεγαλύτερη (την αρχική δηλαδή) καλούμε σαν υπορουτίνα τη συνάρητηση lk.m που υλοποιήσαμε για το προηγούμενο ερώτημα. Ως αρχική τιμή της οπτικής ροής, την 1η φορά δίνουμε πίνακες μηδενικών, ενώ σε κάθε επόμενη κλήση της, τής δίνουμε σαν αρχική τιμή οπτικής ροής το αποτέλεσμα που μας έδωσε η αμέσως μικρότερη κλίμακα, προφανώς προσαρμόζοντας κατάλληλα το μέτρο και τις διαστάσεις του ώστε να συμβαδίζει με τη μεγαλύτερη κλίμακα. (2*imresize(d)) .

iii. Αφού φτάσουμε στο τέλος, στην κλήση της συνάρτησης lk.m για την αρχική μας εικόνα (αυτή με τη μεγαλύτερη κλίμακα δηλαδή) με τις κατάλληλες αρχικές τιμες για την οπτική ροή επιστρέφουμε σαν αποτέλεσμα του multiscale Lucas-Kanade αυτό που μας επιστρέφει η τελευταία κλήση της lk.m χωρίς resizing αυτή τη φορά.

Αυτο που μπορούμε να παρατηρήσουμε για τη μέθοδο αυτή είναι το γεγονός ότι εντοπίζει καλύτερα τις κινήσεις , ακόμη και τις μεγάλες που παρατηρούνται στα διαδοχικά καρέ μας, καθώς λαμβάνει υπόψη για τους υπολογισμούς της τις κλιμακωμένες εικόνες και έτσι εντοπίζει κίνηση στην ουσία με αυτόν τον τρόπο σε μεγαλύτερο έυρος από Pixels.

2.3 Υπολογισμός της μετατόπισης του Χεριού απο το Πεδίο Οπτικής Ροής

Στα δύο προηγούμενα ερτωτήματα υπολογίσαμε το πεδίο οπτικής ροής με απλή μέθοδο Lucas-Kanade και με Multiscale μέθοδο Lucas-Kanade. Στο συγκεκριμένο ερώτημα θα εκμεταλλευτούμε τα αποτελέσματα των μεθόδων αυτών και θα υπολογίσουμε τη μετατόπιση του Bounding Box ώστε να συμβαδίζει με την κίνηση του χεριού της εικονιζόμενης. Είναι εύκολο να παρατηρήσουμε ότι τα διανύσματα του πεδίου της οπτικής ροής έχουν μεγαλύτερο μήκος σε σημεία που ανήκουν σε περιοχές με έντονη πληροφορία υφής (π.χ. ακμές, κορυφές) και σχεδόν μηδενικό μήκος σε σημεία που ανήκουν σε περιοχές με ομοιόμορφη και επίπεδη υφή. Λόγω αυτού ο υπολογισμός της μέσης τιμής των διανυσμάτων μετατόπισης μας οδηγεί σε ανακριβή αποτελέσματα. Αντ'αυτού θα χρησιμοποιήσουμε ως κριτήριο την ενέργεια. Θα υπολογίσουμε εδώ, δηλαδή, τη μέση τιμή των διανυσμάτων μετατόπισης που έχουν ενέργεια μεγαλύτερη από μια τιμή κατωφλίου.

Ο τύπος που μας δινει τη ενέργεια αυτή είναι : ‖d‖2=d x 2+d y 2 .
Για την υλοποίηση των παραπάνω φτιάξαμε τη συνάρτηση με όνομα displ(d_x,d_y) που παίρνει σαν όρισμα την οπτική ροή που υπολογίστηκε για το bounding box και επιστρέφει τη μετατόπιση των συντεταγμενων της γωνίας αυτού ώστε να ακολουθεί με σωστό τρόπο την κίνηση. Η συνάρτηση αυτή κάνει τα εξής :

i. Υπολογίζει αρχικά την ενέργεια απο τα d_x d_y .

ii. Βρίσκει το μέγιστο αυτής .

iii. Ορίζει ένα κατώφλι (Πειραματιστήκαμε αρκετά με τις τιμές για να βρούμε την κατάλληλη, καθώς οι μεγάλες τιμές κατωφλίου δεν λάμβαναν υπόψη ακόμη και σημαντικές συνιστώσες της κίνησης,δηλαδή σημεία με σημαντική τιμή ενέργειας, ενώ οι πολύ χαμηλές συμπεριλάμβαναν στο μέσο όρο που θα υπολογίσουμε στη συνέχεια ακομή και σημεία της εικόνας στα οποία δεν παρατηρούταν ή υπήρχε ελάχιστη κινητική συμπεριφορά. Επιλέξαμε το 40% της μέγιστης ενέργειας ως κατώφλι).

iv. Κρατάει τις τιμές του πίνακα της ενέργειας που ξεπερνουν αυτό το κατώφλι ως bool πίνακα (1,0) .

v. Κρατάει με πολλαπλασιασμό τις αντίστοιχες τιμες των d_x d_y, που αντιστοιχούν δηλαδή στα συγκεκριμένα Pixels κίνησης και

vi. τέλος , υπολογίζει το μέσο όρο των στοιχείων του d_x και d_y τα οποία εκφράζουν την μετατόπιση που πρέπει να κάνει το Bounding Box.

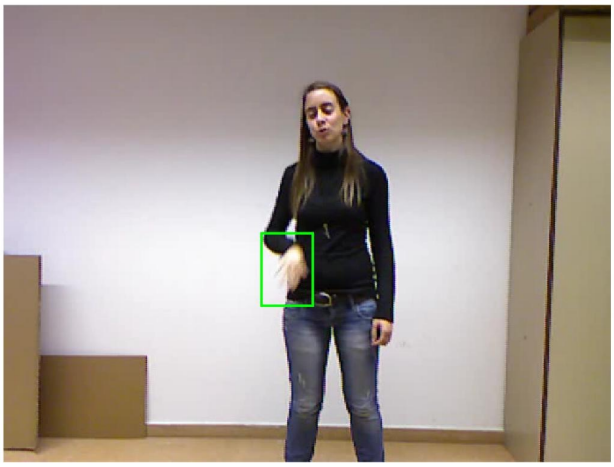
Στη συνέχεια παρατίθενται ενδεικτικά τα αποτελέσματα των συναρτήσεών μας για 4 frames. Δίδονται τόσο η εικόνα με το bounding box καθώς και η ενέργεια και η οπτική ροή όπως μας προέκυψαν αρχικά απο τη Singlescale Lucas-Kanade μέθοδο και στη συνέχεια από την Multiscale Lucas-Kanade μέθοδο.Ακόμη περισσότερα αποτελέσματα υπάρχουν στους παρακάτω φακέλους που υπάρχουν στο zip αρχείο μας (οι φάκελοι αυτοί πρέπει να υπάρχουν στο directory που βρίσκονται τα script μας καθώς εκει έχουμε ρυθμίσει να αποθηκεύονται τα αποτελέσματα που προκύπτουν απο την εκτέλεσή τους) :

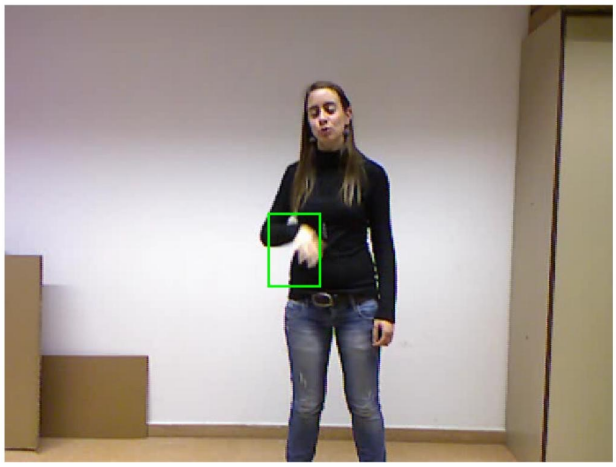
- **resuls-handrec**
- **results-energy**
- **results-opticalflow**

Ενδεικτικά επιλέγω τα παρακάτω frames για να σας τα παρουσιάσω :

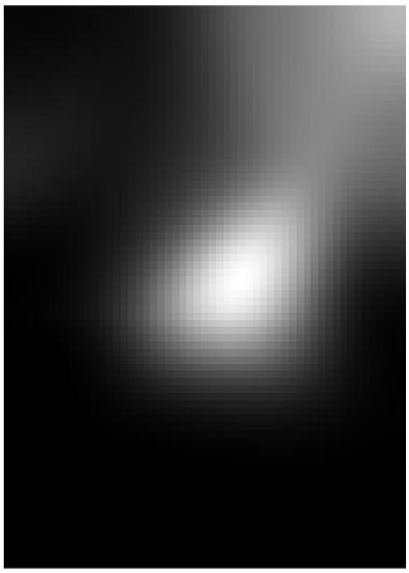
Για τα frame 5 και 6 :

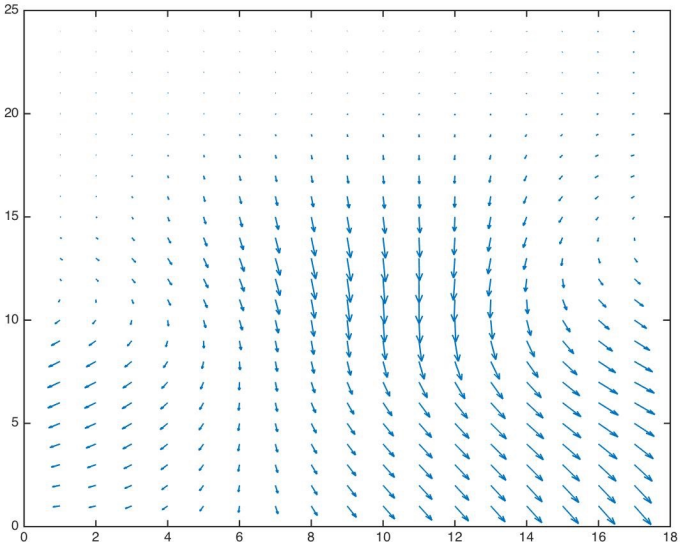
- **Απλός Lucas-Kanade**:



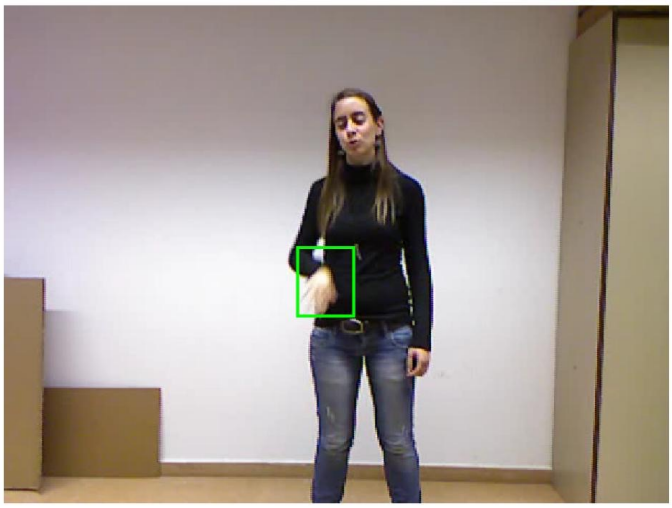


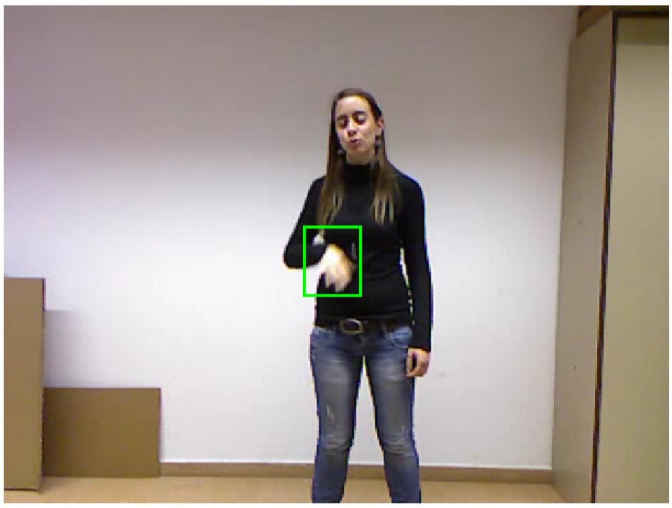
- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) απλού Lucas-Kanade:**



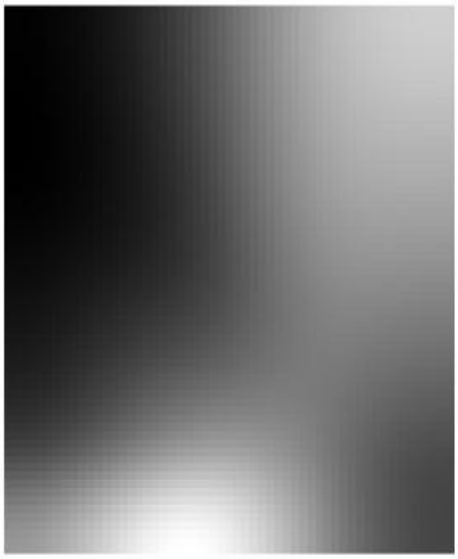


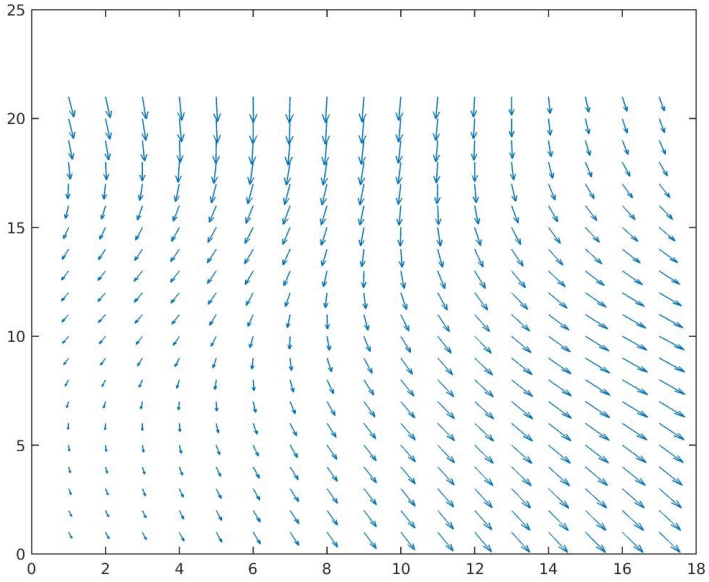
- **Multiscale Lucas-Kanade :**





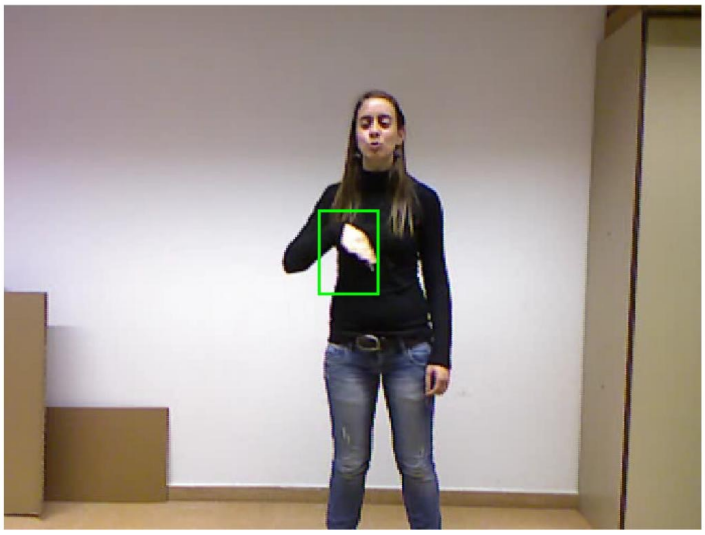
- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) Multiscale Lucas-Kanade:**

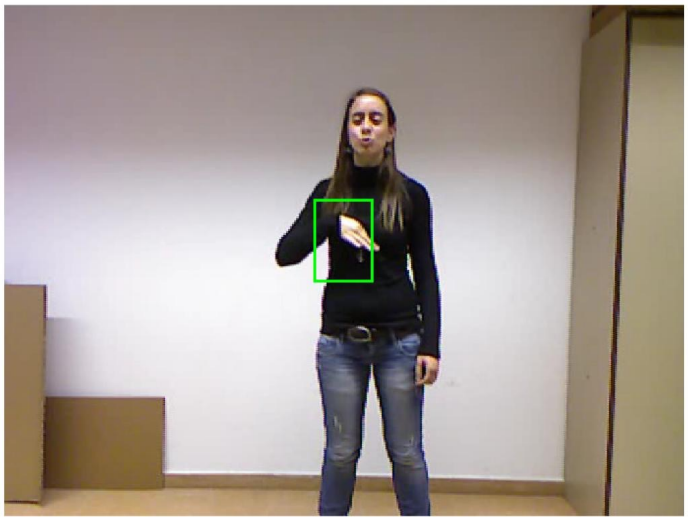




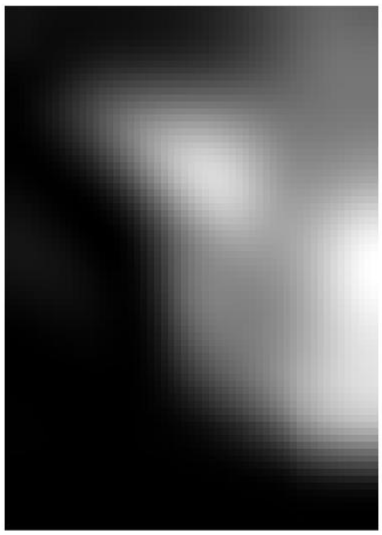
Για τα frame 14 και 15 :

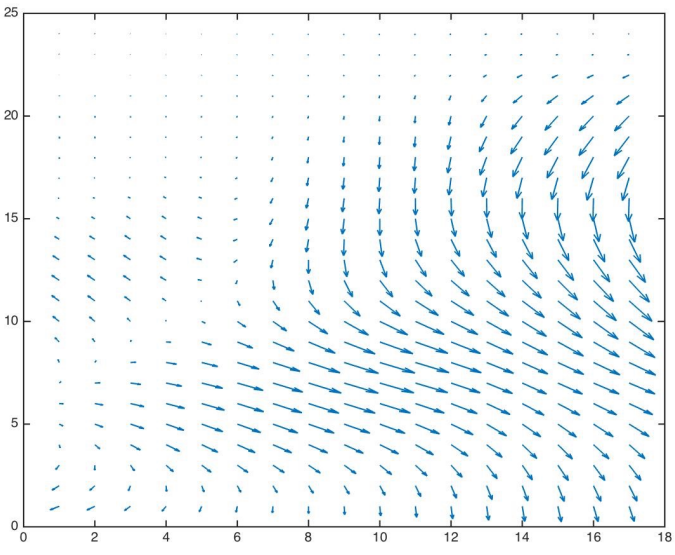
- **Απλός Lucas-Kanade**:



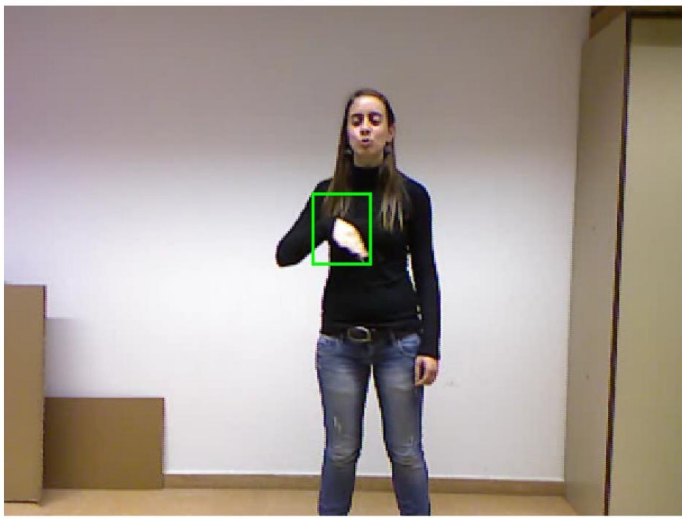


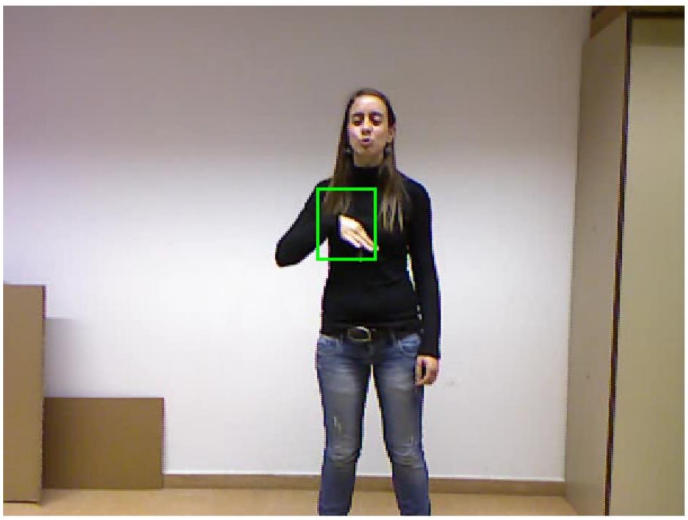
- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) απλού Lucas-Kanade:**



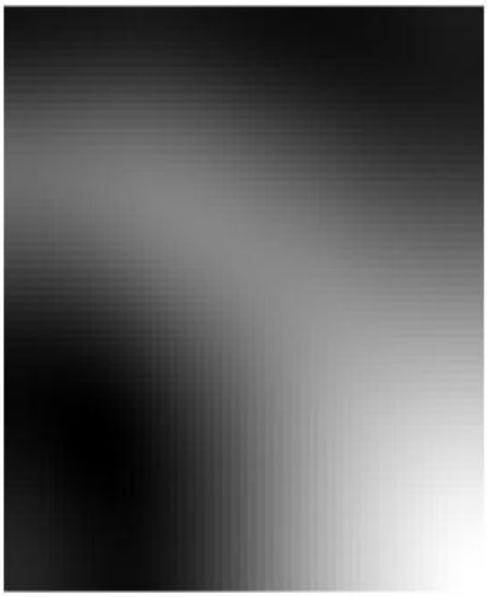


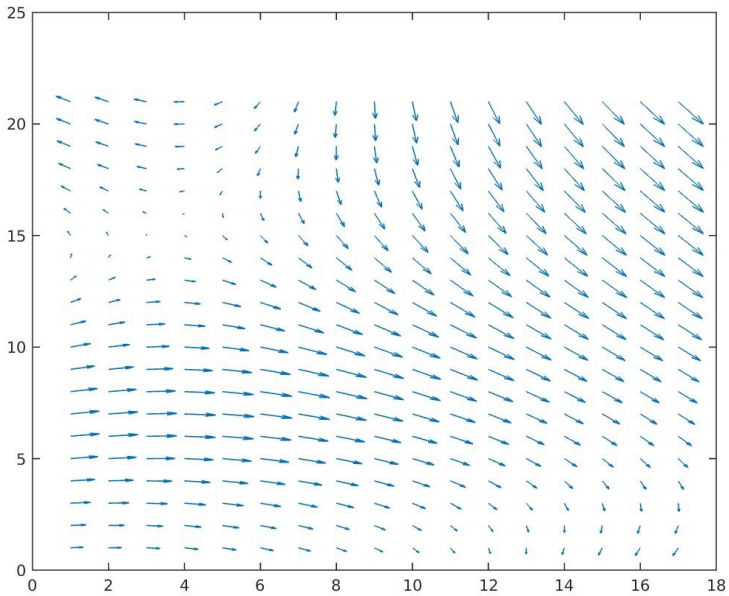
- **Multiscale Lucas-Kanade :**





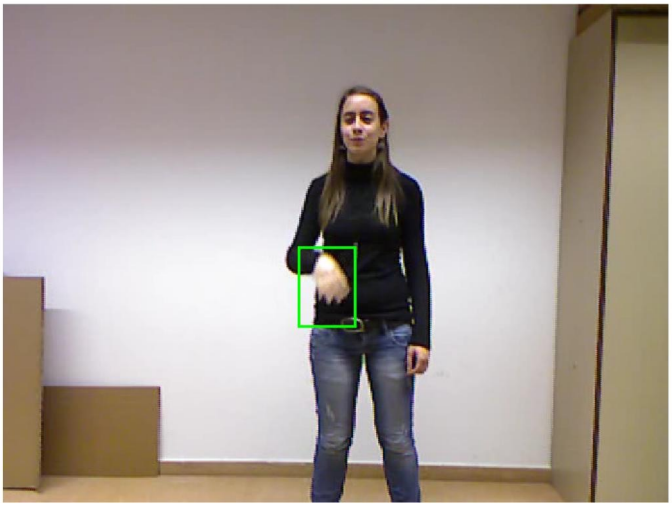
- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) Multiscale Lucas-Kanade:**

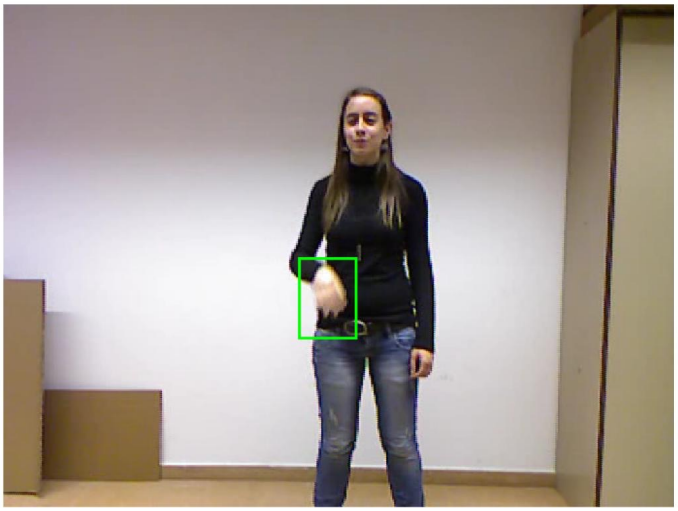




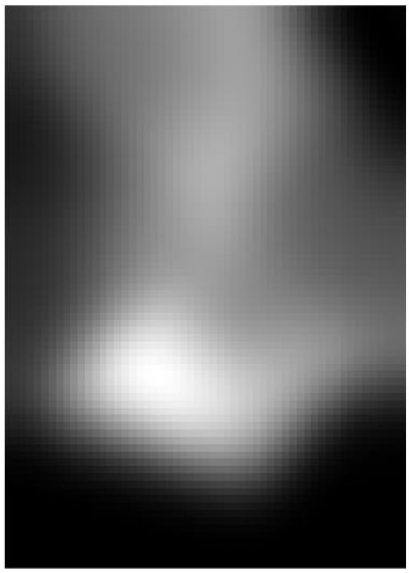
Για τα frame 35 και 36 :

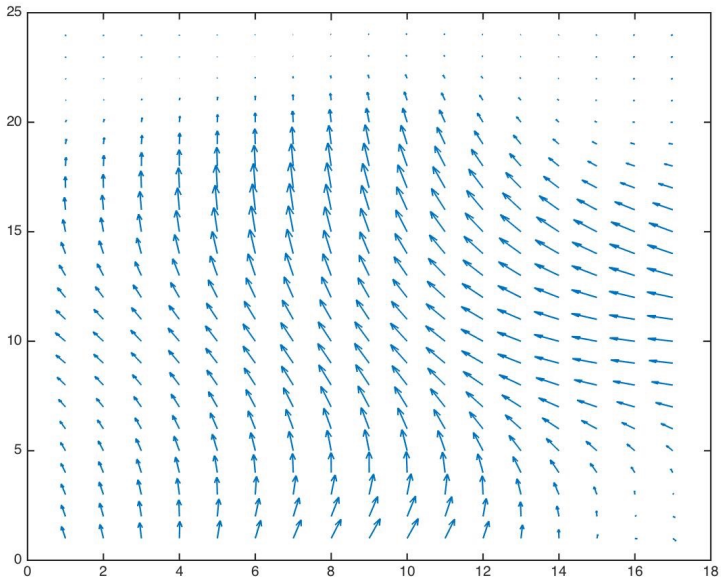
- **Απλός Lucas-Kanade**:



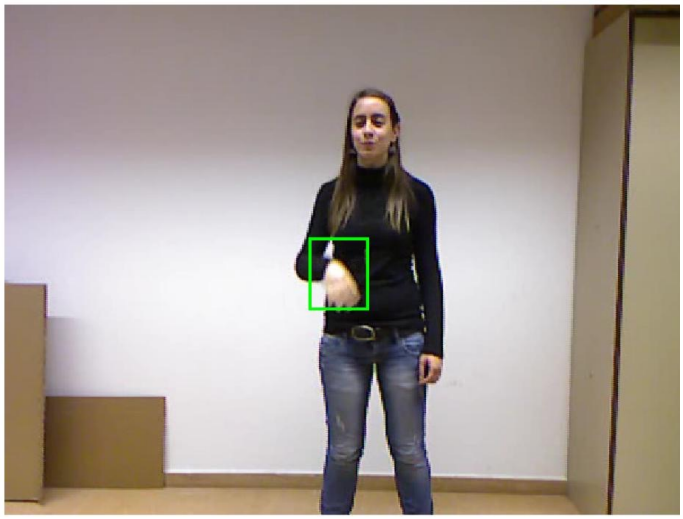


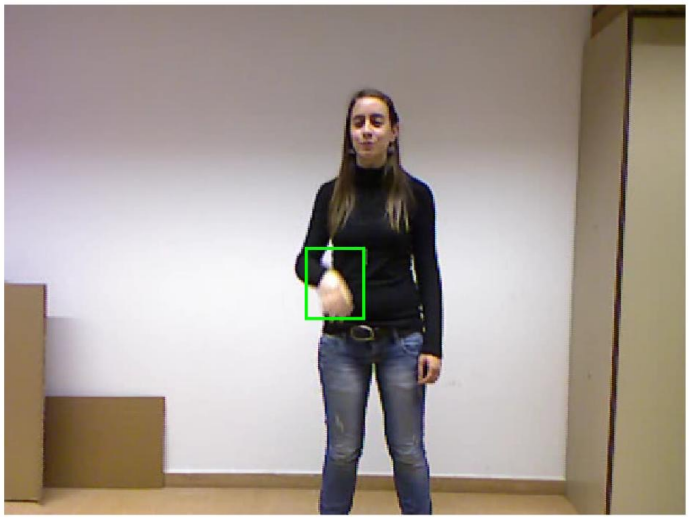
- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) απλού Lucas-Kanade:**



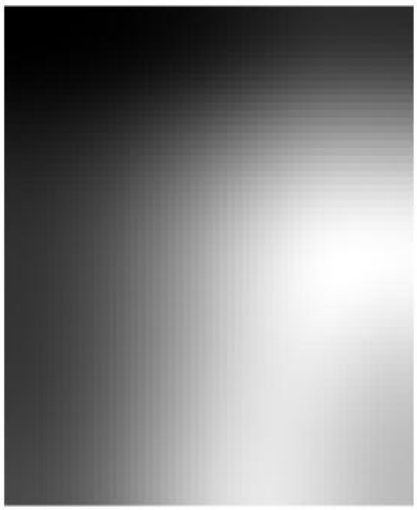


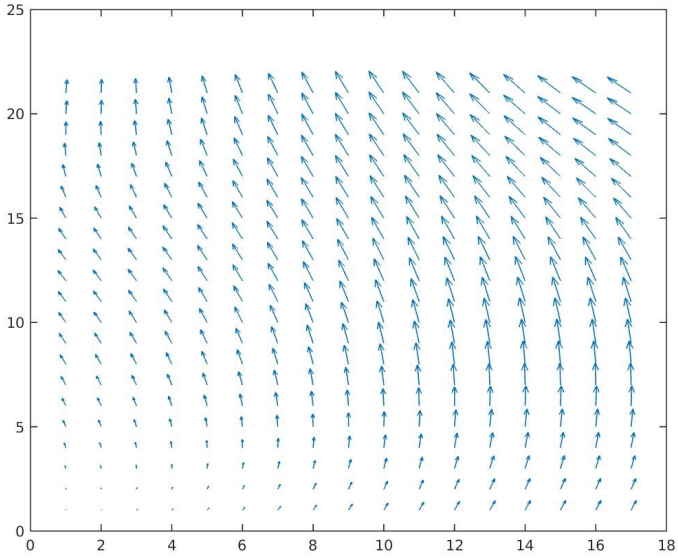
- **Multiscale Lucas-Kanade :**





- **Ενέργεια(αριστερά) και Οπτική Ροή(δεξιά) Multiscale Lucas-Kanade:**





Τα υπόλοιπα αποτελέσματα βρίσκονται μέσα στο zip αρχείο που σας στέλνω και μπορείτε να τα δείτε εκεί. Κάποια ίσως λείπουν απο εκεί λόγω περιορισμού που υπάρχει στο μέγεθος του zip αρχείου. Αν θέλετε κάποιο αποτέλεσμα που για τον παραπάνω λόγο δεν το έχω στο zip μπορείτε να τρέξετε τον κώδικα για να το πάρετε αλλά πρώτα διαβάστε το αρχείο με όνομα **README.txt.**