ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Εργασία Συστήματα Πολυμέσων 2018-2019

Μέλη ομάδας :

Π16097 ΔΙΟΝΥΣΗΣ ΝΙΚΑΣ,

Π16195 ΝΗΣΙΩΤΗΣ ΜΑΡΙΝΟΣ

Περιεχόμενα

Για την άσκηση 6.16 η αρχική εικόνα αλλά και η τελική μετά το πέρας της εκτέλεσης του πρόγραμματος, βρίσκονται στον ίδιο φάκελο με το εκτέλεσιμο. Για τις άλλες 3 ασκήσεις 6.17- 8.17- 8.18 τα αρχικά βίντεο βρίσκονται στον φάκελο original_videos, ενώ τα τελικά βίντεο μετά την εκτέλεση του προγράμματος βρίσκονται στον φάκελο final_videos. Και οι 2 φάκελοι βρίσκονται μέσα στον κεντρικο φάκελο Multimedia-Systems-2019

Άσκηση 6.16

1.1 Εκφώνηση άσκησης

«Σε αυτήν την προγραμματιστική άσκηση, θα υλοποιήσετε έναν απλό κωδικοποιητή μήκους διαδρομής, ο οποίος επεξεργάζεται δεδομένα εικόνας ή οποιονδήποτε δισδιάστατο πίνακα τιμών, κβαντίζοντας αρχικά τις τιμές εισόδου και στη συνέχεια εφαρμόζοντας σε αυτές κωδικοποίηση μήκους διαδρομής. Η κβάντιση θα τονίσει το φαινόμενο της επανάληψης τιμών. Το πρόγραμμά σας πρέπει να δέχεται ως είσοδο μια εικόνα αποχρώσεων του γκρι (ή μια έγχρωμη εικονα RGB) και μια τιμή που θα ρυθμίζει τις τιμές επιπέδων κβάντισης.»

1.2 Περιγραφή συστήματος/ψευδοκώδικας

Για την υλοποίηση και την επίλυση της άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python 3. Για να μπορεί να γίνει η εκτέλεση του προγράμματος θα πρέπει να υπάρχουν εγκατεστημένες στο σύστημα οι εξής βιβλιοθήκες: PIL & numpy.

Ψευδοκώδικας

Συνάρτηση κβάντισης(τιμή_κβάντισης, πίνακας_με_τα_pixels):
    Για 1 μέχρι το ύψος της εικόνας:
        Πίνακας_κβάντισης = διαίρεση των pixels με την τιμή_κβάντισης
    Εμφάνιση του *πίνακα_κβάντισης*
    Εμφάνιση της επεξεργασμένης εικόνας
    Επιστροφή του πίνακα_κβάντισης
Συνάρτηση κωδικοποίησης(πίνακας_κβάντισης):
    Κωδικοποίηση σε string τον πίνακα_κβάντισης
    Όσο i < ύψος της εικόνας:
    | Αν το i ==0 τότε:
    | Προηγούμενο = προηγούμενο_pixel_της στήλης[i, j-1]
    | Αλλιώς :
    | Προηγούμενο = προηγούμενο_pixel_της_γραμμής [i-1, j]
    | Όσο j <πλάτος της εικόνας:
    | | Παρόν = πίνακας_κβάντισης [i, j]
    | | Αν το παρόν == προηγούμενο:
    | | *Κωδικοποιήση* των στοιχείων σε string
    | | προηγούμενο=παρόν
    | | Πήγαινε στο επόμενο στοιχείο κατά πλάτος (στήλη)
    | Πήγαινε στο επόμενο στοιχείο κατά ύψος (γραμμή)
    Επιστροφή κωδικοποίηση
Συνάρτηση αποκωδικοποίησης (αποτέλεσμα_κωδικοποίησης):
    τέλος_γραμμής = ψευδής
    αποκωδικοποίηση = πίνακας_μηδενικών (ύψος, πλάτος)
    κωδικοποίηση_πίνακα = αποτέλεσμα_κωδικοποίησης(διαχωρισμός '|')

```
παρον = 0
προηγούμενη_στήλη = 0
Για i μέχρι κωδικοποίηση_πίνακα:
|   i = διαχωρισμός (‘-’)
|   εύρος_χρωμάτων = i[1] διαχωρισμός (‘;’)
|   στήλη_εκίννησης = int (εύρος_χρωμάτων[0]) #έλεγχος από που ξεκινάει η επόμενη στήλη
|   γραμμή_εκκίνησης = int(εύρος_χρωμάτων [1]) #έλεγχος από που ξεκινάει η επόμενη γραμμή
|   τιμές_χρώματος = i[0] διαχωρισμός(‘;’)
|   #έλεγχος αν υπάρχουν στοιχεία στην επόμενη στήλη
|   Αν παρόν <= πλάτος  & στήλη> προηγούμενη_στήλη:
|   |στήλη = προηγούμενη_στήλη
|   |   γραμμή_εκκίνησης = πλάτος
|   |   τέλος_γραμμής = αληθής
|
|       Όσο παρόν < γραμμή_εκκίνησης:
|       |   αποκωδικοποιήση [στήλης][παρόν] = πίνακα(τιμές_χρώματος)
|       |   Αν τέλος_γραμμής & εύρος_χρωμάτων[1] != 0 & και παρόν ==πλάτος -1
|       |   |   παρόν = 0
|       |   |   γραμμή_εκκίνησης = int(εύρος_χρωμάτων[1])
|       |   |   στήλη = int([0])
|       |   |   τέλος_γραμμής = ψευδής
|       |   αλλιώς:
|       |   |   παρόν +=1
|       |   προηγούμενη_στήλη = στήλη
Επιστροφή αποκωδικοποίηση

Συνάρτηση (από)-κβάντισης(τιμή_κβάντιστης, αποτέλεσμα_αποκωδικοποίησης):
    Για 1 μέχρι το ύψος της εικόνας:
        Πίνακας_(από)-κβάντισης = πολλαπλασιασμός των pixels με την τιμή_κβάντισης
    εικόνα = δημιουργία_εικόνας από τον πίνακας_(από)-κβάντισης
    Εμφάνιση εικόνας
    Επιστροφή πίνακας_(από)-κβάντισης
```

Κυρίως Πρόγραμμα

Φόρτωση εικόνας
Ύψος και πλάτος εικόνας
Πίνακας_με_τα_pixels
Εμφάνιση της εικόνας πρωτού την επεξεργασία
Πίνακας_κβάντισης = συνάρτηση κβάντισης(τιμή κβάντισης, πίνακας_με_τα _pixels)
Αποτέλεμσα_κωδικοποίησης = συνάρτηση κωδικοποίησης(πίνακας_κβάντισης)
Αποτέλεσμα_αποκωδικοποίησης = συνάρτηση αποκωδικοποίησης (αποτέλεσμα_κωδικοποίησης)
Νέα_είκονα = συνάρτηση (από)-κβάντισης(τιμή κβάντισης, απότελεσμα_αποκωδικοποίησης)
Εικόνα = δημιουργία_εικόνας(Νέα_εικόνα)
Εμφάνιση Εικόνα
Εγγραφή αποτελεσμάτων σε αρχείο .txt

1.3 Υλοποίηση

Αρχικά φορτώνουμε την εικόνα, και τις διαστάσεις της. Λαμβάνουμε τα εικονοστοιχεία (pixels) της εικόνας και τα αποθηκεύουμε σε μορφή πίνακα. Με την συνάρτηση quantization, διαιρούμε όλα τα εικονοστοιχεία με την τιμή κβάντισης και στρογγυλοποιούμε προς τα κάτω. Η συνάρτηση quantization δέχεται ως παραμέτρους, την τιμή κβάντισης και τον πίνακα με τα εικονοστοιχεία της αρχικής εικόνας. Επιστρέφει ως αποτέλεσμα τον νέο κβαντισμένο πίνακα.
Με την συνάρτηση encoder, υλοποιούμε τον κωδικοποιητή μήκους διαδρομής, όπου δέχεται ως είσοδο τον πίνακα που επέστρεψε η συνάρτηση quantization. Η κωδικοποίηση έχει γίνει ως εξής: διαβάζουμε τον πίνακα με τις κβαντισμένες τιμές, ανά γραμμή σημειώνοντας το αρχικό pixel. Όσο οι κβαντισμένες τιμές είναι ίδιες (λόγω της κβάντισης εμφανίζεται συχνά το φαινόμενο της επανάληψης) πηγαίνουμε στο επόμενο εικονοστοιχείο. Μόλις λάβουμε τιμή διαφορετική από την

προηγούμενη τότε, σημειώνουμε ως τέλος με το σύμβολο ‘|’, και κρατάμε την θέση στην οποία εμφανίστηκε η νέα τιμή. Αυτή η διαδικασία επαναλαμβάνεται όσο είναι το μέγεθος της εικόνας. Μόλις τελειώσει η διαδικασία, το αποτέλεσμα αποθηκεύεται με την μορφή ‘string’ και το επιστρέφει.

Ο αποκωδικοποιητής λαμβάνει ως παράμετρο το αποτέλεσμα που επέστρεψε η συνάρτηση encoder. Δημιουργούμε ένα πίνακα μηδενικών 3 διαστάσεων για τιμές RGB, στον οποίο θα αποθηκεύσει τις τιμές μετά την αποκωδικοποίηση. Αναθέτουμε την παράμετρο σε πίνακα όπου γίνεται ο διαχωρισμός βάση του συμβόλου ‘|’, δηλαδή σε κάθε σημείο στο οποίο άλλαζε η τιμή του εικονοστοχείου. Διαχωρίζουμε τον πίνακα με το σύμβολο ‘-‘, όπου δείχνει πόσα εικονοστοιχεία έχουν την ίδια τιμή κβάντισης, π.χ. *11;11;11-10;130,* σημαίνει πως από την θέση 10 μέχρι 130 έχουν τις RGB τιμές 11,11,11. Μόλις τελειώσει η διαδικασία αποκωδικοποίησης η συνάρτηση επιστρέφει τον πίνακα με τις RGB τιμές.

Αφού έχουμε κάνει την αποκωδικοποιήση, αυτό που μένει είναι να δημιουργήσουμε τον πίνακα με τα εικονοστοιχεία που είχε η αρχική εικόνα. Αυτό μπορεί να επιτευχθεί αν πολλαπλασιάσου με την ίδια τιμή, με την οποία κάναμε την κβάντιση. Προφανώς και το αποτέλεσμα θα είναι διαφορετικό, αφού κατά την διαδικασία της κβάντισης τα δεκαδικά μέρη τα οποία είχαν προκύψει από την διαίρεση χάθηκαν, και δεν υπάρχει τρόπος ανάκτησης τους σε αυτό το στάδιο. Επομένως θα υπάρχει μια μικρή διαφορά της αρχικής εικόνας με της κβαντισμένης.

Το αποτέλεσμα της κωδικοποιήσης το αποθηκεύουμε σε ένα ένα αρχείο .txt με το όνομα ‘*apotelesmata.txt’.*

1.4 Αποτελέσματα

Η εικόνα πριν την κβαντίση:

Κβαντισμένη εικόνα:

Εικόνα από την επαναφορά της κβάντισης:

1.5 Συμπεράσματα

Κβαντίζοντας την εικόνα ρίξαμε τα επίπεδα του γκρι, με αποτέλεσμα οι τιμές της κβαντισμένης εικόνας να είναι πιο κοντά στο μηδέν, για αυτό και έχει μειωθεί η ένταση του άσπρου. Αντιστρέφοντας την διαδικασία για να πάρουμε την αρχική εικόνα, λαμβάνουμε μια εικόνα η οποία δεν είναι ακριβώς όπως η αρχική, αλλά διαφορετική όπου υπάρχουν και

εμφανείς απώλειες στην ποιότητα της εικόνας. Αυτό συμβαίνει γιατί δεν μπόρεσαν να επανέλθουν ακριβώς οι αρχικές τιμές, αλλά με μικρή διαφορά προς τα κάτω. Στα κυκλωμένα σημεία παρατηρούμε την μεγαλύτερη διαφορά, που υπάρχει μεταξύ της αρχικής και της τελικής εικόνας. Οι κύκλοι δείχνουν ότι το σημείο δεν είναι το ίδιο λείο σε σχέση με το αρχικό, παρά υπάρχει μια αισθητή διαφορά και μείωση στην ποιότητα της εικόνας.

Μέγεθος αρχικής εικόνας 33 KB

Μέγεθος συμπιεσμένης εικόνας 11 KB

Υπάρχουν εμφανείς διαφορές στην ποιότητα της εικόνας αφού ο ουρανός δεν είναι το ίδιο «καθαρός» και έχει αλλοιωθεί παρουσιάζοντας την αλλοίωση σε αυτό που φαίνεται σαν «γραμμές»

Ερωτήματα 1,2,3 Βιβλίου σελίδα 215

1) Για τις παρεχόμενες εικόνες εισόδου (9 διαφορετικές εικόνες) ο **λόγος συμπίεσης** που λάβαμε ήταν **1.4** (με τιμή κβάντισης 10).

2) Όσο αυξάναμε την τιμή της κβάντισης λαμβάνουμε όλο και μικρότερους λόγους συμπίεσης, το οποίο είναι φυσιολογικό αφού οι τιμές των εικονοστοιχείων τείνουν όλο και περισσότερο στο μηδέν. Το μειονέκτημα σε αυτή την περίπτωση είναι πως αν χρησιμοποιηθεί μεγάλή τιμή κβάντισης, μετά θα έχουμε χάσει μεγάλο μέρος σημαντικής πληροφορίας και η ανακατασκευή της τελικής εικόνας θα διαφέρει πάρα πολύ από την κανονική.

3) Μπορούμε να αποθηκεύσουμε τις συμπιεσμένες εξόδους με κωδικοποίηση μήκους διαδρομής ως ένα αποτελεσματικότερο τρόπο διαβάσματος και επεξεργασίας σε μεταγενέστερο στάδιο. Βέβαια υπάρχουν και μειονεκτήματα που θα πρέπει να λάβουμε υπόψιν σε αυτή την περίπτωση, όπως για παράδειγμα την κωδικοποίηση επαναλήψεων μήκους 1. Σε αυτή την περίπτωση οδηγούμαστε σε χειρότερα αποτελέσματα διότι εκεί που είχαμε 1 σύμβολο, τώρα υπάρχουν 2 (ο δείκτης που δείχνει τον αριθμό των συμβόλων και το σύμβολο). Επομένως η κωδικοποίηση μήκους διαδρομής αποδίδει καλύτερα όταν υπάρχουν αρκετές επαναλήψεις συμβόλων στην πληροφορία.

Άσκηση 6.17

2.1 Εκφώνηση άσκησης

«Στην άσκηση αυτή, θα εφαρμόσετε τεχνικές DPCM για να συμπιέσετε βίντεο. Αν και αυτός δεν είναι ο πιο αποδοτικός τρόπος για τη συμπίεση βίντεο, αποτελεί εντούτοις μια καλή εισαγωγή στη συμπίεση βίντεο που θα ακολουθήσει στα επόμενα κεφάλαια. Το βίντεο μπορεί να οριστεί ως ακολουθία πλαισίων. Υπάρχει μεγάλη συνάφεια των δεδομένων από πλαίσιο σε πλαίσιο – δεν μεταβάλλονται δηλαδή όλα τα εικονοστοιχεία από πλαίσιο σε πλαίσιο, όπως για παράδειγμα συμβαίνει με τα εικονοστοιχεία του παρασκηνίου. Αυτό ακριβώς το γεγονός θα εκμεταλλευτείτε σε αυτήν την άσκηση. Εδώ σας ζητείτε η συγγραφή προγράμματος το οποίο θα διαβάσει μια ακολουθία πλαισίων βίντεο ως είσοδο, μαζί με μία παράμετρο κβάντισης. Το καθήκον σας είναι η υλοποίηση ενός αλγορίθμου παρόμοιου με τον DPCM, που θα αφήνει το πρώτο πλαίσιο ως έχει, αλλά θα υπολογίζει τις διαφορές μεταξύ διαδοχικών πλαισίων και θα κβαντίζει κάθε τιμή της διαφοράς των εικονοστοιχείων με τρόπο εξαρτώμενο από την τιμή κβάντισης που δέχεται ως είσοδο.»

2.2 Περιγραφή συστήματος/ψευδοκώδικας

Για την υλοποίηση και την επίλυση της άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python 3.
Για να μπορεί να γίνει η εκτέλεση του προγράμματος θα πρέπει να υπάρχουν εγκατεστημένες στο σύστημα οι εξής βιβλιοθήκες: cv2 & numpy.

Ψευδοκώδικας

Απαραίτητες ρυθμίσεις για την ανάγνωση και εγγραφή βίντεο με τη βοήθεια της βιβλιοθήκης cv2

Όσο υπάρχουν πλαίσια από το βίντεο για να διαβάσουμε:
| Εικόνα = βιντεο.διαβασεΕπόμενηΕικόνα()
| Εικόνες.πρόσθεσε(εικόνα)
| Πλήθος_εικόνων += 1

Εικόνες = Εικόνες.numpyArray()
Διαστάσεις_εικόνας = Εικόνες.διαστάσεις()

Εμφάνισε(«Δώσε παράμετρο κβάντισης»)
Δώσε(Παραμετρο_κβάντισης)

Για i από 1 μέχρι Πλήθος_εικόνων:
| Πίνακας_διαφορών(i) = Εικόνες(i) – Εικόνες(i-1)
| Κβαντισμένες_εικόνες(i) = Στρογγυλοποίησε(Πίνακας_διαφορών(i)/ Παράμετρο_κβάντισης)
|

Για i από 1 μέχρι Πλήθος_εικόνων -1:
| Ανακατασκευασμένες εικόνες(i) = Κβαντισμένες εικόνες(i) +
| Κβαντισμένες εικόνες (i+1)
| Γράψε εικόνα στο αρχείο

2.3 Υλοποίηση

- Ανάγνωση βίντεο

Αρχικά φορτώνουμε το βίντεο, και παίρνουμε τις διαστάσεις του. Μετά δηλώνουμε αντίστοιχα τις ιδιότητες και τη κωδικοποίηση του βίντεο που θα δημιουργήσουμε και τις διαστάσεις. Μετά αρχίζουμε και διαβάζουμε το βίντεο πλαίσιο προς πλαίσιο και για κάθε πλαίσιο το βάζουμε προσωρινά σε μία λίστα ως λίστα των αριθμητικών τιμών των pixels.

- Υπολογισμός διαφορών διαδοχικών πλαισίων

Αφού διαβάσουμε όλα τα πλαίσια του βίντεο μετατρέπουμε τη λίστα που αναφέραμε σε numpy array για να μας βοηθήσει στις πράξεις μεταξύ πινάκων. Έχοντας πια τα πλαίσια του βίντεο σε πίνακες numpy array για κάθε πλαίσιο παίρνουμε διαδοχικά αυτό και το αμέσως προηγούμενο του και υπολογίζουμε το πλαίσιο διαφορών τους και αποθηκεύουμε το αποτέλεσμα στην θέση που βρισκόταν το πλαίσιο που ελέγχαμε εκείνη την στιγμή.

- Κβάντιση των τιμών των πλαισίων

Έπειτα υπολογίζουμε τις κβαντισμένες τιμές κάθε πλαισίου διαφορών διαιρώντας κάθε κβαντισμένο πλαίσιο με τη παράμετρο κβάντισης και στρογγυλοποιώντας τις πιθανές δεκαδικές τιμές που προκύπτουν.

- Ανακατασκευή των πλαισίων

Τέλος ανακατασκευάζουμε κάθε πλαίσιο εικόνας από τα κβαντισμένα πλαίσια εικόνων προσθέτοντας το κάθε κβαντισμένο πλαίσιο με το επόμενο διαδοχικό του και «γράφοντας» το στο αρχείο βίντεο που αναφέραμε στην αρχή.

2.4 Αποτελέσματα

Το αρχικό μας βίντεο:

Με μέγεθος

12,9 MB

Εδώ έχουμε τα αποτελέσματα με παράμετρο κβαντοποίησης:

- K = 2

Με μέγεθος:

10,3 MB

```
Enter quantization parameter:
2
106
Video Compression is complete
The compression ratio is:  1.2522133543182916
```

- K = 4

Με μέγεθος:

7 MB

```
Enter quantization parameter:
4
106
Video Compression is complete
The compression ratio is:  1.8412833630223198
```

- K = 8

```
Enter quantization parameter:
8
106
Video Compression is complete
The compression ratio is:  2.7859053206624584
```

Με μέγεθος:

4,6 MB

2.5 Συμπεράσματα και απαντήσεις

Βλέποντας το αρχικό βίντεο και το συμπιεσμένο βίντεο και συγκρίνοντας μεταξύ τους, βλέπουμε ότι στο συμπιεσμένο βίντεο έχοντας κβαντίσει το κάθε πλαίσιο ρίξαμε τα επίπεδα των χρωμάτων (RGB), με αποτέλεσμα οι τιμές των κβαντισμένων πλαισίων να είναι πιο κοντά στο μηδέν, για αυτό και έχει μειωθεί η ένταση της φωτεινότητας. Αντιστρέφοντας την διαδικασία για να πάρουμε την αρχική εικόνα σε κάθε πλαίσιο, λαμβάνουμε μια εικόνα η οποία δεν είναι ακριβώς όπως η αρχική, αλλά διαφορετική και λίγο υπό-φωτισμένη. Αυτό συμβαίνει γιατί δεν μπόρεσαν να επανέλθουν ακριβώς οι αρχικές τιμές, αλλά με μικρή διαφορά προς τα κάτω. Καθώς ανεβάζουμε τη παράμετρο κβάντισης οι συνέπειες αυτές θα είναι όλο και πιο φανερές καθώς και το μέγεθος του αρχείου θα μικραίνει όλο και περισσότερο λόγο της συμπίεσης και της μεγαλύτερης παραμέτρου κβαντοποίησης.

- <u>Λόγος συμπίεσης</u>

Από τα αποτελέσματα πριν συμπεραίνουμε ότι ο λόγος συμπίεσης που είχαμε με αυτά τα δεδομένα ήταν ανάμεσα στο [1.25-2.78] με παράμετρο κβαντισης K = [2,8].

- <u>Πρόβλημα σε πραγματικό σενάριο</u>
  Κατά τον υπολογισμό διαφορών διαδοχικών πλαισίων και κβάντιση της τιμής αυτών, υπάρχει ένα μικρό σφάλμα, το οποίο συσσωρεύεται και μεταδίδεται και στα επόμενα πλαίσια. Αυτό το σφάλμα μετά από κάποιες επαναλήψεις θα είναι αρκετά μεγάλο και δεν θα μπορούμε να ανακατασκευάσουμε σωστά το πλαίσιο. Ένας τρόπος που μπορούμε να το αποφύγουμε αυτό είναι μετά από κάποιο αριθμό πλαισίων P (τα οποία κωδικοποιούνται με τεχνική πρόβλεψης, άρα λίγα δυαδικά ψηφία) να βάλουμε πλαίσιο

I, όπου το συγκεκριμένο είναι κωδικοποιημένο ως ανεξάρτητη εικόνα. Η κωδικοποιήση του μπορεί να γίνει με JPEG διαδικασία. Αυτό όμως απαιτεί περισσότερα δυαδικά ψηφία σε σχέση με το πλαίσιο P, και περισσότερο χώρο.

Άσκηση 8.17

3.1 Εκφώνηση άσκησης

«Στην άσκηση αυτή, θα υλοποιήσετε την τεχνική της αντιστάθμισης κίνησης και θα μελετήσετε πως επηρεάζει τα σφάλματα πρόβλεψης. Υποθέστε ότι το πρώτο θα είναι πάντα ένα πλαίσιο I και ότι το υπόλοιπα πλαίσια θα είναι τύπου P. Η υπόθεση αυτή στην περίπτωση σύντομων ακολουθιών βίντεο που επεξεργάζεστε, που έχουν μήκος το πολύ 100 πλαισίων.»

1ο μέρος άσκησης:
«Στο πρώτο μέρος της άσκησης υποθέστε ότι θέλετε να προβλέπετε ολόκληρα P πλαίσια και όχι κατά τμήματα. Η πρόβλεψη κάθε ολόκληρου πλαισίου γίνεται με βάση το προηγούμενο πλαίσιο. Υλοποιήστε μια διαδικασία που δέχεται είσοδο δύο πλαίσια, υπολογίζει τη διαφορά τους και επιστρέφει ένα πλαίσιο σφαλμάτων. Δεν υπολογίζετε διάνυσμα κίνησης. Να προβάλετε τα πλαίσια σφαλμάτων.»

2ο μέρος άσκησης:
«Στο δεύτερο θα υλοποιήσετε τεχνική πρόβλεψης κίνησης, η οποία υπολογίζει διανύσματα κίνησης ανά μπλοκ. Κάθε μπλόκ έχει το τυπικό MPEG μέγεθος 16X16. Υλοποιήστε μία συνάρτηση που δέχεται είσοδο δύο πλαίσια: ένα πλαίσιο αναφοράς, το οποίο θα χρησιμοποιηθεί κατά την αναζήτηση των διανυσμάτων κίνησης, και ένα πλαίσιο στόχο, το οποίο θα προβλεφθεί. Διαιρέστε το πλαίσιο στόχο σε μακρομπλόκ μεγέθους 16X16. Εάν το πλάτος και ύψος του πλαισίου δεν είναι πολλαπλάσια του 16, συμπληρώστε κατάλληλα το πλαίσιο με μαύρα εικονοστοιχεία. Για κάθε μπλοκ στο πλαίσιο-στόχο, ανατρέξτε στην αντίστοιχη θέση στο πλαίσιο αναφοράς και βρείτε την περιοχή που δίνει το καλύτερο ταίριασμα, όπως έχει

εξηγηθεί στο κείμενου του κεφαλαίου. Χρησιμοποιήστε τη μετρική SAD σε περιοχή αναζήτησης που προκύπτει για k=16, έτσι ώστε τα διανύσματα κίνησης να έχουν μέγεθος το πολύ 16 εικονοστοιχείων ως προς κάθε κατεύθυνση. Με βάση το μπλοκ πρόβλεψης, υπολογίστε το μπλοκ σφαλμάτων ως τη διαφορά μεταξύ του αρχικού μπλόκ και του προβλεφθέντος. Αφού αυτή η διαδικασία ολοκληρωθεί για όλα τα μπλοκ, θα προκύψει ένα πλαίσιο σφαλμάτων. Να γίνει η προβολή όλων των πλαισίων σφαλμάτων»

3.2 Περιγραφή συστήματος/ψευδοκώδικας

Για την υλοποίηση και την επίλυση της άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python 3.
Για να μπορεί να γίνει η εκτέλεση του προγράμματος θα πρέπει να υπάρχουν εγκατεστημένες στο σύστημα οι εξής βιβλιοθήκες: scipy, numpy & cv2.

Ψευδοκώδικας

1ο μέρος Άσκησης

```
Ρυθμίσεις και ιδιότητες βίντεο
|
|       αντικείμενο για ανάγνωση βίντεο ()
|       ύψος πλαισίου ()
|       μήκος πλαισίου ()
|       κωδικοποίση βίντεο ()
|       αρχείο για εγγραφή του βίντεο ()
|
Τέλος ρυθμίσεων και ιδιοτήτων του βίντεο

Προηγούμενο πλαίσιο = Διάβασε πρώτο πλαίσιο I ()
Προηγούμενο πλαίσιο = Μετατροπή σε ασπρόμαυρο (Προηγούμενο πλαίσιο)
```

Όσο υπάρχουν πλαίσια P για διάβασμα:
| Τωρινό πλαίσιο = Διάβασε επόμενο πλαίσιο P ()
| Αν δεν υπάρχει επόμενο πλαίσιο
| | ΕΞΟΔΟΣ από loop
| Αλλιως
| | Τωρινό πλαίσιο = Μετατροπή σε ασπρόμαυρο (Τωρινό πλαίσιο)
| Υπολόγισε πλαίσιο σφαλμάτων με το προηγούμενο πλαίσιο()
| Γράψε στο καινούριο βίντεο το πλαίσιο σφαλμάτων ()
| Προηγούμενο πλαίσιο = Τωρινό πλαίσιο

Υπολόγισε εντροπία του καινούριου βίντεο ()

2ο μέρος Άσκησης

Μέθοδοι / συναρτήσεις προγράμματος

Συνάρτηση χωρισμού σε μακρομπλόκς(πλήθος_γραμμών, πλήθος_στηλών, πίνακας_εικόνας):
Μπλοκς = []
Για k από 0 μέχρι το (πλάτος_εικόνας - πλήθος_γραμμών+1) με βήμα = πλήθος_γραμμών :
| Για j μέχρι (ύψος_εικόνας - πλήθος_στηλών+1) με βήμα = πλήθος_στηλών:
| μακρομπλόκ = πίνακας [k:k+πλήθος_γραμμών, c:c+πλήθος_στηλών]
| Μπλοκς.πρόσθεσε(μακρομπλόκ)
μετατροπή του Μπλοκς σε πίνακα numpy array
Επιστροφή πίνακα Μπλοκς

Συνάρτηση προσθήκη_μαύρων_εικονοστοιχείων(εικόνα):
Ύψος_εικόνας
Πλάτος_εικόνας
προσθήκη_μαύρων_εικονοστοιχείων = μετατροπή_σε_πίνακα(μαύρο_pixel() * (((πλάτος-1 // 16+1)* 16)- πλάτος))
Για i μέχρι μέγεθος_εικόνας
|γραμμή = μετατροπή_σε_πίνακα(i, προσθήκη_μαύρων_εικονοστοιχείων, άξονας_γραμμών)
| προσωρινή_εικόνα = γραμμή
προσθήκη_μαύρης_γραμμής = μετατροπή_σε_πίνακα(μαύρο_pixel() *

(((πλάτος-1 // 16+1) * 16)- πλάτος))
	Για j μέχρι (((πλάτος-1 // 16+1) * 16)- πλάτος)):
		προσωρινή_εικόνα = προσθηκη_μαύρης_γραμμής
	Επιστροφή μετατροπή_σε_πίνακα(προσωινή_εικόνα)

Συνάρτηση μαύρο_εικονστοιχείο():
	Επιστροφή [0]

Συνάρτηση SAD(εικόνα 1 , εικόνα 2, μακρομπλόκ**):**
	i = μακρομπλόκ
	γειτονικα_Μπλόκ = []
	διαφορές_Μπλόκ = []
	γειτονικά_μπλόκ.πρόσθεσε(μακρομπλόκ)
	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i]))
	Αν i+1 <= πλάτος_εικόνας: #δεξί μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i+1]))
	Αν i-1 >= 0: #αριστερό μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i+1]))
	|
	Αν i+πλάτος_εικόνας <= (πλάτος_εικόνας)^2: #απο κάτω μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i+πλάτος_εικόνας]))
	|
	Αν i+πλάτος_εικόνας-1 <= (πλάτος_εικόνας)^2:#διαγώνια κάτω αριστερά μάκρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i+ πλάτος_εικόνας-1]))
	|
	Αν i+πλάτος_εικόνας + 1 <= (πλάτος_εικόνας)^2:#διαγώνια κάτω δεξία μακρομπλόκ
	|
	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i],εικόνα1[i+1]))
	Αν i-πλάτος_εικόνας >= 0: #πάνω μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i-πλάτος_εικόνας]))
	|
	Αν i-πλάτος_εικόνας - 1 >= 0: #διαγώνια πάνω αριστερά μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i-πλάτος_εικόνας-1]))
	|
	Αν i-πλάτος_εικόνας +1 >= 0: #διαγώνια πάνω δεξιά μακρομπλόκ
	|	διαφορές_μπλόκ.πρόσθεσε(απόλυτο_άθροισμα_διαφορών(εικόνα2[i], εικόνα1[i-πλάτος_εικόνας+1]))

```
        |
Επιστροφή -
        γειτονικά_μπλόκ[μακρομπλόκ_με μικρότερο_άθροισμα_απόλυτων_διαφορών]


Συνάρτηση άθροισμα_απόλυτης_διαφοράς(μακρομπλόκ1,μακρομπλόκ2):
        άθροισμα = 0
        N = πλάτος(μακρομπλόκ1)
        Για i από 0 μέχρι N
        |       Για j από 0 μέχρι N
                |       |       άθροισμα = άρθοισμα + απόλυτη_τιμη(μακρομπλόκ1[i,j] -
                μακρομπλόκ2[i,j])
        Επιστροφή άθροισμα

Συνάρτηση ιεραρχική_αναζήτηση(πίνακα):
        πίνακα1 = μετατροπή_σε_πίνακα()
        x, y, = μορφή_πίνακα /* να έχουμε τις συντεταγμένες σε μορφή πίνακα*/
        πίνακα2 = []  /*αρχικοποιήση του πίνακα2 σαν λίστα*/
        Για i από 0 μέχρι x με_βήμα 2:
        |       Για j από 0 μέχρι y με_βήμα 2:
        |       |       πίνακα2 = πίνακα1([x], [y])  /*ο πίνακα2 είναι σε μορφή λίστας */
      πίνακα2 = μετατροπή_σε_πίνακα()
        πίνακα2 = σχηματισμός(x/2, y/2) /* για να αποκτήσει τις διαστάσεις που
θέλουμε */
        Επιστροφή πίνακα2

Συνάρτηση ανάλυση_σε_επίπεδα (εικόνα ανφοράς, εικόνα στόχος):
        εικόνα ανφοράς = μετατροπή_σε_πίνακα()
        εικόνα στόχος = μετατροπή_σε_πίνακα()
        Για i μέχρι 2: //τα επίπεδα
        |       ιεραρχική_εικόνα1 = ιεραρχική_εικόνα1 + [ιεραρχική_αναζήτηση(εικόνα
        αναφοράς)]
        |       εικόνα αναφοράς = ιεραρχική_αναζήτηση(εικόνα αναφοράς)
        |       ιεραρχική_εικόνα2 = ιεραρχική_εικόνα2 + [ιεραρχική_αναζήτηση(εικόνα
        στόχος)]
        |       εικόνα στόχος = ιεραρχική_αναζήτηση(εικόνα στόχος)

Συνάρτηση βελτιστοποίηση_Επιπέδων(blocks_move, ιεραρχική_εικόνα1,
ιεραρχική_εικόνα2):
        size_block = [8, 16]
        Για k μέχρι 2:
        |       /* διαίρεση της μικρής εικόνας σε blocks μεγέθους size_block  *
        size_block */
        |       εικόνα1 = blocks(size_block [k], size_block [k],
                ιεραρχική_εικονα1[size_block [1- k])
        |       εικόνα2 = blocks(size_block [k], size_block [k],
                ιεραρχική_εικονα2[size_block [1- k])
        |       blocks_χωρίς_κίνηση=[]
```

```
|       Για i μέχρι το μέγεθος(blocks_move):/*ελέγχουμε μόνο τα blocks στα
        οποία βρήκαμε κίνηση*/
|       |       Αν εύρεση_κίνησης (εικόνα1[blocks_move[i]] –
        εικονα2[blocks_move[i]]):
|       |       |       Συνέχισε
|       |       Αλλιώς:
|       |       |       blocks_χωρίς_κίνηση = blocks_χωρίς_κίνηση  [i] /* αν δεν
        υπάρχει κίνηση
|       |       |
βάζουμε το block στην λίστα*/
|       blocks_move =[αν κάποιο block_χωρίς_κίνηση δεν είναι στη λίστα
                blocks_moves,αφαίρεσε το]
```

Επιστροφή εικόνα1, εικόνα2, blocks_move

Συνάρτηση εύρεση_κίνησης(πίνακα):

```
πίνακα1 = μετατροπή_σε_πίνακα()
x, y = μορφή_πίνακα
πλήθος_μηδενικών_στοιχείων = x * y – μη_μηδενικά_στοιχεία(πίνακα1)
Αν πλήθος_μηδενικών_στοιχείων >= 0.8 *x *y:
        Επέστρεψε 0
Αλλιώς:
        Επέστρεψε 1
```

Συνάρτηση ανακατασκευή_εικόνας(x, y, εικόνα2):

```
ρ =1
Για i μέχρι x:
|  έξοδος = μετατροπή_σε_πίνακα(εικόνα2 [i *y])  /* για να πάρουμε τις
διαστάσεις
|                                                       της κάθε εικόνας*/
|  Για j μέχρι (y-1):
|  |       έξοδος = ένωσε(έξοδο, εικόνα2[ρ], άξονα_στηλών)
   |       |       ρ +=1
   |     ρ +=1
   |       Αν i == 0:
   |               ανακατασκευασμένη_εικόνα = έξοδο
   |       Αλλιώς:
   |               ανακατασκευασμένη _εικόνας = ένωσε((εμφάνιση_εικονας,
έξοδο), άξονας_γραμμών)
```

Επιστροφή ανακατασκευασμένη _εικόνας

Κυρίως πρόγραμμα

Ρυθμίσεις και ιδιότητες βίντεο
|
| αντικείμενο για ανάγνωση βίντεο ()
| ύψος πλαισίου ()
| μήκος πλαισίου ()
| κωδικοποίση βίντεο ()
| αρχείο για εγγραφή του βίντεο ()
|
Τέλος ρυθμίσεων και ιδιοτήτων του βίντεο

Προηγούμενο πλαίσιο = Διάβασε πρώτο πλαίσιο I ()
Προηγούμενο πλαίσιο = Μετατροπή σε ασπρόμαυρο (Προηγούμενο πλαίσιο)
Αρχικοποίηση ιεραρχικης εικόνας 1 ως GLOBAL
Αρχικοποίηση ιεραρχικης εικόνας 2 ως GLOBAL

Όσο υπάρχουν πλαίσια P για διάβασμα:
| Τωρινό πλαίσιο = Διάβασε επόμενο πλαίσιο P ()
| Αν δεν υπάρχει επόμενο πλαίσιο
| | ΕΞΟΔΟΣ από loop
| Αλλιως
| | Αν μέγεθος προηγούμενου πλαισίου δεν διαρείται με το 16:
| | | **προσθήκη_μαυρων_εικονστοιχείων**(προηγούμενο πλαίσιο)
| | Αν μέγεθος τωρινού πλαισίου δεν διαρείται με το 16:
| | | **προσθήκη_μαυρων_εικονστοιχείων**(τωρινό πλαίσιο)
| Τωρινό πλαίσιο = Μετατροπή σε ασπρόμαυρο (Τωρινό πλαίσιο)
| Ιεραρχική εικόνα 1 = [Προηγούμενο πλαίσιο]
| Ιεραρχική εικόνα 2 = [Τωρινό πλαίσιο]
| **Ανάλυση_σε_επίπεδα(**Προηγούμενο πλαίσιο, Τωρινό πλαίσιο)
| Μπλόκ_κίνησης = []
| Για i από 0 μέχρι μέγεθος Προηγούμενου πλαισίου:
| | Αν **εύρεση_κίνησης(**Προηγούμενο πλαίσιο[i] – Τωρινό πλαίσιο[i]):
| | | Μπλόκ_κίνησης.πρόσθεσε(i)
| Εικόνα 1, Εικόνα 2, Μπλόκ_κίνησης = **Βελτιστοποίηση_επιπέδων**(Μπλόκ κίνησης, Ιεραρχική εικόνα 1, Ιεραρχική εικόνα 2)
| Για i από 0 μέχρι μέγεθος Μπλόκ κίνησης:
| | προβλεπόμενο Μπλόκ = **SAD**(Εικόνα 1, Εικόνα 2, Μπλόκ κίνησης[i])
| | Εικόνα 1[Μπλόκ κίνησης[i]] = Εικόνα 1[προβλεπόμενο Μπλόκ]
| Πλαίσια σφαλμάτων = Εικόνα 2 – Εικόνα 1
| **ανακατασκευή_εικόνας(**Πλαίσια σφαλμάτων**)**
| Γράψε στο καινούριο βίντεο την ανακατασκευασμένη εικόνα των πλαισίων σφαλμάτων ()
| Προηγούμενο πλαίσιο = Τωρινό πλαίσιο

Υπολόγισε εντροπία του καινούριου βίντεο ()

3.3 Υλοποίηση

Πρώτο μέρος
Για την υλοποίηση του πρώτου μέρους της άσκησης ξεκινήσαμε διαβάζοντας ένα ένα τα πλαίσια του βίντεο και απλά για κάθε δυο διαδοχικά πλαίσια τα μετατρέπαμε σε δύο πίνακες με τις τιμές για τα pixel των εικόνων αντίστοιχα και αφαιρούσαμε αυτούς τους δύο πίνακες για να υπολογίσουμε τα πλαίσια σφαλμάτων, γράφοντας στο καινούριο βίντεο κάθε φορά το κάθε πλαίσιο σφαλμάτων.

Δεύτερο μέρος
Για την υλοποίηση του δεύτερου μέρους του προβλήματος χρησιμοποιήθηκε η τεχνική ιεραρχικής αναζήτησης και η μετρική SAD(Sum absolute differences). Θα μπορούσε να χρησιμοποιηθεί και η τεχνική λογαριθμικής αναζήτησης αλλά θα έπαιρνε πολύ περισσότερο χρόνο για κάθε πλαίσιο λόγω της ανάλυσης των βίντεο και επιπλέον μας βοηθάει και στην επίλυση της επόμενης άσκησης.

- Ανάγνωση βίντεο και έλεγχος διαίρεσης με το 16

Πιο συγκεκριμένα πάλι ξεκινάμε διαβάζοντας ένα ένα τα πλαίσια του βίντεο ελέγχοντας στην αρχή αν η εικόνα μπορεί να διαιρεθεί σε μακρομπλόκ μεγέθους 16Χ16 και αν όχι γεμίζουμε τις υπολειπόμενες θέσεις του πίνακα τιμών του πλαισίου με μηδενικά (δηλαδή μαύρα pixel) για να μπορεί να διαιρεθεί.

- Τεχνική Ιεραρχικής αναζήτησης και ανάλυσης

Στη συνέχεια μετατρέπουμε τα πλαίσια σε ασπρόμαυρα για ευκολότερες πράξεις και αρχικοποιούμε ως πρώτο επίπεδο της ιεραρχικής μας ανάλυσης τις εικόνες στις αρχικές τους μορφές. Συνεχίζουμε καλώντας τη συνάρτηση που κάνει ανάλυση των εικόνων σε επίπεδα, δηλαδή για κάθε δύο διαδοχικά πλαίσια τα

αναλύουμε σε επίπεδα, όπου σε κάθε επίπεδο μειώνουμε στο μισό την ανάλυση, το μέγεθος των μακρομπλόκ αλλά και την παράμετρο k που καθορίζει την περιοχή αναζήτησης που χρησιμοποιείται στην SAD. Έτσι ενώ ξεκινάμε με την εικόνα στη πλήρες μορφή της, μετά την χωρίζουμε σε μακρομπλόκ 16Χ16 και την αποθηκεύουμε ως δεύτερο επίπεδο όπως ζητείται, μετά σε 8Χ8 κ.ο.κ μέχρι να φτάσουμε στο 4Χ4 που είναι το μικρότερο που θέλουμε και το υψηλότερο επίπεδο στην ιεραρχική αναζήτηση. Με αυτή την τεχνική επιτυγχάνουμε πολύ γρήγορες πράξεις και αναζήτηση του προβλεπόμενου μπλόκ στην SAD καθώς τα μακρομπλόκ στο υψηλότερο επίπεδο έχουν μικρότερο μέγεθος και η περιοχή αναζήτησης είναι μικρότερη.

- <u>Εύρεση διανυσμάτων κίνησης</u>

Έτσι στο υψηλότερο επίπεδο γίνεται μια πλήρης αναζήτηση και βρίσκονται τα διανύσματα κίνησης, όπου τα υπολογίζουμε βρίσκοντας τη διαφορά ανάμεσα σε δυο μακρομπλόκ στις αντίστοιχες θέσεις του κάθε μακρομπλόκ και έπειτα για κάθε από αυτά ελέγχουμε αν το ποσοστό των μηδενικών στο μακρομπλόκ διαφορών είναι μεγαλύτερο του 90% δηλαδή δεν υπήρχε αξιοσήμαντη κίνηση, ενώ αν υπήρχε κίνηση αποθηκεύουμε τη θέση του μακρομπλόκ για να τη ξαναχρησιμοποιήσουμε στο κατώτερο επίπεδο. Έπειτα προχωράμε στο κατώτερο επίπεδο βελτιώνοντας στην ουσία τα μπλόκ και τα διανύσματα κίνησης που υπήρχε κίνηση στο υψηλότερο επίπεδο και τα ξαναελέγχουμε στο χαμηλότερο επίπεδο αυτή τη φορά.

- <u>Εφαρμογή μετρικής SAD</u>

Έχοντας υπολογίσει πλέον όλα τα διανύσματα κίνησης για κάθε μακρομπλόκ, επιλέγουμε εκείνα που εντοπίσαμε κίνηση και εφαρμόζουμε την μετρική SAD για να βρούμε πιο μακρομπλόκ του πλαισίου αναφοράς αντιστοιχεί καλύτερα στο αντίστοιχο μακρομπλόκ του πλαισίου στόχου. Ανατρέχουμε λοιπόν κάθε φορά στην αντίστοιχη θέση και υπολογίζουμε το άθροισμα της

απόλυτης διαφοράς του μακρομπλόκ του πλαισίου αναφοράς, με τα αντίστοιχα γειτονικά μακρομπλόκ του πλαισίου στόχου (που ορίζουμε με τη περιοχή αναζήτησης). Δηλαδή αν έχουμε περιοχή αναζήτησης k = 16 και όλα τα μακρομπλόκ έχουν μήκος και πλάτος 16Χ16, θα υπολογίσουμε το άθροισμα τον απόλυτων διαφορών του μακρομπλόκ του πλασίου αναφοράς με τα γειτονικά μακρομπλόκ του πλαισίου στόχου, που βρίσκονται σε μέγιστη απόσταση ίση ή μικρότερη με 16 προς όλες τις κατευθύνσεις από αυτό. Το ιδανικότερο μακρομπλόκ που θα είναι το μακρομπλόκ πρόβλεψης, είναι αυτό που μας δίνει το μικρότερο αποτέλεσμα καθώς δηλώνει ότι υπάρχουν λιγότερες διαφορές με το πλαίσιο αναφοράς από τα άλλα.

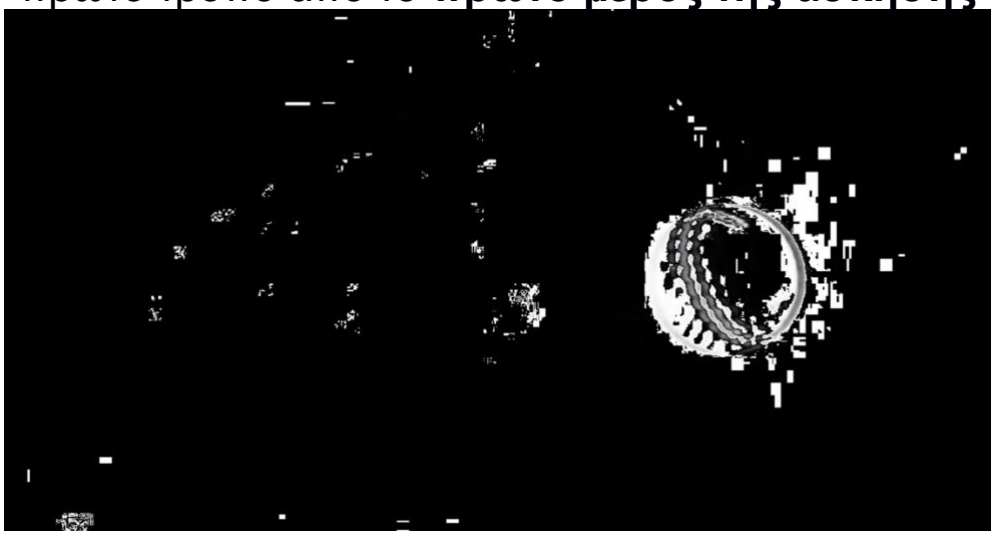
- <u>Ανακατασκευή εικόνας πλαισίων σφαλμάτων με τα προβλεπόμενα μακρομπλόκ</u>

Έχοντας υπολογίσει τα προβλεπόμενα μπλόκ με βάση την μετρική SAD για κάθε μακρομπλόκ αντικαθιστούμε σε κάθε πλαίσιο στη θέση που εντοπίζεται κίνηση το ήδη υπάρχον μπλόκ του πλαισίου αναφοράς με το αντίστοιχο προβλεπόμενο από το πλαίσιο στόχο που βρήκαμε πριν. Μετά υπολογίζουμε τα πλαίσια σφαλμάτων μεταξύ των μακρομπλόκ του προβλεπόμενου πλαισίου αναφοράς και τα μαρκομπλόκ του πλαισίου στόχου αφαιρώντας κάθε μακρομπλόκ του πλαισίου στόχου με το αντίστοιχο μακρομπλόκ του προβλεπόμενου πλαισίου αναφοράς. Τέλος καλούμε τη συνάρτηση ανακατασκευής της εικόνας που παίρνει όλα τα μακρομπλόκ 16Χ16 και τα επαναφέρει στο αρχικό μέγεθος «ενώνοντάς» τα, δίνοντας μας την πλήρη εικόνα στην ανάλυση που θέλουμε, έχοντας πια το ολοκληρωμένο πλαίσιο σφαλμάτων όπου το γράφουμε και στο τελικό αρχείο βίντεο

3.4 Αποτελέσματα

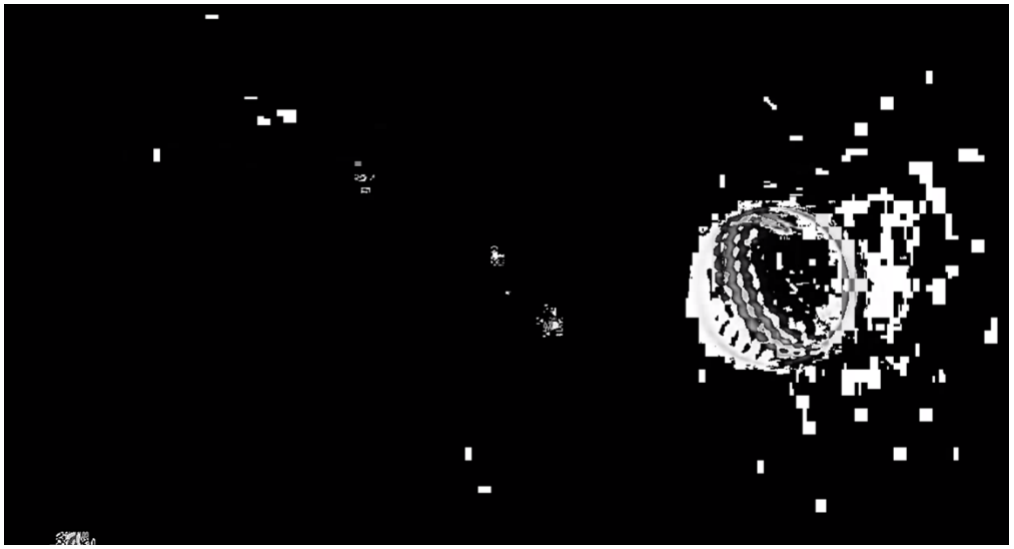
Εδώ βλέπουμε αρχικά ένα πλαίσιο από το αρχικό βίντεο

Το αποτέλεσμα του πλαισίου σφαλμάτων χρησιμοποιώντας τον πρώτο τρόπο από το **πρώτο μέρος της άσκησης**



```
The video has been created.
 entropy = 0.41211439257092364
```

Το αποτέλεσμα του πλαισίου σφαλμάτων χρησιμοποιώντας τον δεύτερο τρόπο από το **δεύτερο μέρος της άσκησης**



```
Frame 73 complete
Frame 74 complete
Frame 75 complete
Frame 76 complete
The video has been created.
 entropy = 0.39752542729378804
```

3.5 Συμπεράσματα

Χρησιμοποιώντας το πρώτο τρόπο από το πρώτο μέρος της άσκησης παρατηρούμε ότι η διαδικασία είναι πολύ γρηγορότερη από τον δεύτερο τρόπο στο δεύτερο μέρος της άσκησης. Όμως από τα αποτελέσματα διακρίνουμε ότι ο δεύτερος τρόπος με τη χρήση της μετρικής SAD, αν και θέλει περισσότερη ώρα για την ολοκλήρωσή του δίνει χαμηλότερη εντροπία για τα δεδομένα μας με τα αποτελέσματα να είναι πολύ παρόμοια με το πρώτο τρόπο και ίσως με την ίδια η καλύτερη ακρίβεια.

Άσκηση 8.18

4.1 Εκφώνηση άσκησης

«Σε αυτήν την άσκηση θα δείτε ότι η τεχνική της τμηματικής πρόβλεψης με βάση την αντιστάθμιση κίνησης, μπορεί επίσης να χρησιμοποιηθεί σε εφαρμογές εκτός συμπίεσης. Μία τέτοια ενδιαφέρουσα εφαρμογή είναι η απομάκρυνση αντικειμένων ή προσώπων από τη ροή του βίντεο. Για παράδειγμα, έστω βίντεο στο οποίο η κάμερα δεν έχει κινηθεί και το παρασκήνιο είναι σχετικά στατικό, αλλά κινούνται ορισμένα αντικείμενα στο προσκήνιο. Στόχος σας είναι να προσεγγίσετε το αντικείμενο χρησιμοποιώντας μπλοκ και στη συνέχεια να αντικαταστήσετε αυτά τα μπλοκ με παρασκήνιο, σαν να μην ήταν ποτέ παρόν το αντικείμενο. Στη γενική περίπτωση, η λύση είναι πολύ δύσκολη, αλλά στο πλαίσιο αυτής της άσκησης θα επεξεργαστείτε ορισμένες απλούστερες ιδέες. Κατά την υλοποίηση, βεβαιωθείτε ότι μπορείτε να χειρίζεστε το μέγεθος του μπλοκ ως παράμετρο, προκειμένου να ελέγξετε πόσο καλά λειτουργεί ο αλγόριθμος απομάκρυνσης αντικείμενών για διάφορα μεγέθη μακρομπλόκ.»

4.2 Περιγραφή συστήματος/ψευδοκώδικας

Για την υλοποίηση και την επίλυση της άσκησης χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python 3.
Για να μπορεί να γίνει η εκτέλεση του προγράμματος θα πρέπει να υπάρχουν εγκατεστημένες στο σύστημα οι εξής βιβλιοθήκες: scipy, numpy & cv2.

Ψευδοκώδικας

Μέθοδοι / συναρτήσεις προγράμματος

Συνάρτηση χωρισμού σε μακρομπλόκς(πλήθος_γραμμών, πλήθος_στηλών, πίνακας_εικόνας):
    Μπλοκς = []
    Για k από 0 μέχρι το (πλάτος_εικόνας - πλήθος_γραμμών+1) με βήμα = πλήθος_γραμμών :
    | Για j μέχρι (ύψος_εικόνας - πλήθος_στηλών+1) με βήμα = πλήθος_στηλών:
    | μακρομπλόκ = πίνακας [k:k+πλήθος_γραμμών, c:c+πλήθος_στηλών]
    | Μπλοκς.πρόσθεσε(μακρομπλόκ)
    μετατροπή του Μπλοκς σε πίνακα numpy array
    Επιστροφή πίνακα Μπλοκς

Συνάρτηση προσθήκη_μαύρων_εικονοστοιχείων(εικόνα):
    Ύψος_εικόνας
    Πλάτος_εικόνας
    προσθήκη_μαύρων_εικονοστοιχείων = μετατροπή_σε_πίνακα(μαύρο_pixel() * (((πλάτος-1 // 16+1)* 16)- πλάτος))
    Για i μέχρι μέγεθος_εικόνας
    |γραμμή = μετατροπή_σε_πίνακα(i, προσθήκη_μαύρων_εικονοστοιχείων, άξονας_γραμμών)
    | προσωρινή_εικόνα = γραμμή
        προσθήκη_μαύρης_γραμμής = μετατροπή_σε_πίνακα(μαύρο_pixel() * (((πλάτος-1 // 16+1) * 16)- πλάτος))
    Για j μέχρι (((πλάτος-1 // 16+1) * 16)- πλάτος)):
        προσωρινή_εικόνα = προσθηκη_μαύρης_γραμμής
    Επιστροφή μετατροπή_σε_πίνακα(προσωινή_εικόνα)

Συνάρτηση μαύρο_εικονστοιχείο():

Επιστροφή [0]

Συνάρτηση ιεραρχική_αναζήτηση(πίνακα):
πίνακα1 = μετατροπή_σε_πίνακα()
x, y, = μορφή_πίνακα /* να έχουμε τις συντεταγμένες σε μορφή πίνακα*/
πίνακα2 = [] /*αρχικοποιήση του πίνακα2 σαν λίστα*/
Για i από 0 μέχρι x με_βήμα 2:
| Για j από 0 μέχρι y με_βήμα 2:
| | *πίνακα2* = πίνακα1([x], [y]) /*ο πίνακα2 είναι σε μορφή λίστας */
πίνακα2 = μετατροπή_σε_πίνακα()
πίνακα2 = σχηματισμός(x/2, y/2) /* για να αποκτήσει τις διαστάσεις που θέλουμε */
Επιστροφή πίνακα2

Συνάρτηση ανάλυση_σε_επίπεδα (εικόνα ανφοράς, εικόνα στόχος):
εικόνα ανφοράς = μετατροπή_σε_πίνακα()
εικόνα στόχος = μετατροπή_σε_πίνακα()
Για i μέχρι 2: //τα επίπεδα
| ιεραρχική_εικόνα1 = ιεραρχική_εικόνα1 + [ιεραρχική_αναζήτηση(εικόνα αναφοράς)]
| εικόνα αναφοράς = ιεραρχική_αναζήτηση(εικόνα αναφοράς)
| ιεραρχική_εικόνα2 = ιεραρχική_εικόνα2 + [ιεραρχική_αναζήτηση(εικόνα στόχος)]
| εικόνα στόχος = ιεραρχική_αναζήτηση(εικόνα στόχος)

Συνάρτηση βελτιστοποίηση_Επιπέδων(blocks_move, ιεραρχική_εικόνα1, ιεραρχική_εικόνα2):
size_block = [8, 16]
Για k μέχρι 2:
| /* διαίρεση της μικρής εικόνας σε blocks μεγέθους size_block * size_block */
| εικόνα1 = blocks(size_block [k], size_block [k], ιεραρχική_εικονα1[size_block [1- k])
| εικόνα2 = blocks(size_block [k], size_block [k], ιεραρχική_εικονα2[size_block [1- k])
| blocks_χωρίς_κίνηση=[]
| Για i μέχρι το μέγεθος(blocks_move):/*ελέγχουμε μόνο τα blocks στα οποία βρήκαμε κίνηση*/
| | Αν εύρεση_κίνησης (εικόνα1[blocks_move[i]] – εικονα2[blocks_move[i]]):
| | | Συνέχισε
| | Αλλιώς:
| | | blocks_χωρίς_κίνηση = blocks_χωρίς_κίνηση [i] /* αν δεν υπάρχει κίνηση
| | |
βάζουμε το block στην λίστα*/
| blocks_move =[αν κάποιο block_χωρίς_κίνηση δεν είναι στη λίστα blocks_moves,αφαίρεσε το]

Επιστροφή εικόνα1, εικόνα2, blocks_move

Συνάρτηση εύρεση_κίνησης(πίνακα):
πίνακα1 = μετατροπή_σε_πίνακα()
x, y = μορφή_πίνακα
πλήθος_μηδενικών_στοιχείων = x * y – μη_μηδενικά_στοιχεία(πίνακα1)
Αν πλήθος_μηδενικών_στοιχείων >= 0.8 *x *y:
Επέστρεψε 0
Αλλιώς:
Επέστρεψε 1

Συνάρτηση ανακατασκευή_εικόνας(x, y, εικόνα2):
ρ =1
Για i μέχρι x:
| έξοδος = μετατροπή_σε_πίνακα(εικόνα2 [i *y]) /* για να πάρουμε τις διαστάσεις
| της κάθε εικόνας*/
| Για j μέχρι (y-1):
| | έξοδος = ένωσε(έξοδο, εικόνα2[ρ], άξονα_στηλών)
| | ρ +=1
| ρ +=1
| Αν i == 0:
| ανακατασκευασμένη_εικόνα = έξοδο
| Αλλιώς:
| ανακατασκευασμένη _εικόνας = ένωσε((εμφάνιση_εικονας, έξοδο), άξονας_γραμμών)
Επιστροφή ανακατασκευασμένη _εικόνας

Κυρίως πρόγραμμα

Ρυθμίσεις και ιδιότητες βίντεο
|
| αντικείμενο για ανάγνωση βίντεο ()
| ύψος πλαισίου ()
| μήκος πλαισίου ()
| κωδικοποίση βίντεο ()
| αρχείο για εγγραφή του βίντεο ()
|
Τέλος ρυθμίσεων και ιδιοτήτων του βίντεο

```
Background = Διάβασε πρώτο πλαίσιο I ()
Background = Μετατροπή σε ασπρόμαυρο (Προηγούμενο πλαίσιο)
Αρχικοποίηση ιεραρχικης εικόνας 1 ως GLOBAL
Αρχικοποίηση ιεραρχικης εικόνας 2 ως GLOBAL

Όσο υπάρχουν πλαίσια P για διάβασμα:
|       Τωρινό πλαίσιο = Διάβασε επόμενο πλαίσιο P ()
|       Αν δεν υπάρχει επόμενο πλαίσιο
|       |       ΕΞΟΔΟΣ από loop
|       Αλλιως
|       |     Αν μέγεθος προηγούμενου πλαισίου δεν διαρείται με το 16:
|       |       |       προσθήκη_μαυρων_εικονστοιχείων(προηγούμενο πλαίσιο)
|       |     Αν μέγεθος τωρινού πλαισίου δεν διαρείται με το 16:
|       |       |       προσθήκη_μαυρων_εικονστοιχείων(τωρινό πλαίσιο)
        Τωρινό πλαίσιο = Μετατροπή σε ασπρόμαυρο (Τωρινό πλαίσιο)
|       Ιεραρχική εικόνα 1 = [Background]
|       Ιεραρχική εικόνα 2 = [Τωρινό πλαίσιο]
|       Ανάλυση_σε_επίπεδα(Background, Τωρινό πλαίσιο)
|       Μπλόκ_κίνησης = []
|       Για i από 0 μέχρι μέγεθος Backgorund:
|       |       Αν εύρεση_κίνησης(Backgorund[i] – Τωρινό πλαίσιο[i]):
|       |       |       Μπλόκ_κίνησης.πρόσθεσε(i)
|       Εικόνα 1, Εικόνα 2, Μπλόκ_κίνησης = Βελτιστοποίηση_επιπέδων(Μπλόκ
        κίνησης, Ιεραρχική εικόνα 1, Ιεραρχική εικόνα 2)
|       Για i από 0 μέχρι μέγεθος Μπλόκ κίνησης:
|       |       Εικόνα 2[Μπλόκ κίνησης[i]] = Εικόνα 1[Μπλόκ κίνησης[i]]
|       ανακατασκευή_εικόνας(Πλαίσια σφαλμάτων)
|       Γράψε στο καινούριο βίντεο την ανακατασκευασμένη εικόνα ()

Υπολόγισε εντροπία του καινούριου βίντεο ()
```

4.3 Υλοποίηση

Για την υλοποίηση του προβλήματος χρησιμοποιήθηκε η τεχνική ιεραρχικής αναζήτησης. Θα μπορούσε να χρησιμοποιηθεί και η τεχνική λογαριθμικής αναζήτησης αλλά θα έπαιρνε πολύ περισσότερο χρόνο για κάθε πλαίσιο λόγω της ανάλυσης των βίντεο.

- Ανάγνωση βίντεο και έλεγχος διαίρεσης με το 16

Πιο συγκεκριμένα πάλι ξεκινάμε διαβάζοντας ένα-ένα τα πλαίσια του βίντεο ελέγχοντας στην αρχή αν η εικόνα μπορεί να διαιρεθεί σε μακρομπλόκ μεγέθους 16Χ16 και αν όχι γεμίζουμε τις υπολειπόμενες θέσεις του πίνακα τιμών του πλαισίου με μηδενικά (δηλαδή μαύρα pixel) για να μπορεί να διαιρεθεί.

- Τεχνική Ιεραρχικής αναζήτησης και ανάλυσης

Στη συνέχεια μετατρέπουμε τα πλαίσια σε ασπρόμαυρα για ευκολότερες πράξεις και αρχικοποιούμε ως πρώτο επίπεδο της ιεραρχικής μας ανάλυσης τις εικόνες στις αρχικές τους μορφές. Το πρώτο πλαίσιο του βίντεο όπου θα είναι και το παρασκήνιο μας, αποθηκεύεται ως η πρώτη εικόνα. Συνεχίζουμε καλώντας τη συνάρτηση που κάνει ανάλυση των εικόνων σε επίπεδα, δηλαδή για κάθε δύο διαδοχικά πλαίσια τα αναλύουμε σε επίπεδα οπού σε κάθε επίπεδο μειώνουμε στο μισό την ανάλυση, το μέγεθος των μακρομπλόκ αλλά και την παράμετρο k που καθορίζει την περιοχή αναζήτησης. Έτσι ενώ ξεκινάμε με την εικόνα στη πλήρες μορφή της μετά την χωρίζουμε σε μακρομπλόκ 16Χ16 και την αποθηκεύουμε ως δεύτερο επίπεδο όπως ζητείται και μετά σε 8Χ8 κ.ο.κ μέχρι να φτάσουμε στο 4Χ4 που είναι το μικρότερο επίπεδο, και το υψηλότερο αντίστοιχα στην ιεραρχική αναζήτηση. Με αυτή την τεχνική επιτυγχάνουμε πολύ γρήγορες πράξεις για την εύρεση των διανυσμάτων κίνησης και την αναζήτηση του καλύτερου ταιριαστού μακρομπλόκ, καθώς τα μακρομπλόκ στο υψηλότερο επίπεδο

έχουν μικρότερο μέγεθος και η περιοχή αναζήτησης είναι μικρότερη.

- Εύρεση διανυσμάτων κίνησης

Έτσι στο υψηλότερο επίπεδο γίνεται μια πλήρης αναζήτηση και βρίσκονται τα διανύσματα κίνησης, όπου τα υπολογίζουμε βρίσκοντας τη διαφορά ανάμεσα σε δυο μακρομπλόκ στις αντίστοιχες θέσεις του κάθε μακρομπλόκ και έπειτα για κάθε ένα από αυτά ελέγχουμε αν το ποσοστό των μηδενικών στο μακρομπλόκ διαφορών, είναι μεγαλύτερο του 90% δηλαδή δεν υπήρχε αξιοσήμαντη κίνηση. Αντιθέτως αν υπήρχε κίνηση αποθηκεύουμε τη θέση του μακρομπλόκ για να τη ξαναχρησιμοποιήσουμε στο κατώτερο επίπεδο. Έπειτα προχωράμε στο κατώτερο επίπεδο βελτιώνοντας στην ουσία τα μπλόκ και τα διανύσματα κίνησης που υπήρχε κίνηση στο υψηλότερο επίπεδο και τα ξαναελέγχουμε στο χαμηλότερο επίπεδο αυτή τη φορά.

- Ανακατασκευή εικόνας με τα αντικατεστημένα μακρομπλόκ

Έχοντας υπολογίσει τα διανύσματα κίνησης για κάθε μακρομπλόκ αντικαθιστούμε σε κάθε μακρομπλόκ στη θέση που εντοπίζεται κίνηση το ήδη υπάρχον μπλόκ του πλαισίου αναφοράς με το αντίστοιχο μακρομπλόκ του παρασκηνίου στην αντίστοιχη θέση.
Τέλος καλούμε τη συνάρτηση ανακατασκευής της εικόνας που παίρνει όλα τα μακρομπλόκ 16Χ16 και τα επαναφέρει στο αρχικό μέγεθος «ενώνοντάς» τα, δίνοντας μας την πλήρης εικόνα στην ανάλυση που θέλουμε έχοντας πιά τα ολοκληρωμένα πλαίσια με την «εξαφάνιση» του αντικειμένου, όπου και το γράφουμε στο τελικό αρχείο βίντεο.

4.4 Αποτελέσματα

Το αρχικό μας βίντεο:

Το βίντεο μετά την απομάκρυνση αντικειμένων:

4.5 Συμπεράσματα

Για την απομάκρυνση του αντικείμενου σπάσαμε σε μακρομπλόκς τα μπλόκς για να μπορέσουμε να εργαστούμε σε αυτά. Χρησιμοποιήσαμε ιεραρχική αναζήτηση, αλλιώς ο χρόνος που χρειαζόταν για να γίνει η αναζήτηση των κινούμενων αντικειμένων αλλά και απομάκρυνσή τους θα ήταν πολύ μεγάλος.Επομένως συμπεραίνουμε πως είναι πολύ πρακτικό να δουλεύουμε/αναζητούμε σε μικρότερα μπλόκς, αντι σε όλο το μπλόκ. Αν δεν είχαμε υλοποιήσει την ιεραρχική αναζήτηση ο χρόνος ο οποιος θα χρειάζοταν για να γίνει η αναζήτηση αλλά και η απομάκρυνση θα ήταν μεγάλος. *Τέλος βλέπουμε πως δεν εύκολο να απομακρύνουμε το αντικείμενο 100%, παρά μένουν κάποια pixels τα οποία δεν καταφέραμε να τα αφαιρέσουμε.*

5 Βιβλιογραφία και βιβλιοθήκες που χρησιμοποιήθηκαν

5.1 Βιβλιοθήκες Python

Στη διάρκεια της υλοποίησης της άσκησης χρησιμοποιήθηκαν γνώσεις και μέθοδοι από διάφορες έτοιμες βιβλιοθήκες της python που είναι επιστημονικού περιεχομένου και είναι φτιαγμένες για τέτοιου είδους προβλήματα.

Πιο συγκεκριμένα:

Για το ερώτημα 6.16 :
1)numPy
2)PIL

Για το ερώτημα 6.17 :
1)numPy
2)cv2
3)os

Για το ερώτημα 8.17 και 8.18 :
1)numPy
2)cv2
3) SciPy (συγκεκριμένα απο το scipy.stats το entropy)

Τα numPy, SciPy είναι βιβλιοθήκες με πιο πολύ μαθηματικές πράξεις και υλοποίηση αλγορίθμων

Το cv2 (https://pypi.org/project/opencv-python/)είναι για το αντίστοιχο tool της C++ που είναι γνωστό ως OpenCV και είναι γραμμένο σε C++, αλλά τρέχει σε python με τη μορφή framework του cv2.

Το PIL (Python Image Library), είναι βιβλιοθήκη που περιέχει συναρτήσεις για την επεξεργασία εικόνων αλλά και την εμφάνιση.

5.2 Βιβλιογραφία

Είναι πολύ σημαντικό να σημειωθεί όμως ότι η έμπνευση και ο σχεδιασμός των λύσεων όλων των ασκήσεων έγινε βάση της θεωρίας και των παραδειγμάτων του βιβλίου.
Πιο συγκεκριμένα:
Για την άσκηση 6.16 : Κεφ 6 σελ. 187 ενότητα 4.1
Για την άσκηση 6.17 : Κεφ 6 σελ. 194-196 ενότητα 5.1
Για την άσκηση 8.17 και 8.18 : Κεφ 8 σελ. 256-263 ενότητα 1.1-1.3
Για την άσκηση 8.17 Κεφ 8 σελ. 264-267 ενότητα 1.5,2.1,2.2
Για την υλοποίηση της ιεραρχικής αναζήτησης : Κεφ 8 σελ. 274-276 ενότητα 3.3
Για τα πρότυπα κωδικοποίησης των βίντεο κατά την εγγραφή : Κεφ 8 σελ.281 ενότητα 4.6

5.3 ΒΙΒΛΙΟΓΡΑΦΙΑ ΔΙΑΔΙΚΤΥΟΥ

Από το διαδίκτυο χρησιμοποιήσαμε τα εξής links και documentations για την επεξήγηση των αλγορίθμων αλλά και των χαρακτηριστικών τους:

https://pillow.readthedocs.io/en/stable/ τελευταία προσπέλαση 17/6/2019

https://docs.scipy.org/doc/ τελευταία προσπέλαση 17/6/2019

https://auth0.com/blog/image-processing-in-python-with-pillow/ τελευταία προσπέλαση 17/6/2019

https://docs.scipy.org/doc/ τελευταία προσπέλαση 17/6/2019

https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/ τελευταία προσπέλαση 17/6/2019