ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΉΣ

ΜΑΘΗΜΑ: «ΣΥΣΤΗΜΑΤΑ ΠΟΛΥΜΕΣΩΝ»
ΔΙΔΑΣΚΩΝ: «ΠΙΚΡΑΚΗΣ ΑΓΓΕΛΟΣ»

ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΚΑΡΑΓΙΑΝΝΗ ΔΕΣΠΟΙΝΑ	Π20073
ΠΟΛΥΧΡΟΝΗ ΕΛΕΝΗ-ΧΡΙΣΤΙΝΑ	Π20161
ΑΝΑΣΤΑΣΙΟΥ ΣΤΥΛΛΙΑΝΗ - ΜΑΡΙΑ	Π20016

Πίνακας περιεχομένων

ΘΕΜΑ 1

ΕΚΦΩΝΗΣΗ ΘΕΜΑΤΟΣ

Θέμα 1 (1.5 βαθμοί): Έστω ασυμπίεστο video της επιλογής σας, διάρκειας 5 s – 15 s. Υποθέστε ότι το Frame 1 είναι πάντα I frame και ότι τα επόμενα πλαίσια είναι P frames.

i) Κάθε πλαίσιο P προβλέπεται χωρίς αντιστάθμιση κίνησης από το προηγούμενο πλαίσιο. Υπολογίστε και απεικονίστε την ακολουθία εικόνων σφάλματος και κωδικοποιήστε την χωρίς απώλειες. Υλοποιήστε τον κωδικοποιητή/αποκωδικοποιητή.

ii) Υλοποιήστε την τεχνική αντιστάθμισης κίνησης για την συμπίεση της ακολουθίας πλαισίων χρησιμοποιώντας αντιστάθμιση κίνησης σε macroblocks 64x64, ακτίνα αναζήτησης k=32 και τεχνική σύγκρισης macroblocks της επιλογής σας. Να επιταχυνθεί η διαδικασία υλοποιώντας ιεραρχική αναζήτηση. Υπολογίστε, αποθηκεύστε και απεικονίστε την ακολουθία εικόνων πρόβλεψης και εικόνων σφαλμάτων. Υλοποιήστε τον κωδικοποιητή/αποκωδικοποιητή.

ΛΥΣΗ ΠΡΩΤΟΥ ΥΠΟΕΡΩΤΗΜΑΤΟΣ

Αρχική μας δουλειά, είναι να **χωρίσουμε** το βίντεο της επιλογής μας σε **πλαίσια** (frames), καλώντας την αντίστοιχη συνάρτηση. Στην συνέχεια και έχοντας πλέον **συλλέξει** και **αποθηκεύσει** σε κατάλληλο φάκελο τα frames του βίντεο, μπορούμε να προβούμε στην **πρόβλεψη** των **P frames**. Με βάση τα δεδομένα της εκφώνησης, πως κάθε πλαίσιο P προβλέπεται **χωρίς** αντιστάθμιση κίνησης από το προηγούμενό του, και ανάμεσα στους διάφορους τρόπους που υπάρχουν για να υλοποιηθεί η πρόβλεψη, επιλέγουμε να προβλέψουμε κάθε πλαίσιο P, χρησιμοποιώντας την **πρόβλεψη της διαφοράς** και πιο συγκεκριμένα με την βοήθεια τους εξής τύπου:

P(x,y) = A(x,y) + (B(x,y) – A(x,y))

όπου:

- P: ο πίνακας του πλαισίου P που θα προβλεφθεί
- A: ο πίνακας του πλαισίου που προηγείται του πλαισίου B (reference frame)
- B: ο πίνακας του πλαισίου που έπεται του πλαισίου A (target frame)

*** Οι πίνακες A και B που αναφέρθηκαν παραπάνω είναι δυσδιάστατοι και φέρουν μέσα τους τιμές, που είναι στην πραγματικότητα τα εικονοστοιχεία (pixel) του αντίστοιχου πλαισίου του βίντεο .**

*** Ο πίνακας P θα είναι και αυτός δυσδιάστατος και θα φέρει την διαφορά που υπολογίζεται βάση τον παραπάνω τύπο.**

Η **μετατροπή** των **frames** σε **δυσδιάστατους πίνακες** που φέρουν μέσα τους τις τιμές των **pixel** του αντίστοιχου frame, γίνεται με την κλήση της αντίστοιχης συνάρτησης.

Έχοντας πλέον υπολογίσει το προβλεφθέν πλαίσιο P, μπορούμε να υπολογίσουμε την **εικόνα σφάλματος**. Η εικόνα σφάλματος δεν είναι παρά ένας δυσδιάστατος πίνακας με την **διαφορά** του **προβλεφθέν πλαισίου – του προηγούμενού του**.

Στην συνέχεια πρέπει να **κωδικοποιήσουμε** την **εικόνα σφάλματος** που υπολογίσαμε στο προηγούμενο βήμα. Χρησιμοποιούμε την **κωδικοποίηση Huffman** για να το επιτύχουμε.

Κωδικοποίηση Huffman (Huffman encoding)

class Node:

Για να υλοποιήσουμε την κωδικοποίηση Huffman, ξεκινάμε με μια **κλάση Node**, που αναφέρεται στους **κόμβους** του **δυαδικού Huffman δέντρου**. **Κάθε** κόμβος της κλάσης αυτής έχει:

1. ένα **σύμβολο (symbol)**
2. την **πιθανότητα** του συμβόλου αυτού **(prob)**
3. ένα **αριστερό** παιδί **(left)**
4. ένα **δεξί** παιδί **(right)**
5. έναν **κωδικό**. Ο κωδικός παίρνει τιμές **0 ή 1**, ανάλογα με το πως **διασχίζουμε** το δέντρο. (**αριστερά 0**, **δεξιά 1**).

Η **βασική** μας συνάρτηση για την κωδικοποίηση είναι η **Huffman_encoding** και έχουμε **3 βοηθητικές** συναρτήσεις:

1. **calculate_probability**, για να υπολογίσουμε τις **πιθανότητες** εμφάνισης των **συμβόλων** που έρχονται ως είσοδο, που είναι στην πραγματικότητα οι **τιμές** (pixel) του πίνακα της **εικόνας σφάλματος**
2. **calculate_codes**, για την **ανάθεση κωδικών** στα σύμβολα και
3. **output_encoded,** που **παράγει** και **επιστρέφει** τα **κωδικοποιημένα** δεδομένα.

Συνάρτηση Huffman_encoding:

Η συνάρτηση αυτή, παίρνει ως παράμετρο τον πίνακα της **εικόνας σφάλματος** (data) και επιστρέφει τα **κωδικοποιημένα δεδομένα** (encoded_output) και το **δέντρο Huffman** κάνοντας χρήση των **3 συναρτήσεων** που αναφέρθηκαν προηγουμένως.

Στην συνάρτηση αυτά τα δεδομένα αποστέλλονται ως **μονοδιάστατος πίνακας** (μορφή λίστας) και όχι ως **δισδιάστατος** που είναι η κανονική μορφή. Αυτό γίνεται για **ευκολία πράξεων**.

Λειτουργία συνάρτησης:

→ **Υπολογισμός Πιθανοτήτων και Διάταξη Συμβόλων**

- Αρχικά, με την βοήθεια της αντίστοιχης **συνάρτησης** υπολογίζουμε τις **φορές** που εμφανίζεται το **κάθε σύμβολο** στον πίνακα της **εικόνας σφάλματος**. Αποθηκεύουμε σε **λεξικό** το εκάστοτε σύμβολο μαζί με τις φορές εμφάνισης του.

- Θεωρούμε το κάθε σύμβολο του λεξικού μαζί με την πιθανότητά του, ως έναν **κόμβο** ενός **δυαδικού δέντρου τύπου Huffman**.
- **Ταξινομούμε** σε **αύξουσα σειρά**, τους κόμβους του δέντρου βάση την **πιθανότητα τους**.

→ **Κατασκευή Δυαδικού Δέντρου**

- Διαλέγουμε κάθε φορά, εκείνους τους **2 κόμβους** με την **μικρότερη** πιθανότητα και τους συνενώνουμε σε έναν **νέα κόμβο** με ρίζα το **άθροισμα** των πιθανοτήτων των κόμβων που μόλις συγχωνεύσαμε.
- **Ανακατασκευάζουμε** το νέο δέντρο.
- Η διαδικασία αυτή **επαναλαμβάνεται**, μέχρι να κατασκευαστεί ένα δέντρο που να περιλαμβάνει **όλες** τις πιθανότητες εισόδου

→ **Ανάθεση Κωδικών στα Σύμβολα του Δέντρου**

- Κάθε φορά που πηγαίνουμε σε **αριστερό παιδί** του δέντρου αναθέτουμε την τιμή **0** και κάθε φορά που πηγαίνουμε σε **δεξί παιδί** την τιμή **1**. Αυτό επιτυγχάνεται με την κλήση της αντίστοιχης συνάρτησης.

Αποκωδικοποίηση Huffman (Huffman decoding)

Έχοντας κατασκευάσει το **Δυαδικό δέντρο Huffman** κατά την διάρκεια της **κωδικοποίησης**, η φάση της **αποκωδικοποίησης** είναι σχετικά εύκολη να υλοποιηθεί.

Το μόνο πράγμα που έχουμε να κάνουμε είναι να ξεκινήσουμε από την **αρχή** του **δέντρου** (ρίζα δέντρου) και από την **αρχή** των **κωδικοποιημένων** δεδομένων και κάθε φορά που συναντάμε **1** πηγαίνουμε **δεξιά** ενώ όταν συναντάμε **0** πηγαίνουμε **αριστερά** στο **δέντρο Huffman**. Όταν φτάσουμε σε κάποιο **φύλλο** του δέντρου, παίρνουμε το επιθυμητό **σύμβολο**. Ξεκινάμε **ξανά** από την αρχή του δέντρου προχωρώντας με τα κωδικοποιημένα δεδομένα.

Συνάρτηση Huffman_decoding:

Η συνάρτηση **Huffman_decoding** υλοποιεί την διαδικασία της αποκωδικοποίησης που αναφέρθηκε παραπάνω. Παίρνει ως είσοδο τα **κωδικοποιημένα δεδομένα** (encoded_data) και **το δέντρο Huffman** (huffman_tree).

Τα **αποκωδικοποιημένα** δεδομένα πριν επιστραφούν στο **κυρίως πρόγραμμα** βρίσκονται σε **μονοδιάστατο** πίνακα. Για να τα επιστρέψουμε σωστά, **μετατρέπουμε** τον πίνακα σε **δισδιάστατο** δηλαδή στην αρχική του μορφή.

* Για να βρούμε τις **διαστάσεις** του πίνακα της **εικόνας σφάλματος**, αρκεί να βρούμε τις γραμμές και τις στήλες ενός πίνακα **πλαισίου**, αφού έτσι και αλλιώς θα έχουν την ίδια διάσταση.

Έλεγχος Λειτουργίας Αποκωδικοποίησης

Η **όλη διαδικασία**, από την πρόβλεψη του πλαισίου P μέχρι την αποκωδικοποίηση του αρχικού πίνακα σφάλματος, επαναλαμβάνεται για **όλα** τα **frames** του βίντεο (σε **ζευγαράκια [**προβλεφθέν πλαίσιο, προηγούμενο πλαίσιο]).

Πρέπει να βεβαιωθούμε πως η **αποκωδικοποίηση** λειτουργεί **ορθά**, δηλαδή παράγει **σωστά** τον **αρχικό πίνακα σφάλματος** για το **κάθε πλαίσιο** του βίντεο που έχει προβλεφθεί. Για να το πετύχουμε αυτό χρησιμοποιούμε έναν **μετρητή** και κάθε φορά που **συγκρίνουμε** τον **αρχικό πίνακα σφάλματος** με τον **πίνακα** που μας επιστρέφει η **συνάρτηση αποκωδικοποίησης, αυξάνουμε** την τιμή του μετρητή **κατά 1**, αν οι 2 πίνακες **ταυτίζονται,** δηλαδή περιέχουν ακριβώς την **ίδια τιμή** στην αντίστοιχη γραμμή και στήλη.

Αν για **όλα** τα ζευγαράκια πλαισίων, ισχύει ότι ο αρχικός πίνακας σφάλματος είναι **ίδιος** με τον τελικό, εμφανίζουμε στην οθόνη κατάλληλο μήνυμα **επιτυχίας**, αλλιώς **αποτυχίας**.

Στιγμιότυπα εκτέλεσης προγράμματος

```
Reference Frame:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Target Frame:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Predict PFrame is:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Error table:
 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

1η επανάληψη

```
Reference Frame:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Target Frame:
 [[ 1  2  6 ... 39 47 50]
 [38 46 49 ... 28 33 37]
 [28 33 37 ... 38 45 63]
 ...
 [ 8 13  9 ...  1 10  5]
 [ 5 10  6 ... 11 17  7]
 [ 9 15  5 ...  6  8  5]]

Predict PFrame is:
 [[ 1  2  6 ... 39 47 50]
 [38 46 49 ... 28 33 37]
 [28 33 37 ... 38 45 63]
 ...
 [ 8 13  9 ...  1 10  5]
 [ 5 10  6 ... 11 17  7]
 [ 9 15  5 ...  6  8  5]]

Error table:
 [[ 0  0  0 ...  1 -1  0]
 [-1  2 -1 ...  5  0  3]
 [ 3 -2  0 ...  0 -1 -2]
 ...
 [ 2 -3  1 ... -3 -5 -4]
 [ 1 -5 -3 ...  0  1 -2]
 [ 1 -1  0 ... -2 -2 -2]]
```

2η επανάληψη

```
Reference Frame:
 [[ 0  0  0 ... 57 64 74]
 [58 67 76 ... 31 36 40]
 [30 35 39 ... 41 46 52]
 ...
 [10 12 11 ...  3 17  4]
 [ 3 14  6 ...  8 18 10]
 [11 16 10 ...  6 10  9]]

Target Frame:
 [[ 0  0  0 ... 57 64 74]
 [58 67 76 ... 31 36 40]
 [30 35 39 ... 41 46 52]
 ...
 [10 12 11 ...  3 17  4]
 [ 3 14  6 ...  8 18 10]
 [11 16 10 ...  6 10  9]]

Predict PFrame is:
 [[ 0  0  0 ... 57 64 74]
 [58 67 76 ... 31 36 40]
 [30 35 39 ... 41 46 52]
 ...
 [10 12 11 ...  3 17  4]
 [ 3 14  6 ...  8 18 10]
 [11 16 10 ...  6 10  9]]

Error table:
 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Success!
```

τελευταία επανάληψη

ΛΥΣΗ ΔΕΥΤΕΡΟΥ ΥΠΟΕΡΩΤΗΜΑΤΟΣ

Όπως περιγράφτηκε και παραπάνω, στο 1ο υποερώτημα, ξεκινάμε με το να **χωρίσουμε** το βίντεο σε **πλαίσια** (frames), με την βοήθεια της αντίστοιχης συνάρτησης. Στην συνέχεια ορίζουμε με βάση τα δεδομένα της εκφώνησης, **block_size = 64** και ακτίνα αναζήτησης **(k = 32)** που θα τα χρειαστούμε παρακάτω στο πρόγραμμα.

Ιεραρχική Αναζήτηση (Hierarchical search)

Συνάρτηση hierarchical_search

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. το **πλαίσιο αναφοράς** (reference frame),
2. το **πλαίσιο στόχο** (target frame) που είναι και το πλαίσιο που θα προβλεφθεί,
3. το **μέγεθος του macroblock** (block_size) και
4. την **ακτίνα αναζήτησης** (k).

Η συνάρτηση αυτή **επιστρέφει**:

1. μια λίστα με τα **διανύσματα κίνησης** του εκάστοτε **πλαισίου αναφοράς** και
2. μια λίστα με τα **υποψήφια μακρομπλοκ ταιριάσματος**.

Λειτουργία συνάρτησης:

Ξεκινάμε ορίζοντας με την βοήθεια ενός μετρητή το **αρχικό επίπεδο** που βρισκόμαστε (**l=1**). Στην συνέχεια ορίζουμε να έχουμε **4 επίπεδα** στην ιεραρχία. **Μέσα στο βρόχο επανάληψης** για την **ανάβαση** επιπέδων, πραγματοποιούμε **υποδειγματοληψία κατά 2** και **φιλτράρουμε και στις 2 διαστάσεις** τόσο στο πλαίσιο αναφοράς όσο και στο πλαίσιο στόχο (με την βοήθεια των αντίστοιχων συναρτήσεων). Με τον τρόπο αυτό **ανεβαίνουμε** επίπεδα. Σε **κάθε επίπεδο**, το **μέγεθος της εικόνας** μειώνεται στο **μισό**, το **μέγεθος των μακρομπλοκ** επίσης μειώνεται στο **μισό** και το ίδιο ισχύει και για την **περιοχή αναζήτησης** (λόγω της υποδειγματοληψίας -**μειώνουμε** τις αντίστοιχες τιμές των μεταβλητών στον κώδικα). Έχοντας πλέον φτάσει **στην κορυφή της πυραμίδας (l=4)**, στο σημείο εκείνο δηλαδή που το **μέγεθος των μακρομπλοκ** είναι αρκετά **μικρό** και η **περιοχή αναζήτησης** αρκετά **μικρή,** πραγματοποιούμε μια **πλήρης αναζήτηση.** Πιο αναλυτικά, υπολογίζουμε το **διάνυσμα κίνησης** στο ανώτατο αυτό επίπεδο (με την χρήση της αντίστοιχης συνάρτησης) και το συγκρίνουμε με **υποψήφια** στα **κατώτερα επίπεδα**.

Πως πραγματοποιείται η **κατάβαση των επιπέδων**?

- Εργαζόμαστε μέσα σε μια **λούπα μέχρι** να φτάσουμε στο **κατώτατο επίπεδο** (l=1). Για να κατέβουμε τα επίπεδα θα **πολλαπλασιάσουμε επί 2** την **περιοχή αναζήτησης** (k), τις **διαστάσεις των macroblocks** (block_size), καθώς και τις **συντεταγμένες (x,y)** που δείχνουν την **πάνω αριστερή γωνία** των macroblocks. Ακόμη θα **πολλαπλασιάσουμε επί 2** τις διαστάσεις του **καλύτερου διανύσματος κίνησης** που βρέθηκε στο **ανώτατο επίπεδο**. Έπειτα θα πραγματοποιήσουμε **αναδειγματοληψία κατά 2** (με την κλήση της αντίστοιχης συνάρτησης) τόσο στο **πλαίσιο αναφοράς** όσο και **στο πλαίσιο στόχο**. Η **περιοχή**

αναζήτησης στο **ανώτερο επίπεδο,** έχουμε βρει πως ισούται με **k=4**. Επομένως στο **επόμενο κατώτερο** επίπεδο θα έχουμε άλλες **οκτώ υποψήφιες θέσεις** πιθανής βελτίωσης του **διανύσματος κίνησης**. Οι συγκεκριμένες υποψήφιες θέσεις είναι εκείνες που θα προκύψουν από τη διαδικασία της **αναδειγματοληψίας** με την αύξηση των γραμμών και των στηλών, δηλαδή την αύξηση των **διαστάσεων** του πλαισίου αναφοράς. Από τις υποψήφιες αυτές θέσεις αναζητούμε εκείνη που θα έχει τη **μικρότερη τιμή της μετρικής SAD**. Αν βρεθεί **καλύτερο διάνυσμα κίνησης** τότε το **αντικαθιστούμε** το προηγούμενο διάνυσμα με το νέο.

- Έχοντας πλέον **φτάσει** στο **κατώτατο επίπεδο**, προσθέτουμε σε μια λίστα το καλύτερο μακρομπλοκ ταιριάσματος που βρήκαμε νωρίτερα.
- Ξεκινάμε να υπολογίζουμε το **προβλεφθέν πλαίσιο** με κάθε νέο μακρομπλοκ που βρίσκουμε
- Έχοντας πλέον βρει τα **καλύτερα μακρομπλοκ ταιριάσματος** για το κάθε μακρομπλοκ του **πλαισίου στόχου**, η συνάρτηση πλέον **τερματίζει**.

Η συνάρτηση **επιστρέφει** δύο πράγματα:

1. μια λίστα με τα **καλύτερα διανύσματα κίνησης** για το κάθε **μακρομπλόκ** και
2. το **προβλεφθέν** πλαίσιο.

Βοηθητικές συναρτήσεις:

1. **get_search_area**, για τον υπολογισμό της περιοχής αναζήτησης στο πλαίσιο αναφοράς
2. **downsampling**, για την υποδειγματοληψία κατά 2 των 2 πλαισίων (αναφοράς και στόχου)
3. **upsampling,** για την αναδειγματοληψία ανά 2 των 2 πλαισίων(αναφοράς και στόχου)
4. **get_motion_vector**, για τον εντοπισμό του καλύτερου διανύσματος κίνησης
5. **get_sad**, για το υπολογισμό της μετρικής sad
6. **motion_compensate**, για τον υπολογισμό του υποψήφιου μακρομπλοκ έχοντας βρει το διάνυσμα κίνησης
7. **pFrame**, για το υπολογισμό του προβλεφθέν πλαισίου
8. **error_image**, για το υπολογισμό της εικόνας σφάλματος πρόβλεψης

1. get_search_area

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. τις **συντεταγμένες (x,y)** της πάνω αριστερής γωνίας του μακρομπλοκ
2. το **πλαίσιο αναφοράς** (referenceFrame)
3. το **μέγεθος του μακρομπλοκ** (block_size)
4. την **ακτίνα αναζήτησης** (k)

Λειτουργία συνάρτησης:

Υπολογίζουμε τα **όρια** της περιοχής αναζήτησης, δηλαδή από ποια **γραμμή και στήλη ξεκινάει** ο **ορισμός της περιοχής** και σε ποια **γραμμή και στήλη τελειώνει**.

π.χ. Αν έχουμε το μακρομπλοκ (0,0) πάνω αριστερή γωνία, k = 32 και block_size = 64 τότε για την περιοχή αναζήτησης θα έχουμε:

- → Η **γραμμή** που θα **ξεκινάει** η περιοχή αναζήτησης **(start_row)** θα είναι η:
  x – k (0 – 32 = **-32**), **εκτός** αν είμαστε στην αρχή του πίνακα οπότε **δεν** μπορούμε να βγούμε εκτός ορίων και η γραμμή που θα ξεκινάει, θα είναι η συντεταγμένη **x** του μακρομπλοκ, δηλαδή **start_row = x = 0**.
- → Η **γραμμή** στην οποία θα **τελειώνει** η περιοχή αναζήτησης **(finish_row)** θα είναι η:
  x + block_size + k (0 + 64 + 32 = **96**), **εκτός** αν είμαστε στο **τέλος** του πίνακα ή έχουμε **ξεπεράσει** τα όρια του πίνακα, οπότε η γραμμή θα είναι η **τελευταία γραμμή** του δισδιάστατου πίνακα, δηλαδή **finish_row = len(referenceFrame)**.
- → Η **στήλη** που θα **ξεκινάε**ι η περιοχή αναζήτησης **(start_col)** θα είναι η:
  y – k (0 – 32 = **-32**), **εκτός** αν είμαστε στην αρχή του πίνακα οπότε **δεν** μπορούμε να βγούμε εκτός ορίων και η στήλη που θα ξεκινάει θα είναι η συντεταγμένη **y** του μακρομπλοκ, δηλαδή **start_col = y = 0**.
- → Η **στήλη** στην οποία θα **τελειώνει** η περιοχή αναζήτησης **(finish_col)** θα είναι η: **y + block_size + k** (0 + 64 + 32 = **96**), **εκτός** αν είμαστε στο **τέλος** του πίνακα ή έχουμε **ξεπεράσει** τα όρια του πίνακα, οπότε η στήλη θα είναι η **τελευταία στήλη** του δισδιάστατου πίνακα, δηλαδή **finish_col = len(referenceFrame[0])**.

Η συνάρτηση αυτή **επιστρέφει**:

1. Την **περιοχή αναζήτησης** του πλαισίου αναφοράς βάση τις τιμές που υπολόγισε παραπάνω (start_row, finish_row, start_col, finish_col)
2. το **start_row**
3. το **finish_row**
4. το **start_col**
5. το **finish_col**

2. subsampling

Η συνάρτηση υλοποιεί μια απλή μέθοδο **υποδειγματοληψίας** (subsampling) μείωσης της ανάλυσης μιας εικόνας. Δέχεται μια **εικόνα** ως είσοδο, η οποία αναπαρίσταται ως ένας **πίνακας**.

Λειτουργία συνάρτησης:

Η συνάρτηση δέχεται σαν όρισμα έναν πίνακα δύο διαστάσεων και παράγει έναν πίνακα που θα περιλαμβάνει μόνο κάθε επόμενη στήλη και γραμμή. Με αυτό το τρόπο επιτυγχάνεται η μείωση του μεγέθους του πίνακα δια δύο.

Η συνάρτηση αυτή **επιστρέφει** το αποτέλεσμα, το οποίο είναι ο **υποδειγματοληπτημένος πίνακας** (down_sampled), αντιπροσωπεύοντας την εικόνα με μειωμένη ανάλυση.

3. upsampling

Η συνάρτηση υλοποιεί μια απλή μέθοδο **αναδειγματοληψίας** (upsampling) για τον διπλασιασμό των διαστάσεων της εικόνας. Δέχεται μια **εικόνα** ως είσοδο, η οποία αναπαρίσταται ως ένας **πίνακας**.

Λειτουργία συνάρτησης:

Η συνάρτηση δέχεται σαν **όρισμα** έναν πίνακα **δύο** διαστάσεων και παράγει έναν πίνακα με **διπλάσιο μέγεθος**. Η **διαδικασία** αυτή πραγματοποιείται με τη χρήση της μεθόδου του **πλησιέστερου γείτονα**. Σύμφωνα με αυτή τη μέθοδο οι **νέες γραμμές και στήλες** που προστίθενται παίρνουν **τη τιμή του εικονοστοιχείου (pixel) του πλησιέστερου γείτονα**. Με το τρόπο αυτόν επιτυγχάνεται **ο διπλασιασμός του μεγέθους του πίνακα.**

Η συνάρτηση αυτή **επιστρέφει** το αποτέλεσμα, το οποίο είναι ο **αναδειγματοληπτημένος πίνακας** (up_sampled).

4. get_motion_vector

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. το **πλαίσιο αναφοράς** (referenceFrame)
2. το **πλαίσιο στόχο** (targetFrame)
3. τις **συντεταγμένες** (x,y) της πάνω αριστερής γωνίας του μακρομπλοκ
4. το **μέγεθος του μακρομπλοκ** (blockSize)
5. την **ακτίνα αναζήτησης** (k)
6. το **start_row**
7. το **finish_row**
8. το **start_col**
9. το **finish_col**

Λειτουργία συνάρτησης:

Ορίζουμε αρχικά ως **καλύτερο διάνυσμα κίνησης** (best_mv) το **μηδενικό διάνυσμα**, δηλαδή **best_mv = (0, 0)** και ως **καλύτερο sad** (best_sad) το **άπειρο**, δηλαδή **best_sad = float('inf').**

Στην συνέχεια, **εντός** των ορίων της περιοχής αναζήτησης υπολογίζουμε την **μετρική SAD** και την συγκρίνουμε κάθε φορά με την καλύτερη μετρική, αν είναι δηλαδή **μικρότερη** αυτής και αν ναι τότε θέτουμε ως **καλύτερη sad** την sad που μόλις υπολογίσαμε και κρατάμε τις **νέες συντεταγμένες** του διανύσματος κίνησης. Η διαδικασία **τερματίζει** όταν υπολογίσουμε την μετρική για όλα τα σημεία της περιοχής αναζήτησης.

Η συνάρτηση αυτή **επιστρέφει** το **καλύτερο διάνυσμα κίνησης** (best_mv) όπως υπολογίστηκε με την βοήθεια της μετρικής **SAD**.

5. get_sad

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. το **πλαίσιο αναφοράς** (referenceFrame)
2. το **πλαίσιο στόχο** (targetFrame)
3. τις **συντεταγμένες** (row, column) του διανύσματος κίνησης
4. τις **συνταγμένες** της **πάνω αριστερής γωνίας** του τρέχον μακρομπλοκ (current_p)
5. το **μέγεθος του μακρομπλοκ** (block_size)

Λειτουργία συνάρτησης:

Θέτουμε αρχικά μια μεταβλητή με όνομα **sad = 0**. Υπολογίζουμε την μετρική **sad** για τις συγκεκριμένες τιμές **(row, column)** του διανύσματος κίνησης και **επιστρέφουμε** την τιμή sad που υπολογίστηκε.

6. motion_compensate

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. τις **συνταγμένες** της **πάνω αριστερής γωνίας** του τρέχον μακρομπλοκ (position)
2. τις **συντεταγμένες** του καλύτερου διανύσματος κίνησης που υπολογίστηκε παραπάνω (motion_vector)

Λειτουργία συνάρτησης:

Θέτουμε αρχικά **x,y** τις συντεταγμένες της πάνω αριστερής γωνίας του **τρέχον μακρομπλοκ**. Έπειτα υπολογίζουμε τις συντεταγμένες της πάνω αριστερής γωνίας του **καλύτερου δυνατού ταιριάσματος μακρομπλοκ** με την βοήθεια του **διανύσματος κίνησης** που υπολογίστηκε νωρίτερα. Επομένως θα ισχύει ότι:

- το **νέο x** θα είναι (newX) = x (=παλιό x) + motion_vector[0] (=x macroblock)
- το **νέο y** θα είναι (newY) = y (=παλιό y) + motion_vector[1] (=y macroblock)

Επιστρέφουμε τις νέες τιμές x,y που υπολογίσαμε προηγουμένως.

7. pFrame

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. το **πλαίσιο στόχο** (targetFrame)
2. το **πλαίσιο αναφοράς** (referenceFrame)
3. τις **συνταγμένες** (x,y) της **πάνω αριστερής γωνίας** του τρέχον μακρομπλοκ
4. τις **συντεταγμένες** (newX, newY) του καλύτερου διανύσματος κίνησης που υπολογίστηκε παραπάνω
5. το **μέγεθος του μακρομπλοκ** (block_size)

Λειτουργία συνάρτησης:

Πραγματοποιεί την **αντικατάσταση** του μακρομπλοκ του **πλαισίου στόχου** με το **κατάλληλο** μακρομπλοκ του **πλαισίου αναφοράς** που βρέθηκε με την βοήθεια του **διανύσματος κίνησης**.

Επιστέψει το προβλεφθέν πλαίσιο.

8. error_image

Η συνάρτηση αυτή υπολογίζει την **εικόνα σφάλματος πρόβλεψης**. Παίρνει ως **ορίσματα**:

1. το **προβλεφθέν πλαίσιο** (pFrame)
2. το **πλαίσιο στόχος** (targetFrame)

Λειτουργία συνάρτησης:

Αφαιρεί από το **προβλεφθέν πλαίσιο** το **πλαίσιο στόχο** και το **επιστρέφει.**

Κωδικοποιητής

Συνάρτηση κωδικοποιητή (encoder)

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. την **εικόνα σφάλματος** πρόβλεψης (error_image)
2. μια λίστα με τα **διανύσματα κίνησης** (mv_list)

Λειτουργία συνάρτησης:

Συμπιέζει την **εικόνα σφάλματος** χρησιμοποιώντας αλγόριθμο **με απώλειες** (JPEG encoding) και τα **διανύσματα κίνησης** χρησιμοποιώντας αλγόριθμο **χωρίς απώλειες** (Huffman encoding).

Επιστρέφει τα κωδικοποιημένα διανύσματα κίνησης και την κωδικοποιημένη εικόνας σφάλματος.

Αποκωδικοποιητής

Συνάρτηση αποκωδικοποιητή (decoder)

Η συνάρτηση αυτή παίρνει ως **ορίσματα**:

1. την **κωδικοποιημένη εικόνα σφάλματος** (encodingErrorImage)
2. τα **κωδικοποιημένα διανύσματα κίνησης** (encoding_mvs)
3. το **δέντρο Huffman** που προέκυψε από την κωδικοποίηση των διανυσμάτων (tree)
4. το **πλαίσιο αναφοράς** (referenceFrame)

Λειτουργία συνάρτησης:

Γίνεται η **αποκωδικοποίηση** των **διανυσμάτων κίνησης** και της **εικόνας σφάλματος.**
Πραγματοποιείται η **ανακατασκευή του πλαισίου στόχου** με τον εξής τρόπο:

Γεμίζουμε αρχικά όλα τα **μακρομπλοκ** του **πλαισίου στόχου** με τα κατάλληλα **μακρομπλοκ** του **πλαισίου αναφοράς** όπως προέκυψαν με την βοήθεια των **αποκωδικοποιημένων διανυσμάτων κίνησης**. Στην συνέχεια στο πλαίσιο που ανακατασκευάστηκε **προσθέτουμε** την αποκωδικοποιημένη **εικόνα σφάλματος** και έτσι λαμβάνουμε το **πλαίσιο στόχο**.

Η συνάρτηση επιστρέφει το **ανακατασκευασμένο πλαίσιο**.

Βοηθητικές συναρτήσεις για κωδικοποιητή/αποκωδικοποιητή:

1. **jpeg_encoder,** για την κωδικοποίηση της εικόνας σφάλματος με χρήση JPEG
2. **jpeg_decoder,** για την αποκωδικοποίηση της εικόνας σφάλματος με χρήση JPEG
3. **Huffman_encoding,** (αναλύθηκε στο 1^{ο} υποερώτημα του 1^{ου} θέματος)
4. **Huffman_decoding,** (αναλύθηκε στο 1^{ο} υποερώτημα του 1^{ου} θέματος)

<u>jpeg_encoder:</u>

Το πρότυπο Jpeg βασίζεται σε μια τεχνική κωδικοποίησης μετασχηματισμού εικόνας που χρησιμοποιεί τον Διακριτό Μετασχηματισμό Συνημίτονου (DCT). Η jpeg κωδικοποίηση ακολουθεί τα εξής βήματα:

1. Μετατροπή εικόνας εισόδου στο color space " YCrCb"
2. Διαχωρισμός εικόνας στις συνιστώσες Y, Cr, Cb
3. Διαίρεση κάθε συνιστώσας σε 8 x 8 block
4. Υποδειγματοληψία των συνιστώσων Cr, Cb κατα το ¼ του αρχικού μεγέθους
5. Κωδικοποίηση DCT σε κάθε block κάθε συνιστώσας
6. Κβάντιση σε κάθε block κάθε συνιστώσας
7. Διαχωρισμός σε DC και AC συντελεστές
8. DCPM κωδικοποίηση DC συντελεστή
9. Διάταξη ζικ-ζακ και κωδικοποίηση μήκους διαδρομής στους AC συντελεστές
10. Ένωση όλων των block κάθε συνιστώσας σε μία κοινή εικόνα (την νέα κωδικοποιημένη εικόνα)
11. Κωδικοποίηση εντροπίας με Huffman

Αναλυτικότερα:

1. **<u>Μετατροπή εικόνας εισόδου στο color space " YCrCb"</u>**

Μπορεί να δοθεί ως είσοδος οποιαδήποτε εικόνα, αλλά πάντοτε πρέπει να πρώτα να μετατραπεί στον YCrCb χρωματικό χώρο, για να διαχωριστεί η πληροφορία χρώματος από την φωτεινότητα. Χρησιμοποιώντας την έτοιμη συνάρτηση της openCV *"cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)"* μετατρέπω την εικόνα από BGR σε YCrCb

2. **<u>Διαχωρισμός εικόνας στις συνιστώσες Y, Cr, Cb</u>**

Η εικόνα διαχωρίζεται στα κανάλια/συνιστώσες Y, Cr, Cb προκείμενου να επεξεργαστούμε κάθε ένα ξεχωριστά και να το χωρίσουμε σε block. Χρησιμοποιώντας την έτοιμη συνάρτηση της openCV *"Y, Cr, Cb= cv2.split(frame)"* χωρίζω την εικόνα στα κανάλια (channels) Y, Cr, Cb

3. **<u>Διαίρεση κάθε συνιστώσας σε 8 x 8 block</u>**

Στη συνέχεια, κάθε κανάλι επεξεργάζεται ξεχωριστά. Αρχικά, ορίζουμε τους πίνακες κβάντισης όπου το κανάλι Y έχει διαφορετικό πίνακα από τα Cr, Cb καθως αυτά αργότερα θα υποδειγματοληπτηθουν. Τους πίνακες αυτούς θα τους χρησιμοποιήσουμε αργότερα στην διαδικασία της κβάντισης και τους παρέχει το jpeg πρότυπο. Έπειτα, βίσκω το μέγεθος των block με την εντολή *.shape[] (blockSize = quantizationMatrix_Y.shape[0])* και τον αριθμό των block για κάθε διάσταση της εικόνας

*"*for height -> *blocksHeight = frame.shape[0] // blockSize"* . Προκειμένου να αποθηκεύσουμε κάπου το κβαντοποιημένο frame που θα κατασκευαστεί, δημιουργώ ένα array και το αρχικοποιώ με την τιμή 0 για κάθε θέση του array χρησιμοποιώντας την συνάρτηση *"numpy.zeros_like(a, dtype)"* . Χρησιμοποιώντας for loop προσπελαύνουμε τα block κάθε συνιστώσας προκειμένου να κάνουμε τους απαραίτητους μετασχηματισμούς.

4. **<u>Υποδειγματοληψία των συνιστώσων Cr, Cb κατα το ¼ του αρχικού μεγέθους</u>**

Στην YCrCb αναπαράσταση υποδειγματοληπτούνται στο ένα τέταρτο του αρχικού μεγέθους χρησιμοποιώντας την συνάρτηση της openCV *"cv2.resize()"*

5. **<u>Κωδικοποίηση DCT σε κάθε block κάθε συνιστώσας</u>**

Κάθε τμήμα μεγέθους 8 x 8 διέρχεται από τον DCT μετασχηματισμό, ο οποίος λαμβάνει στην είσοδο εικονοστοιχεία f(x,y) και υπολογίζει τους συντελεστές συχνότητας F(u,v). Ο μετασχηματισμός αυτός πραγματοποιείται με την συνάρτηση της openCV *"cv2.dct()"*

6. **<u>Κβάντιση σε κάθε block κάθε συνιστώσας</u>**

Στη συνέχεια, οι DCT συντελεστές F(u,v) κβαντίζονται σύμφωνα με έναν πίνακα κβάντισης που παρέχει το jpeg πρότυπο. Σε αυτό το πρότυπο οι τιμές μπορεί να φαίνονται τυχαίες αλλά στην πραγματικότητα οι τιμές που αφορούν την χαμηλή συχνοτική περιοχή (πάνω αριστερή γωνία) είναι μικρότερες και αυξάνονται καθώς κινούμαστε προς τους συντελεστές υψηλής συχνότητας στις

υπόλοιπες τρεις γωνίες.Η κβάντιση επιτυγχάνεται βάζοντας στο block την τιμή του στρογγυλοποιημένου αποτελέσματος της διαίρεσης του dct block με τον πίνακα κβάντισης *(np.round(block / quantization_matrix))*

7. **<u>Διαχωρισμός σε DC και AC συντελεστές</u>**

Στη συνέχεια οι κβαντισμένοι συντελεστές κωδικοποιούνται σε μία ενδιάμεση αναπαράσταση. Οι DC συντελεστές των διαφόρων τμημάτων κωδικοποιούνται χρησιμοποιώντας παλμοκωδική διαμόρφωση. Ενώ οι AC συντελεστές σαρώνονται πρώτα με τρόπο ζιγκ-ζακ. Η κβάντιση που γίνεται σύμφωνα με τον jpeg πίνακα κβάντισης, παράγει πολύ περισσότερα μηδενικά στην περιοχή των υψηλότερων συχνοτήτων. Επομένως η σάρωση αυτή παράγει μεγαλύτερη διαδρομή μηδενικών προς το τέλος της σάρωσης , διότι οι συντελεστές υψηλής συχνότητας σαρώνονται τελευταίοι. Αυτό με τη σειρά του παράγει σαρωμένους AC συντελεστές χαμηλότερης εντροπίας οι οποίοι και κωδικοποιούνται χρησιμοποιώντας κωδικοποίηση μήκους διαδρομής. Οι DPCM κώδικες των DC συντελεστών, μαζί με τους κωδικοποιημένους AC συντελεστές συνιστούν την ενδιάμεση αναπαράσταση.

8. **<u>DCPM κωδικοποίηση DC συντελεστή</u>**

Εδώ γίνεται η κωδικοποίηση της εντροπίας της διαδρομής των DC και AC συντελεστών. Πριν την κωδικοποίηση εντροπίας οι συντελεστές αυτοί μετατρέπονται σε ενδιάμεσες αναπαράστασεις. Για να αναπαρασταθεί ο DC συντελεστής, υφίσταται πρώτα παλμοκωδική διαμόρφωση. Η DPCM διάφορα αναπαρίσταται ως δυάδα, που περιέχει το πλήθος των δυαδίκων ψηφίων που θα χρησιμοποιηθούν για την κωδικοποίηση της διαφοράς αυτής και την τιμή της διαφοράς.

9. **<u>Διάταξη ζικ-ζακ και κωδικοποίηση μήκους διαδρομής στους AC συντελεστές</u>**

Για τους AC συντελεστές η ενδιάμεση αναπαράσταση χρησιμοποιείται μόνο για τους μηδενικούς συντελεστές. Κάθε μηδενικός AC συντελεστής αναπαρίσταται με δύο σύμβολα όπου το πρώτο αναπαριστά δύο μέρη πληροφορίας το μήκος διαδρομής και το μέγεθος. Μήκος διαδρομής είναι το πλήθος των διαδοχικών μη μηδενικών συντελεστών σύμφωνα με την ζικ-ζακ σάρωση και το μέγεθος είναι το πλήθος των bit που απαιτούνται για την κωδικοποίηση του πλάτους του AC συντελεστή. Το δεύτερο σύμβολο κωδικοποιεί την ένταση του AC συντελεστή.

10. **<u>Ένωση όλων των block κάθε συνιστώσας σε μία κοινή εικόνα (την νέα κωδικοποιημένη εικόνα)</u>**

Μέτα την ολοκλήρωση του μετασχηματισμού όλων των block με τις απαραίτητες συναρτήσεις και μεθόδους για κάθε καναλι ξεχωριστά, ενώνουμε τα block σε μία κοινή εικόνα (την πλεον κωδικοποιημένη εικόνα). Η διαδικασία αυτή υλοποιείται μια απλή εντολή εισχώρισης. Για παράδειγμα, για την συνιστώσα Y χρησιμοποιούμε την εντολή *“encodedFrame[i*blockSize:(i+1)*blockSize, j*blockSize:(j+1)*blockSize, 0] = blockY”*

11. **<u>Κωδικοποίηση εντροπίας με Huffman</u>**

Τέλος κωδικοποιούμε την εντροπία χρησιμοποιώντας την κωδικοποίηση huffman την οποία έχουμε ήδη υλοποιήσει πιο πάνω*” blockY = Huffman_ecoding(blockY)”*.

Output:

Initial frame

Η είκονα του αρχικού frame βρίσκεται στον φάκελο της εργασίας με όνομα initialFrame.png καθώς δεν μπορούσε να εισαχθεί στο document

Encoded frame

Η είκονα του κωδικοποιημένου frame βρίσκεται στον φάκελο της εργασίας με όνομα encodedFrame.png καθώς δεν μπορούσε να εισαχθεί στο document

jpeg_decoder:

Η **jpeg αποκωδικοποίηση** ακολουθεί τα εξής βήματα:

1. Αποκωδικοποίηση εντροπίας με Huffman
2. Υπερδειγματοληψία των συνιστώσων Cr, Cb
3. Απόκβάντιση σε κάθε block κάθε συνιστώσας
4. Αποκωδικοποιήση IDCT σε κάθε block κάθε συνιστώσας
5. Ένωση όλων των block κάθε συνιστώσας σε μία κοινή εικόνα (την νέα αποκωδικοποιημένη εικόνα)
6. Μετατροπή εικόνας εισόδου από το color space " YCrCb" σε "BGR"

1. **Αποκωδικοποίηση εντροπίας με Huffman**

Στην Jpeg αποκωδικοποίηση καλούμαστε να πράξουμε την αντίστροφη διαδικασία της κωδικοποίησης. Αρχικά λοιπον, αφού έχω δηλώσει ξανά τους πίνακες κβάντισης, το block size, τον αριθμό των block σε κάθε διάσταση και το frame array αρχικοποιημένο με 0, προσπελαύνω ξανά κάθε block συνιστώσας πραγματοποιώντας του απαραίτητους μετασχιματισμούς. Πρώτα, αποκωδικοποιούμε την εντροπία με την συνάρτηση αποκωδικοποίσης του Huffman *" blockY = Huffman_decoding(blockY)"*.

2. **Υπερδειγματοληψία των συνιστώσων Cr, Cb**

Στην συνέχεια, υπερδειγματοληπτούμε τα κανάλια Cr, Cb πριν προχωρήσουμε σε περαιτέρω μετασχηματισμούς. Αυτό το επιτυγχάνουμε χρησιμοποιώντας την ίδια συνάρτηση της openCV που χρησιμοποιήσαμε και στην υποδειγματοληψία *"cv2.resize()"* .

3. **<u>Απόκβάντιση σε κάθε block κάθε συνιστώσας</u>**

Πρώτος μετασχηματισμός είναι η αποκβάντιση κάθε block. Στην συνάρτηση του κωδικοποιητή στην διαδικασία της κβάντισης κληθήκαμε να διαιρέσουμε το block με τον πίνακα κβάντισης Επομένως, προκειμένου να αντιστέψουμε αυτή την διαδικασία θα πολλαπλασιάσουμε το block με τον πίνακα κβάντισης.

4. **<u>Αποκωδικοποιήση IDCT σε κάθε block κάθε συνιστώσας</u>**

Επόμενος μετασχηματισμός είναι η αντίστροφη διαδικασία της DCT κωδικοποίησης. Ο μετασχηματισμός αυτός πραγματοποιείται με την συνάρτηση της openCV *"cv2.idct()"*

5. **<u>Ένωση όλων των block κάθε συνιστώσας σε μία κοινή εικόνα (την νέα αποκωδικοποιημένη εικόνα)</u>**

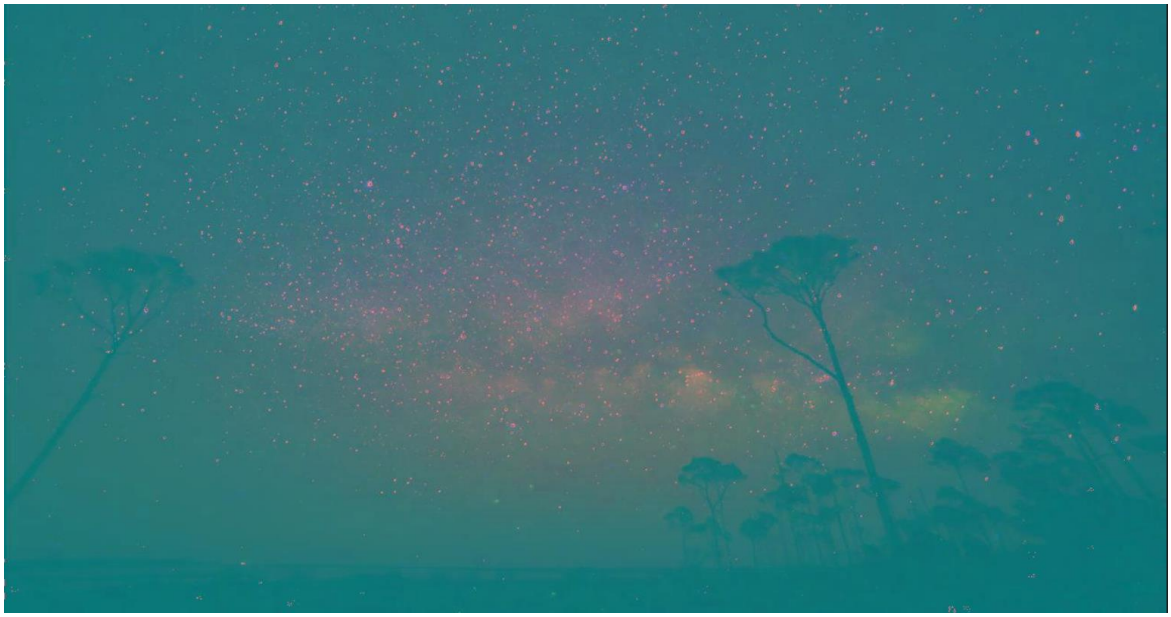
Μέτα την ολοκλήρωση του μετασχηματισμού όλων των block με τις απαραίτητες συναρτήσεις και μεθόδους για κάθε καναλι ξεχωριστά, ενώνουμε τα block σε μία κοινή εικόνα (την πλεον απόκωδικοποιημένη εικόνα). Η διαδικασία αυτή υλοποιείται μια απλή εντολή εισχώρισης. Για παράδειγμα, για την συνιστώσα Y χρησιμοποιούμε την εντολή *"decodedFrame[i * blockSize:(i + 1) * blockSize, j * blockSize:(j + 1) * blockSize, 0] = blockY"*

6. **<u>Μετατροπή εικόνας εισόδου από το color space " YCrCb" σε "BGR"</u>**

Τελευταίο βήμα είναι το αντίστοιχο πρώτο στην διαδικασία της κωδικοποίησης. Πριν επιστρέψουμε πίσω στο πρόγραμμα την αποκωδικοποιημένη εικόνα την μετατρέπουμε από YCrCb color space στο αρχικό της, δηλαδη σε BGR.

<u>Output:</u>

Decoded frame



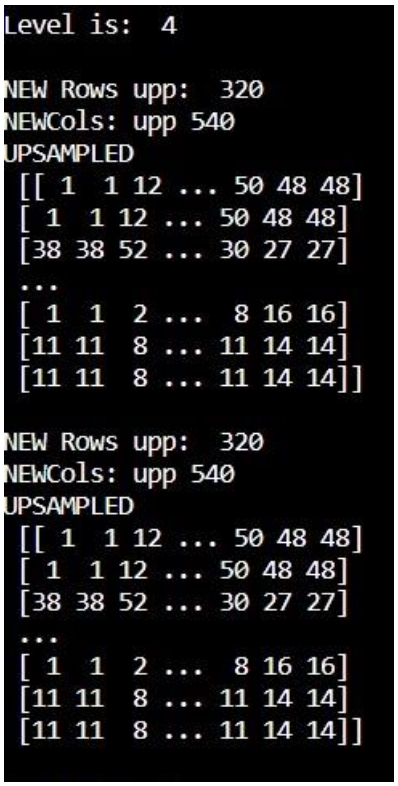
Στιγμιότυπα εκτέλεσης προγράμματος

Το παράδειγμα αφορά ένα **συγκεκριμένο μακρομπλοκ**, στην περίπτωση μας το **(0,0)** -πάνω αριστερή γωνία.

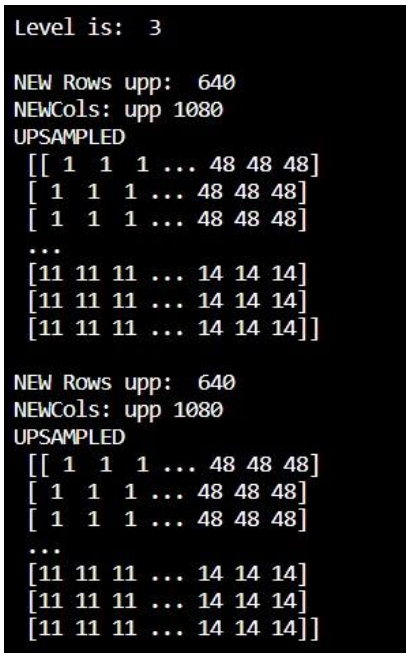
Υποδειγματοληψία κατά 2 (αρχικά πλαίσια στόχου και αναφοράς στο κατώτατο επίπεδο, πλαίσια στόχου και αναφοράς και περιοχή αναζήτησης στο αμέσως επόμενο επίπεδο)

```
Reference Frame:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Target Frame:
 [[ 1  2  6 ... 38 48 50]
 [39 44 50 ... 23 33 34]
 [25 35 37 ... 38 46 65]
 ...
 [ 6 16  8 ...  4 15  9]
 [ 4 15  9 ... 11 16  9]
 [ 8 16  5 ...  8 10  7]]

Height  1280
Width  2160
------------
Rows:  1280
Cols:  2160
Level is: 1

NEW Rows:  640
NEW Cols:  1080
DOWNSAMPLED
 [[ 1  6  0 ... 38 50 48]
 [25 37 35 ... 28 70 46]
 [35 44 40 ... 34 46 45]
 ...
 [10  7 12 ...  7  4 15]
 [ 3  8 14 ...  3 11 15]
 [ 4  9 15 ... 11  9 16]]

NEW Rows:  640
NEW Cols:  1080
DOWNSAMPLED
 [[ 1  6  0 ... 38 50 48]
 [25 37 35 ... 28 70 46]
 [35 44 40 ... 34 46 45]
 ...
 [10  7 12 ...  8  7 13]
 [ 3  8 14 ...  3 11 15]
 [ 4  9 15 ... 11  9 16]]
K is 16
Level is: 2
```

Υπερδειγματοληψία κατά 2 (πλαίσια στόχου και αναφοράς στο **ανώτερο επίπεδο** (level=4)



πλαίσια στόχου και αναφοράς στο **αμέσως επόμενο** επίπεδο (level=3)



εικόνα σφάλματος πρόβλεψης, καλύτερο διάνυσμα κίνσης (24,88), κωδικοποίηση διανύσματος

```
Error image table:
 [[ 36  35  31 ...  10   0  -2]
 [ -2  -7 -13 ...  25  15  14]
 [ 12   2   0 ...  10   2 -17]
 ...
 [  5  -5   3 ...  10  -1   5]
 [  7  -4   2 ...   3  -2   5]
 [  3  -5   6 ...   6   4   7]]
ENCODER...

Symbols with codes:
 {24: '0', 88: '1'}
Encoding mvs:
 ['1', '0']
```

αποκωδικοποίηση διανύσματος

```
Decoding mvs is:
 [(88, 24)]
```

ΘΕΜΑ 2

ΕΚΦΩΝΗΣΗ ΘΕΜΑΤΟΣ 2

Θέμα 2 (1.5 βαθμοί): Σε βίντεο της επιλογής σας, διάρκειας 5s – 10s, στο οποίο υπάρχει ήπια κίνηση αντικειμένου και κάμερας, επιλέξτε ένα αντικείμενο και εξαφανίστε το αλγοριθμικά, αξιοποιώντας την τεχνική αντιστάθμισης κίνησης. Δηλαδή, δημιουργήστε και αποθηκεύστε ένα νέο βίντεο στο οποίο δεν θα υπάρχει το αντικείμενο που επιλέξατε. Υλοποιήστε και τεκμηριώστε το σχετικό σύστημα.