



**ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΙΡΑΙΩΣ**

Εργασία

Συστήματα Πολυμέσων

Ακαδημαϊκό έτος 2018-2019

Π16036 ΙΩΑΝΝΙΔΗΣ ΠΑΝΑΓΙΩΤΗΣ

Π16112 ΠΑΡΑΒΑΝΤΗΣ ΑΘΑΝΑΣΙΟΣ

1. Περιεχόμενα

2. Εισαγωγή	3
3. Άσκηση 6.16.....	4
3.1. Εκφώνηση.....	4
3.2. Εκτέλεση	6
3.3. Επεξήγηση	8
4. Άσκηση 6.17	12
4.1. Εκφώνηση.....	12
4.2. Περιγραφή.....	14
4.3. DPCM.....	14
4.3.1. Εκτέλεση	14
4.3.2. Επεξήγηση	16
4.4. Τροποποιημένος κωδικοποιητής μήκους διαδρομής	19
4.4.1. Εκτέλεση	19
4.4.2. Επεξήγηση	20
5. Άσκηση 8.17	22
5.1. Εκφώνηση.....	22
5.2. Εκτέλεση	24
5.3. Επεξήγηση	26
6. Άσκηση 8.18.....	27
6.1. Εκφώνηση.....	27
6.2. Εκτέλεση	29
6.3. Επεξήγηση	30
7. Βοηθητικές συναρτήσεις	31

7.1. frame_to_macroblocks(frame, window = 16).....	31
7.2. macroblocks_to_frame(macroblocks)	31
7.3. fit_size(x, window = 16).....	32
7.4. get_sad(m_prev, m_next, window = 16):.....	32
7.5. get_best_match(ms_prev, next_row, next_col, m_next, k = 16)	32
8. Αναφορές - Βιβλιογραφία.....	33

2. Εισαγωγή

Στην εργασία του μαθήματος κάναμε την επιλογή να δουλέψουμε με τη γλώσσα προγραμματισμού **Python 3.7** χρησιμοποιώντας μια σειρά από βιβλιοθήκες και δικές μας υλοποιήσεις συναρτήσεων.

Η βιβλιοθήκες που χρησιμοποιήσαμε είναι οι ακόλουθες:

- PIL
- Pillow
- Numpy
- Cv2

Για την εύρεση των εικόνων και βίντεο αναζητήσαμε στο διαδίκτυο κατάλληλο υλικό ώστε να βασίσουμε πάνω σε αυτό όλη την επεξεργασία που απαιτείται στις ασκήσεις. Επειδή η Python είναι μια δυνατή γλώσσα με πολλά εργαλεία και επαρκή τεκμηρίωση, θεωρήσαμε απαραίτητο να δομήσουμε την εργασία ως εξής:

- Ο φάκελος **exercise-6.16** περιλαμβάνει τη λύση της άσκησης 6.16.
- Ο φάκελος **exercise-6.17** περιλαμβάνει τη λύση της άσκησης 6.17.
- Ο φάκελος **exercise-8.17** περιλαμβάνει τη λύση της άσκησης 8.17.
- Ο φάκελος **exercise-8.18** περιλαμβάνει τη λύση της άσκησης 8.18.
- Ο φάκελος **images** περιέχει όλες τις φωτογραφίες που χρησιμοποιούνται στον κώδικα των ασκήσεων.
- Ο φάκελος **videos** περιέχει όλα τα βίντεο που χρησιμοποιούνται στον κώδικα των ασκήσεων.
- Ο φάκελος **screenshots** περιλαμβάνει φωτογραφίες από στιγμιότυπα εκτέλεσης των προγραμμάτων.
- Το αρχείο **multimedia_systems_2019_p16036_p16112.pdf** είναι το παρόν αρχείο και παρέχει όλη την απαραίτητη τεκμηρίωση της εργασίας.

3. Άσκηση 6.16

3.1. Εκφώνηση

Σε αυτήν την προγραμματιστική άσκηση, θα υλοποιήσετε έναν απλό κωδικοποιητή μήκους διαδρομής, ο οποίος θα επεξεργάζεται δεδομένα εικόνας ή οποιονδήποτε δισδιάστατο πίνακα τιμών, κβαντίζοντας αρχικά τις τιμές εισόδου και στη συνέχεια εφαρμόζοντας σε αυτές κωδικοποίηση μήκους διαδρομής. Το πρόγραμμα σας πρέπει να δέχεται ως είσοδο μια εικόνα αποχρώσεων του γκρι και μία τιμή που θα ρυθμίζει τις τιμές των επιπέδων κβάντισης. Μπορείτε να χρησιμοποιήσετε/τροποποιήσετε τον ενδεικτικό πηγαίο κώδικα που σας παρέχουμε ως σημείο εκκίνησης. Η κύρια ιδέα της κωδικοποίησης μπορεί να δοθεί σε τρία βήματα:

- Διαβάστε την εικόνα και κβαντίστε τον πίνακα εισόδου χρησιμοποιώντας την παράμετρο κβάντισης.
- Κωδικοποιήστε το πλήθος επαναλήψεων του πίνακα των στοιχείων, καταγράφοντας πρώτα τη θέση στην οποία απαντάται ένα νέο χρώμα και πραγματοποιώντας νέα καταγραφή μόνο όταν εμφανιστεί νέο χρώμα.
- Οργανώστε την προσκόπτουσα δομή και αποθηκεύστε την αποτελεσματικά.

Υποθέστε ότι πραγματοποιείτε σάρωση της εικόνας κατά γραμμές. Έστω ότι η είσοδος αποτελείται από το μπλοκ εικόνας επιπέδων του γκρι, μεγέθους 5×5 , που φαίνεται παρακάτω και έστω ότι η τιμή της παραμέτρου κβάντισης είναι 10 (10 επίπεδα φωτεινότητας αντιστοιχούν σε ένα βήμα κβάντισης). Κάθε αριθμός αντιπροσωπεύει ένα επίπεδο του γκρι, οπότε οι προσκόπτουσες κβαντισμένες και κωδικοποιημένες τιμές της εξόδου φαίνονται στο σχήμα 6.21.

Απαντήστε τις ακόλουθες ερωτήσεις:

- Για τις παρεχόμενες εικόνες εισόδου, τι λόγος συμπίεσης επιτυγχάνεται κατά μέσο όρο;
- Αυξάνουν ή μειώνονται οι λόγοι συμπίεσης καθώς μεταβάλλεται η τιμή της παραμέτρου κβάντισης;

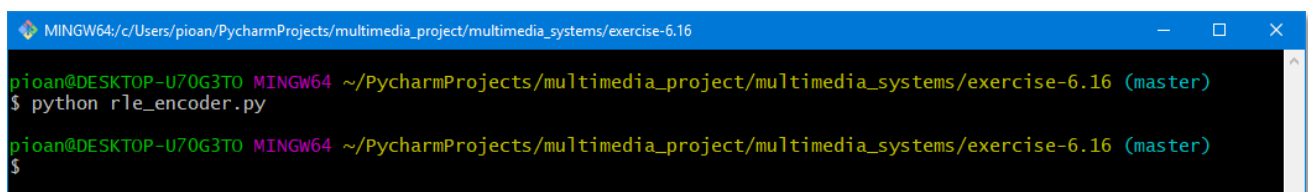
- Υποθέστε ότι πρέπει να αποθηκεύσετε τις συμπιεσμένες εξόδους, έτσι ώστε σε μεταγενέστερο στάδιο να τις διαβάσετε και να τις επεξεργαστείτε. Μπορείτε βέβαια να τις αποσυμπιέσετε και να τις αναπαραστήσετε ως έναν πίνακα-εικόνα. Υπάρχει όμως αποτελεσματικότερος τρόπος αναπαράστασής τους; Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα που παρατηρούνται στην αναπαράστασή σας;

3.2. Εκτέλεση

Η επίλυση της άσκησης βρίσκεται στον φάκελο **exercise-6.16** και συγκεκριμένα στα αρχεία **rle_encoder.py** και **rle_decoder.py**

Το αρχείο **rle_encoder.py** υλοποιεί την κωδικοποίηση μήκους διαδρομής. Για να εκτελέσουμε τον κώδικα ακολουθούμε τα παρακάτω βήματα:

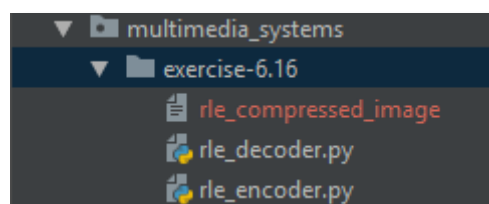
1. Ανοίγουμε την γραμμή εντολών και μεταβαίνουμε στο φάκελο **multimedia_systems/exercise-6.16**
2. Εκτελούμε την εντολή **python rle_encoder.py**



```
MINGW64~/c:/Users/pioan/PycharmProjects/multimedia_project/multimedia_systems/exercise-6.16
pioan@DESKTOP-U70G3TO MINGW64 ~/PycharmProjects/multimedia_project/multimedia_systems/exercise-6.16 (master)
$ python rle_encoder.py
pioan@DESKTOP-U70G3TO MINGW64 ~/PycharmProjects/multimedia_project/multimedia_systems/exercise-6.16 (master)
$
```

Εικόνα 3.1

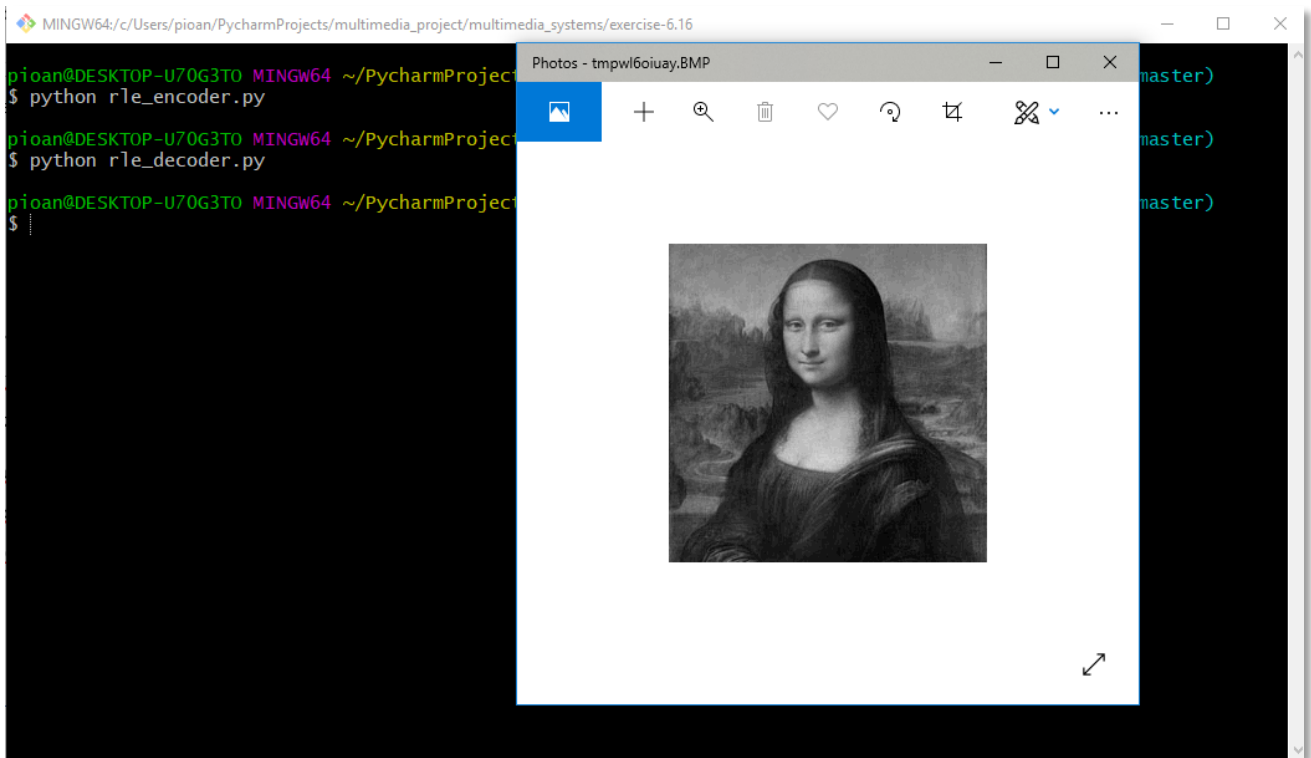
Κατά την εκτέλεση του αρχείου παρατηρούμε ότι στο φάκελο **exercise-6.16** δημιουργείται το αρχείο **rle_compressed_image**. Το αρχείο αυτό περιέχει την κωδικοποιημένη εικόνα σε μορφή string.



Εικόνα 3.2

Στη συνέχεια για να επαναφέρουμε την εικόνα στην αρχική της μορφή εκτελούμε το αρχείο **rle_decoder.py** χρησιμοποιώντας την παρακάτω εντολή βρισκόμενοι πάντα στο φάκελο **exercise-6.16**:

1. `python rle_decoder.py`



Εικόνα 3.3

Για να εκτελέσετε τις περιπτώσεις που ακολουθούν μπορείτε να αλλάξετε τις αντίστοιχες τιμές στον κώδικα, όπως φαίνεται και στις εικόνες. Ακολουθήσαμε αυτή την μέθοδο για να αποφευχθούν τυχών αστοχίες στην εκτέλεση εξαιτίας της λάθος εισαγωγής δεδομένων εισόδου από το χρήστη.

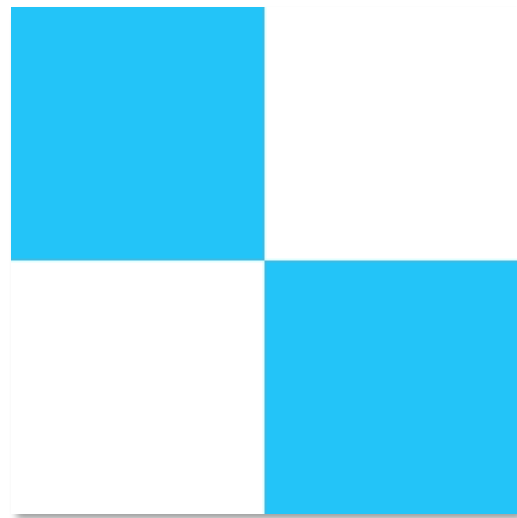
3.3. Επεξήγηση

Για να γίνει καλύτερα κατανοητό το αποτέλεσμα τις άσκησης παρέχουμε δυο εντελώς διαφορετικές εικόνες για συμπίεση.

Πιο συγκεκριμένα την εικόνα **mona-lisa.jpg** οι οποία περιέχει πάρα πολλά χρώματα και την εικόνα **logo.jpg** η οποία αποτελείται στην ουσία από δυο χρώματα. Πριν την διαδικασία της συμπίεσης και οι δυο εικόνες μετατρέπονται από RGB σε GRAYSCALE.



mona-lisa.jpg



logo.jpg

Αρχικά εκτελώντας το αρχείο **rle_encoder.py** συμπιέζουμε την εικόνα **mona-lisa.jpg** με μέγεθος **32.8 KB** έχοντας θέσει το επίπεδο κβαντίσεις ίσο με **10**, εικόνα 3.4.

```
quantization_value = 10

# Convert image to black & white
# Load image with many colors
bw_image = Image.open("../images/mona-lisa.jpg").convert("L")

# Load image with two colors
# bw_image = Image.open("../images/mona-lisa.jpg").convert("L")
```

Εικόνα 3.4

Το αρχείο **rle_compressed_image** που προκύπτει έχει μέγεθος **132 KB**.

Όπως φαίνεται το η συμπιεσμένη πλέον εικόνα έχει μέγεθος **4.02** φορές μεγαλύτερο από το μέγεθος της αρχικής εικόνας.

Επομένως ο λόγος συμπίεσης που επιτυγχάνεται είναι **0.248**.

Στη συνέχεια επιλέγουμε να κβαντίσουμε την εικόνα **logo.jpg** με μέγεθος **8.59 KB** και αμετάβλητο επίπεδο κβάντισες, εικόνα 3.5.

```
quantization_value = 10

# Convert image to black & white
# Load image with many colors
# bw_image = Image.open("../images/mona-lisa.jpg").convert("L")

# Load image with two colors
bw_image = Image.open("../images/logo.jpg").convert("L")
```

Εικόνα 3.5

Αυτή τη φορά το αρχείο **rle_compressed_image** έχει μέγεθος μόνο **3.5 KB**.

Συνεπώς, όπως ευκολά μπορούμε να δούμε, ο λόγος συμπίεσής σε αυτή την περίπτωση είναι **4.68**.

Αυτή η τόσο μεγάλη διαφορά, οφείλεται στο γεγονός ότι τα διαδοχικά pixel της εικόνας mona-lisa.jpg έχουν πάρα πολύ συχνά διαφορετικές τιμές αναπαράστασης χρώματος.

Ενώ στη εικόνα logo.jpg η εναλλαγές στις τιμές αυτές είναι σαφώς πολύ λιγότερες.

Επίσης, σημαντικό ρολό σε όλη τη διαδικασία της συμπίεσης καταλαμβάνει και η τιμή κβαντίσεις που θα επιλέξουμε. Όσο μεγαλύτερη είναι η τιμή κβαντίσεις τόσο μεγαλύτερος είναι και ο λόγος συμπίεσης που επιτυγχάνεται. Αυτό όμως κρύβει και ένα μειονέκτημα, χάνεται αρκετή πληροφορία κατά την συμπίεση και έτσι όταν ξαναδημιουργήσουμε την εικόνα παρατηρούμε ότι είναι αλλοιωμένη. Το αποτέλεσμα φαίνεται στις επόμενες εικόνες όπου ορίζουμε τη κβαντίσει ίση με **50**.

Πρώτα για την εικόνα **mona-lisa.jpg**.

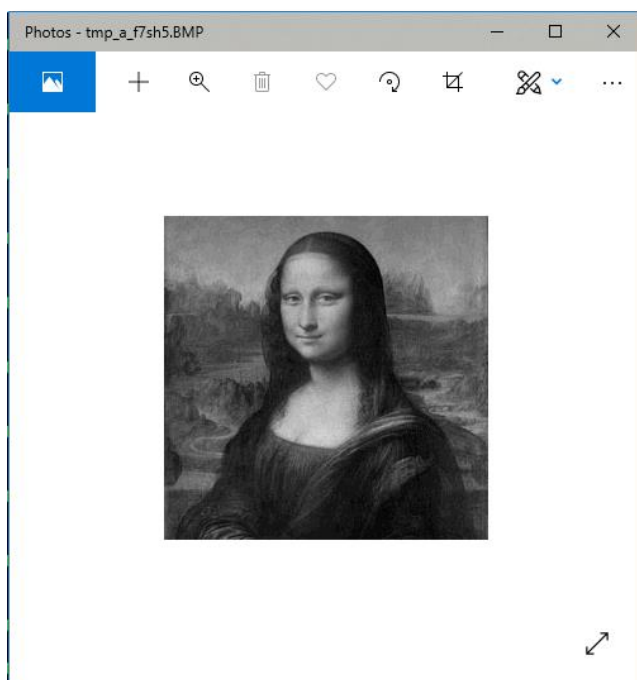
```
quantization_value = 50

# Convert image to black & white
# Load image with many colors
bw_image = Image.open("../images/mona-lisa.jpg").convert("L")

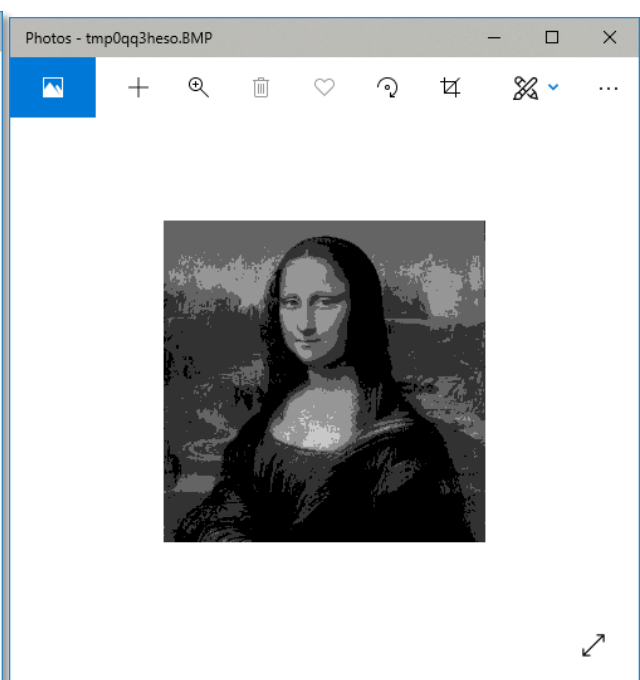
# Load image with two colors
# bw_image = Image.open("../images/logo.jpg").convert("L")
```

Εικόνα 3.6

Παρατηρούμε ότι το **rle_compressed_image** έχει μέγεθος μόνο **29 KB**. Αυτό σημαίνει ότι τώρα έχουμε πέτυχει λόγο συμπίεσης **1.13**, σαφώς καλύτερο από πριν. Αλλά η αποσυμπίεσμένη εικόνα έχει την εξής μορφή:



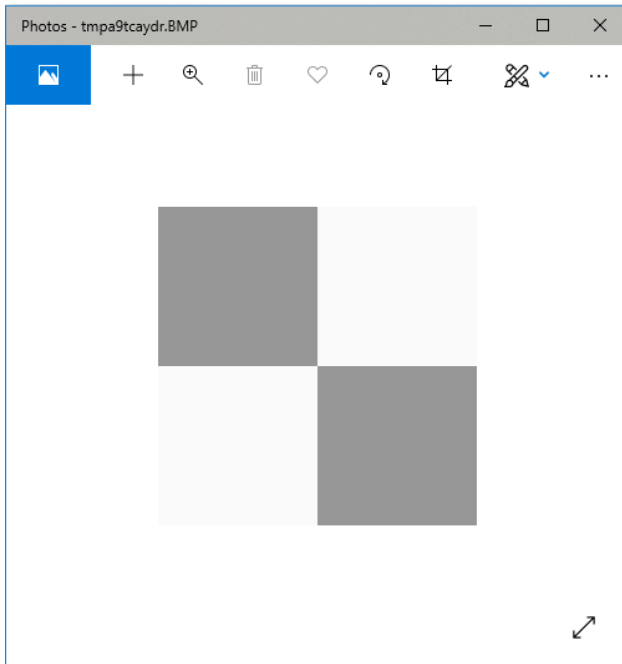
Εικόνα 3.7 – Τιμή κβάντισης = 10



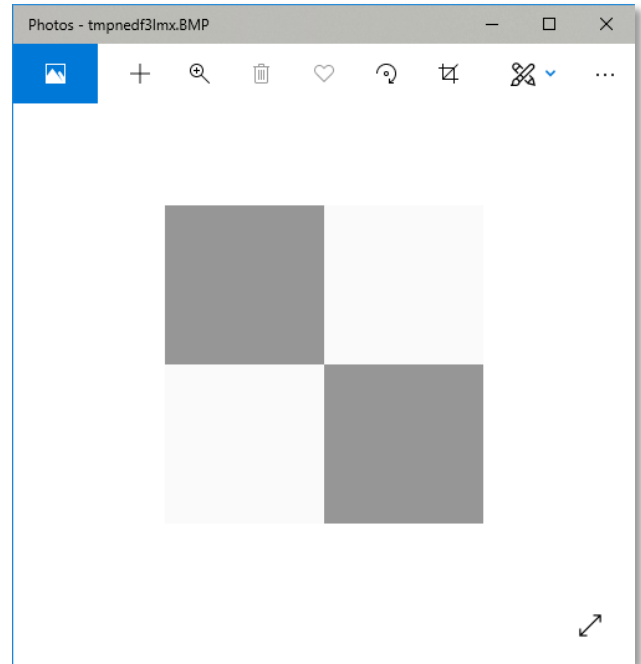
Εικόνα 3.8 - Τιμή κβάντισης = 50

Για την εικόνα **logo.jpg** παρατηρούμε ότι το αρχείο **rle_compressed_image** έχει μέγεθος μόνο **3.0 KB**. Η διαφορά είναι πολύ μικρή σε σχέση με πριν.

Ο λόγος συμπίεσης σε αυτή τη περίπτωση είναι **5.46**. Με την αποσυμπίεσμένη εικόνα να έχει την ίδια μορφή με πριν.



Εικόνα 3.9 - Τιμή κβάντισης = 10



Εικόνα 3.10 - Τιμή κβάντισης = 50

4. Άσκηση 6.17

4.1. Εκφώνηση

Σε αυτή την άσκηση , θα εφαρμόσετε τεχνικές DPCM για να συμπίεσετε βίντεο. Αν και αυτός δεν είναι ο πιο αποδοτικός τρόπος για τη συμπίεση βίντεο, αποτελεί εντούτοις μία 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 μία 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 εξαρτημένο από την τιμή κβάντισης που δέχεται ως είσοδο . Κατά συνέπεια , το πρόγραμμα θα πρέπει να κάνει τα ακόλουθα:

- Ανάγνωση μίας ακολουθίας βίντεο.
- Υπολογισμός των διαφορών διαδοχικών πλαισίων- εικονοστοιχείο προς εικονοστοιχείο.
- Οι διαφορές έχουν χαμηλότερη εντροπία-κβάντιση των τιμών με τρόπο που καθορίζεται από την τιμή της παραμέτρου κβάντισης που δόθηκε στην είσοδο.

Κατά τη δημιουργία του αρχείου εξόδου , καταγράψτε το πρώτο πλαίσιο ως έχει και στη συνέχεια τις κβαντισμένες διαφορές των εικονοστοιχείων για λ πλαίσιο. Απαντήστε στις ακόλουθες ερωτήσεις;

- Τι λόγο συμπίεσης επιτυγχάνετε για τα δεδομένα που έχετε στη διάθεσή σας;
- Για το ίδιο βίντεο, εισάγετε όλα τα πλαίσια στο προηγούμενο πρόγραμμα και υπολογίστε τον συνδυασμένο λόγο συμπίεσης . Ποιο πρόγραμμα υπερτερεί; Γιατί;

Αυτός ο αλγόριθμος επιτυγχάνει συμπίεση και είναι ιδιαίτερα ταχύς. Τι προβλήματα προβλέπετε κατά τη χρήση του σε ένα πραγματικό σενάριο; (Υποθέστε ότι, το εύρος ζώνης δεν είναι πρόβλημα , δηλαδή ο αλγόριθμος συμπιέζει επαρκώς ως προς το διαθέσιμο εύρος ζώνης).

4.2. Περιγραφή

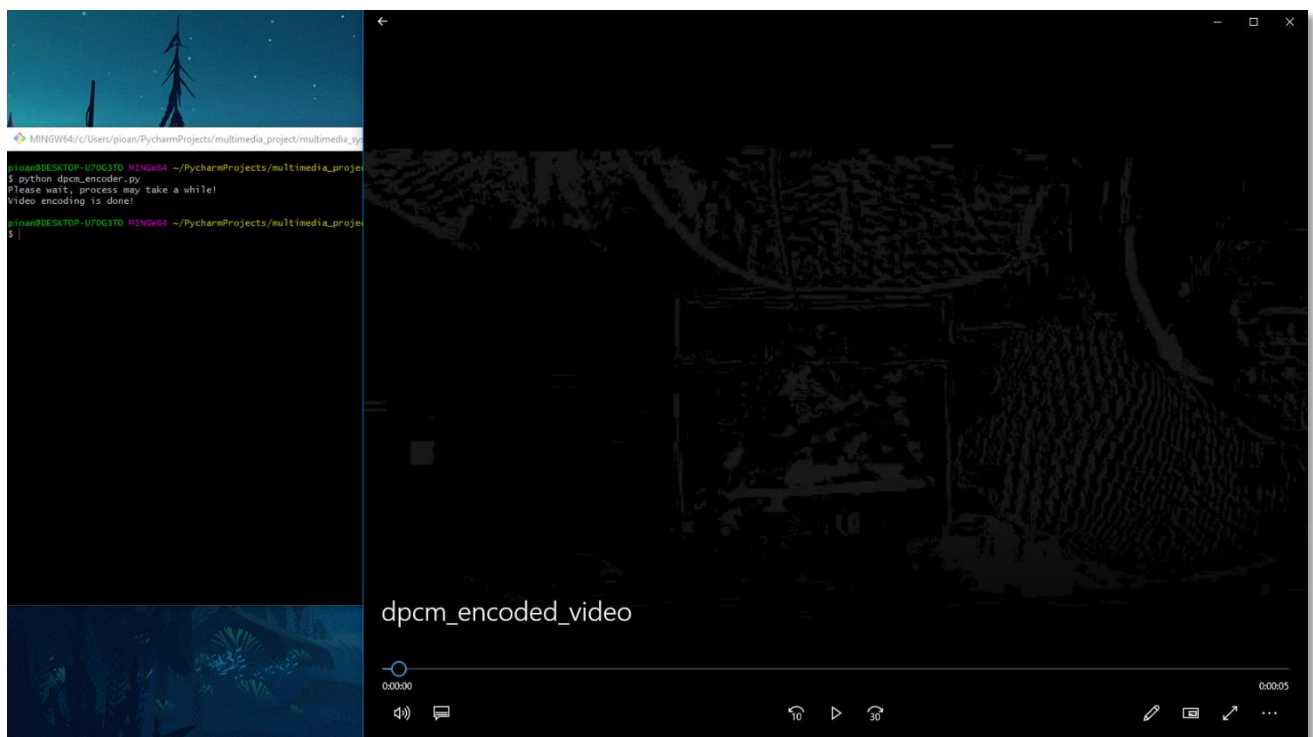
Η επίλυση της άσκησης βρίσκεται στον φάκελο **exercise-6.17**. Τα αρχεία **rle_modified_encoder.py** και **rle_modified_decoder.py** αφορούν τη λειτουργία του τροποποιημένου κωδικοποιητή μήκους διαδρομής και τα αρχεία **dpcm_encoder.py** και **dppcm_decoder.py** αφορούν τη συμπίεση με DPCM.

4.3. DPCM

4.3.1. Εκτέλεση

Αρχικά θα συμπίεσουμε/αποσυμπίεσουμε το βίντεο μας με τεχνικές dpcm.

1. Ανοίγουμε την γραμμή εντολών και μεταβαίνουμε στο φάκελο **multimedia_systems/exercise-6.17**
2. Εκτελούμε την εντολή **python dpcm-encoder.py**



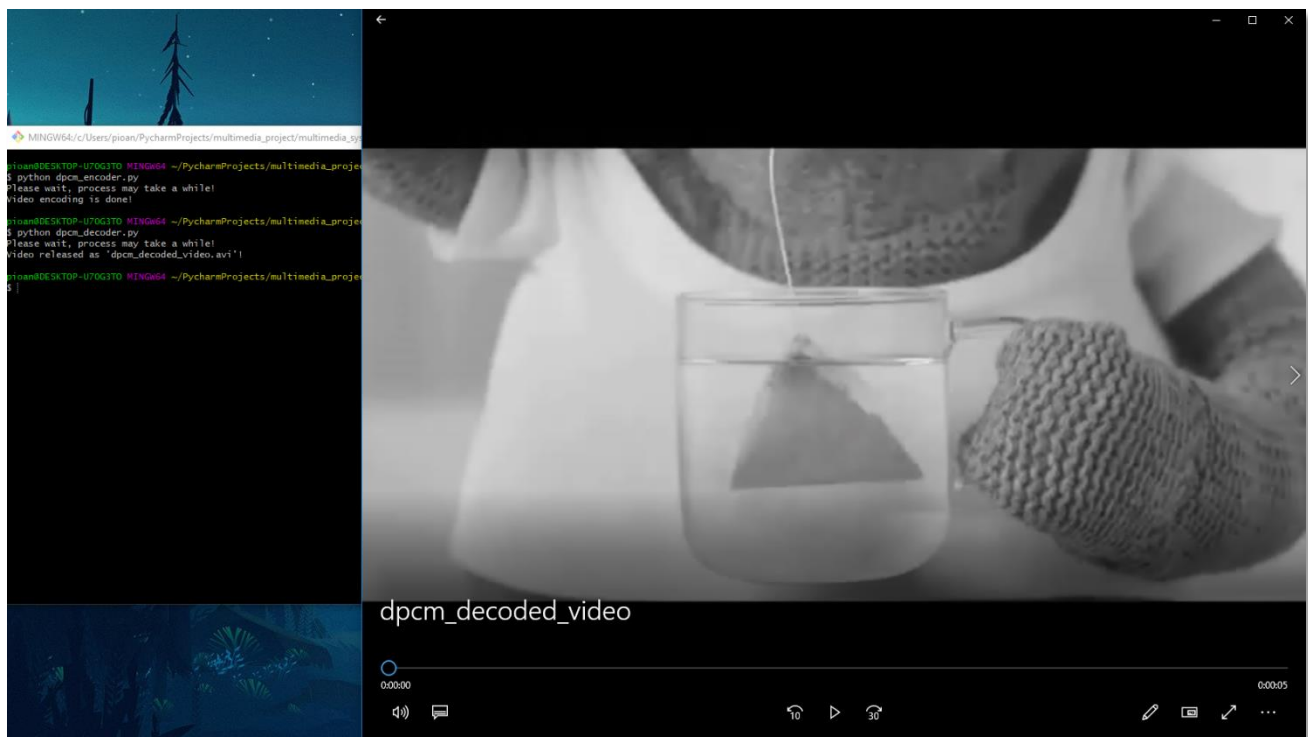
Εικόνα 4.1

Από την εκτέλεση του παραπάνω προγράμματος δημιουργείται το βίντεο με όνομα **dpcm_encoded_video.avi**.

Στη συνέχεια για να αποσυμπιέσουμε το βίντεο αυτό και να δημιουργήσουμε πάλι το αρχικό εκτελούμε το αρχείο **dpcm_decoder.py** χρησιμοποιώντας την παρακάτω εντολή βρισκόμενοι πάντα στο φάκελο **exercise-6.17**:

1. `python dpcm_decoder.py`

Έτσι, δημιουργείται το αποσυμπιεσμένο βίντεο με όνομα **dpcm_decoded_video.avi**.



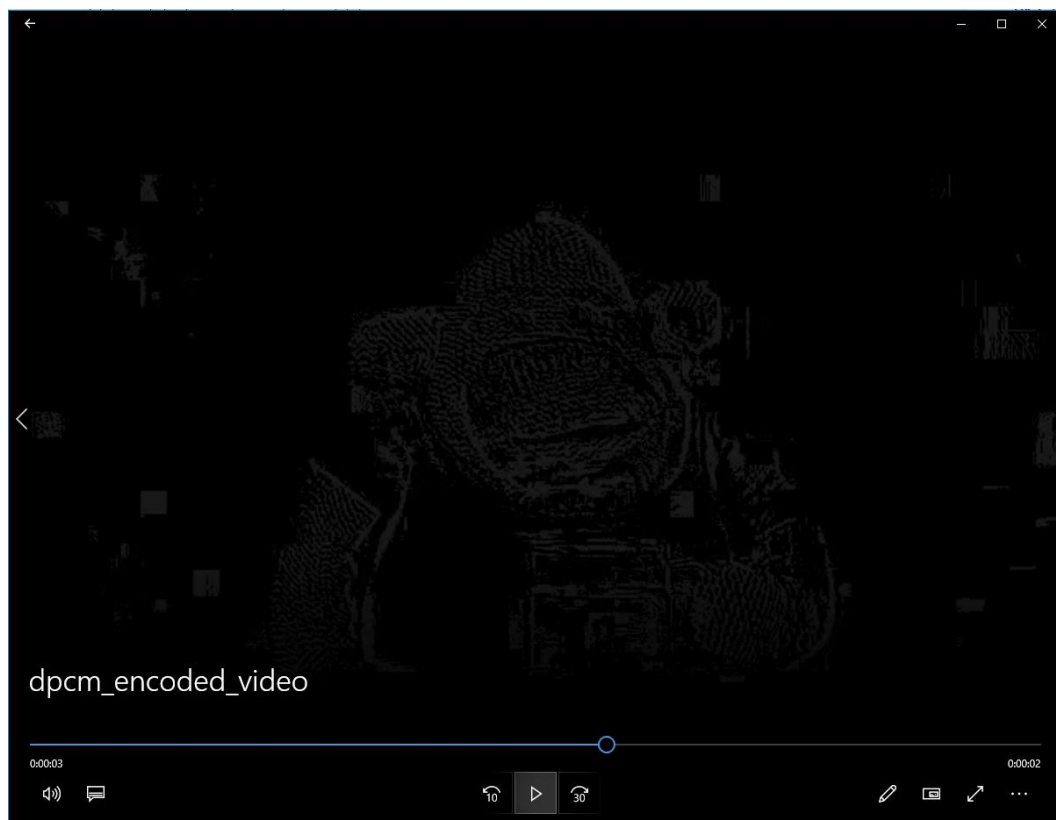
Εικόνα 4.2

4.3.2. Επεξήγηση

Όταν ένα σήμα περιέχει υψηλές συχνότητες μεταβάλλεται αργά, με αποτέλεσμα τα διαδοχικά δείγματα να έχουν παρόμοιες τιμές. Έτσι, λοιπόν, εφόσον έχουν παρόμοιες τιμές, τότε αυτές μπορούν ευκολά να προβλεφθούν χρησιμοποιώντας το τρέχον αλλά και παλιότερα δείγματα. Όσο πλησιέστερα στην πραγματική τιμή βρίσκεται η πρόβλεψη, το μικρότερο είναι και το λάθος πρόβλεψης. Συνεπώς, αντί να μεταδώσουμε αυτούσιο το σήμα, μπορούμε να μεταδώσουμε τα σφάλματα μεταξύ της πρόβλεψης και της πραγματικής τιμής.

Αντί να αποστείλουμε το αρχικό σήμα $y(n)$, υπολογίζουμε τις διαφορές $d(n)$, όπως έχουμε υλοποίηση και στο πρόγραμμα μας. Οπου το $d(n) = y(n) - y(n-1)$. Δηλαδή η διαφορά το τρέχοντος frame με το προηγούμενο, για το σύνολο των frame.

Έτσι δημιουργείται το βίντεο **dpcm_encoded_video.avi**.



Εικόνα 4.3

Όσον αφορά την αποσυμπιεστή του βίντεο ο αποκωδικοποιητής ανακατασκευάζει το σήμα $y'(n)$ με απώλειες, έχοντας ως βάση το κβαντισμένο σήμα διαφόρων $d'(n)$.

$$y'(n) = d'(n)$$

$$y'(n) = y(n-1) + d'(n)$$

Ο αποκωδικοποιητής έχει κατασκευάσει το $y'(n)$ το οποίο είναι ίδιο με το $y(n-1)$, περιλαμβάνοντας όμως και κάποιο λάθος κβάντισης. Το αποτέλεσμα είναι αυτό το λάθος κβαντίσεις να συσσωρεύεται σε κάθε βήμα και τελικά να οδηγούμαστε σε ένα ανακατασκευασμένο βίντεο με αρκετό θόρυβο και έλλειψη σημαντικής λεπτομέρειας.



Εικόνα 4.4

Το συμπιεσμένο βίντεο έχει μέγεθος **2.16 MB**, ενώ το αρχικό έχει μέγεθος **388 KB**. Το νέο ανακατασκευασμένο βίντεο έχει μέγεθος **7.71 MB**.

Ο λόγος συμπίεσης που επιτυγχάνουμε είναι **0,179**.

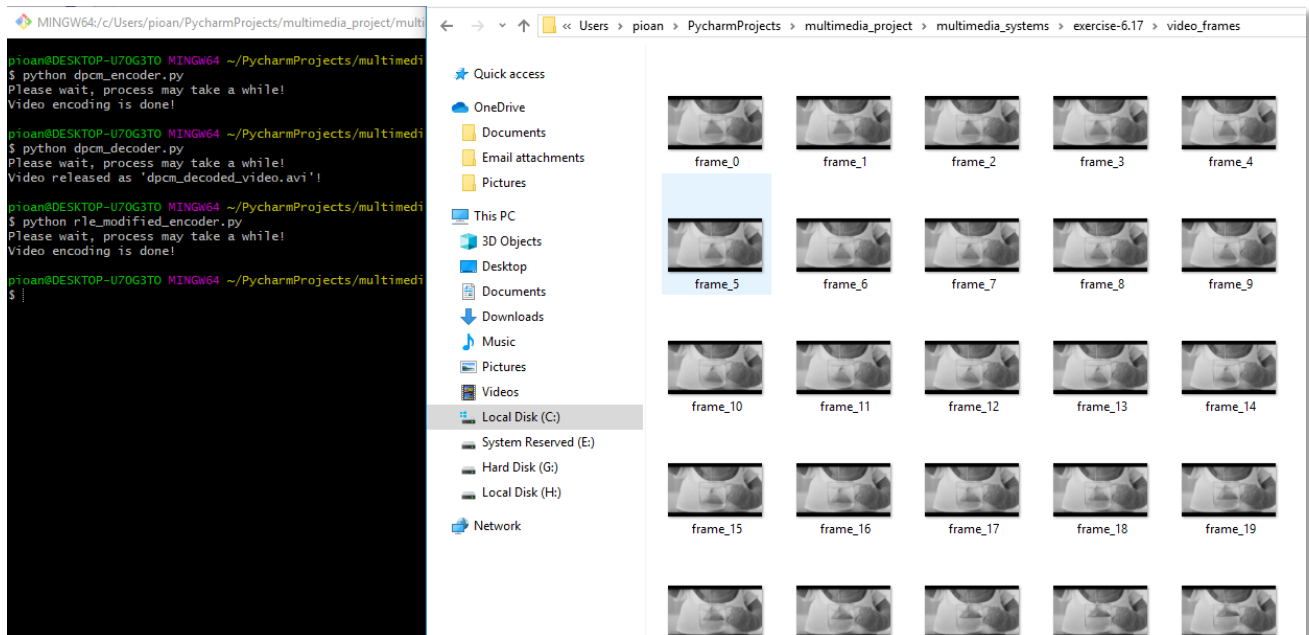
Στην περίπτωση μας παρατηρούμε ότι αυτή η μέθοδος δεν είναι αποδοτική διότι το μέγεθος του συμπιεσμένου βίντεο είναι αρκετά μεγαλύτερο από το αρχικό.

4.4. Τροποποιημένος κωδικοποιητής μήκους διαδρομής

4.4.1. Εκτέλεση

Εκτελούμε το αρχείο **rle_modified_encoder.py** χρησιμοποιώντας την παρακάτω εντολή βρισκόμενοι πάντα στο φάκελο **exercise-6.17**:

1. `python rle_modified_encoder.py`



Εικόνα 4.5

Από την εκτέλεση του παραπάνω προγράμματος δημιουργείται ο φάκελος **video_frames** στον οποίο περιέχονται όλα τα frames από το video μας.

Στη συνέχεια σε κάθε ένα frame εφαρμόζεται κωδικοποίηση μήκους διαδρομής. Το τελικό αποτέλεσμα της κωδικοποίησης βρίσκεται στο αρχείο **rle_compressed_frames**.

Το επόμενο βήμα είναι να ανακατασκευάσουμε πάλι το βίντεο μας χρησιμοποιώντας τα δεδομένα του αρχείου **rle_compressed_frames**.

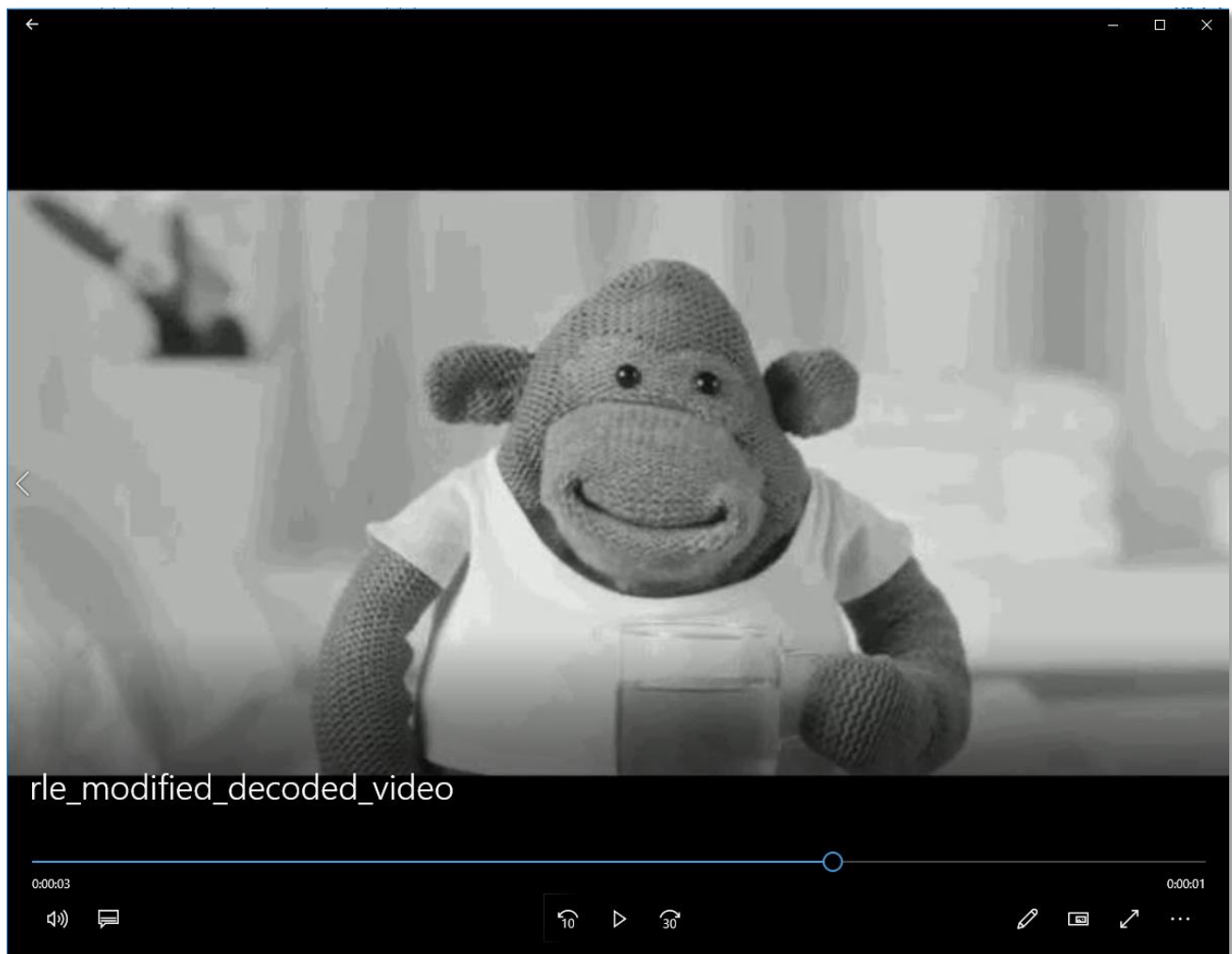
Ενώ είμαστε ακόμα στο φάκελο **exercise-6.17** εκτελούμε το αρχείο **rle_modified_decoder.py** χρησιμοποιώντας την παρακάτω εντολή:

1. `python rle_modified_decoder.py`

Αρχικά, παρατηρούμε ότι ανακατασκευάστηκαν όλα τα πλαίσια το βίντεο. Αυτά βρίσκονται στο φάκελο **decoded_video_frames**. Επιπλέον μπορούμε να δούμε και το ανακατασκευασμένο βίντεο, το οποίο είναι αποθηκευμένο με όνομα **rle_modified_decoded_video.avi**.

4.4.2. Επεξήγηση

Η υλοποίηση του ατμοποιημένου κωδικοποιητή μήκους διαδρομής βασίζεται στον πηγαίο κώδικα για την άσκηση 6.16. Τα επιπλέον βήματα που προστέθηκαν είναι εξαγωγή των πλαισίων του βίντεο και η κωδικοποίηση τους ένα προς ένα. Όπως επίσης η αποκωδικοποίηση σε του string σε πλαίσια και η ανακατασκευή του βίντεο.



Εικόνα 4.6

Το συμπιεσμένο βίντεο έχει μέγεθος **27.3 MB**, ενώ το αρχικό έχει μέγεθος **388 KB**. Το νέο ανακατασκευασμένο βίντεο έχει μέγεθος **937 KB**.

Ο λόγος συμπίεσης που επιτυγχάνουμε είναι **0,014**.

Στην περίπτωση το τροποποιημένου κωδικοποιητή μήκους διαδρομής η συμπιεσμένη πληροφορία ενέχει μέγεθος 27.3 MB, όμως το τελικό ανακατασκευασμένο βίντεο έχει πολύ καλύτερη ποιότητα σε σχέση με την drcm κωδικοποίηση. Όπως και πολύ μικρότερο μέγεθος, μόνο 937 KB, αρκετά κοντά στο μέγεθος του αρχικού βίντεο.

5. Άσκηση 8.17

5.1. Εκφώνηση

Στην άσκηση αυτή θα υλοποιήσετε την τεχνική της αντιστάθμισης κίνησης και θα μελετήσετε πώς επηρεάζει τα σφάλματα πρόβλεψης. Δείγματα αρχείων βίντεο, μαζί με τη σχετική πληροφορία διαμόρφωσης, παρέχονται στην ενότητα αρχείων προς λήψη του ιστοτόπου www.cengage.com. Εκεί θα βρείτε και τον κώδικα για να διαβάσει και να προβάλλει πλαίσια βίντεο. Υποθέστε ότι το πρώτο πλαίσιο θα είναι πάντα ένα πλαίσιο I και ότι τα υπόλοιπα πλαίσια θα είναι τύπου P. Η υπόθεση αυτή ευσταθεί στην περίπτωση των σύντομων ακολουθιών βίντεο που επεξεργάζεστε, που έχουν μήκος το πολύ 100 πλαισίων.

- Στο πρώτο μέρος της άσκησης αυτής, υποθέστε ότι θέλετε να προβλέπετε ολόκληρα P πλαίσια και όχι κατά τμήματα. Η πρόβλεψη κάθε ολόκληρου πλαισίου γίνεται με βάση το προηγούμενο πλαίσιο. Υλοποιήστε μία διαδικασία που να δέχεται είσοδο δύο πλαίσια, υπολογίζει τη διαφορά τους και επιστρέφει ένα πλαίσιο σφαλμάτων. Δεν υπολογίζετε διάνυσμα κίνησης. Να προβάλετε τα πλαίσια σφαλμάτων. Σημειώστε ότι το πληροφοριακό περιεχόμενο του πλαισίου σφαλμάτων θα πρέπει να είναι μικρότερο, συγκρινόμενο με αυτό των πλαισίων.
- Στο δεύτερο βήμα θα υλοποιήσετε τεχνική πρόβλεψης κίνησης, η οποία θα υπολογίζει διανύσματα κίνησης ανά μπλοκ. Κάθε μπλοκ θα έχει το τυπικό MPEG μέγεθος 16×16 . Υλοποιήστε μια συνάρτηση που θα δέχεται είσοδο δύο πλαίσια: ένα πλαίσιο αναφοράς, το οποίο θα χρησιμοποιηθεί κατά την αναζήτηση των διανυσμάτων κίνησης, και ένα πλαίσιο στόχος, το οποίο θα προβλεφθεί. Διαιρέστε το πλαίσιο-στόχο σε μακρομπλοκ μεγέθους 16×16 . Εάν το πλάτος και ύψος του πλαισίου δεν είναι πολλαπλάσιο 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 κείμενο 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 μπλοκ και του προβλεφθέντος. Αφού αυτή η διαδικασία ολοκληρωθεί για όλα τα μπλοκ, θα προκύψει ένα πλαίσιο σφαλμάτων. Να γίνει η πρόβλεψη όλων των πλαισίων σφαλμάτων. Θα διαπιστώσετε ότι τα πλαίσια σφαλμάτων εμφανίζουν σημαντικά μικρότερη εντροπία σε σύγκριση με τη προηγούμενη περίπτωση, μολονότι απαιτείται περισσότερος χρόνος για τον υπολογισμό τους.

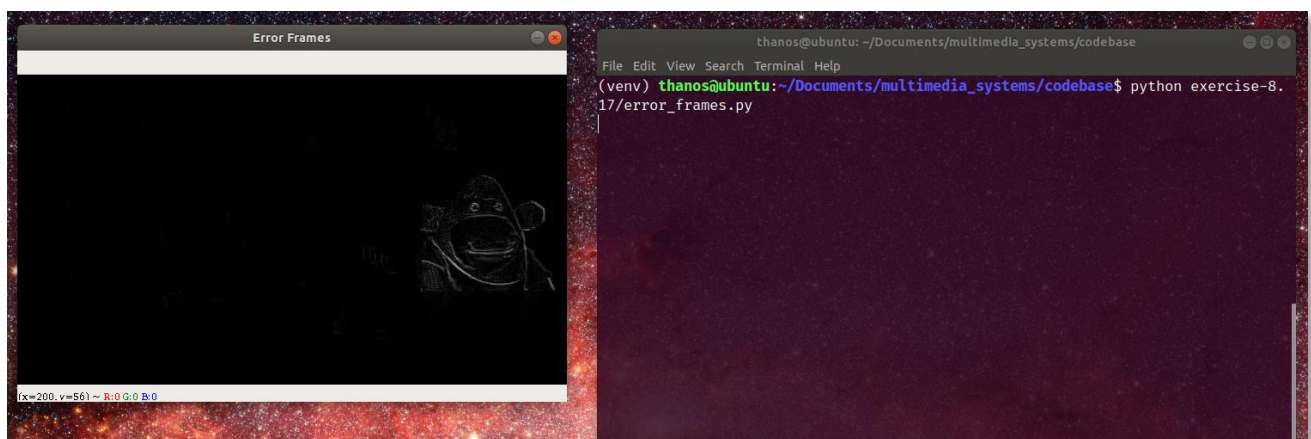
5.2. Εκτέλεση

Η επίλυση της άσκησης βρίσκεται στον φάκελο **exercise-8.17** και συγκεκριμένα στα αρχεία **error_frames.py** και **motion_prediction.py**.

Το πρώτο αρχείο **error_frames.py** υλοποιεί τα πλαίσια σφαλμάτων μεταξύ των πλαισίων του βίντεο. Για να εκτελέσουμε τον κώδικα ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε την γραμμή εντολών και μεταβαίνουμε στο φάκελο **multimedia_systems/exercise-8.17**
2. **python error_frames.py**

Παρατηρούμε ότι εμφανίζεται ένα παράθυρο που δείχνει για όλα τα πλαίσια B τη



Εικόνα 5.1

διαφορά τους με το προηγούμενο:

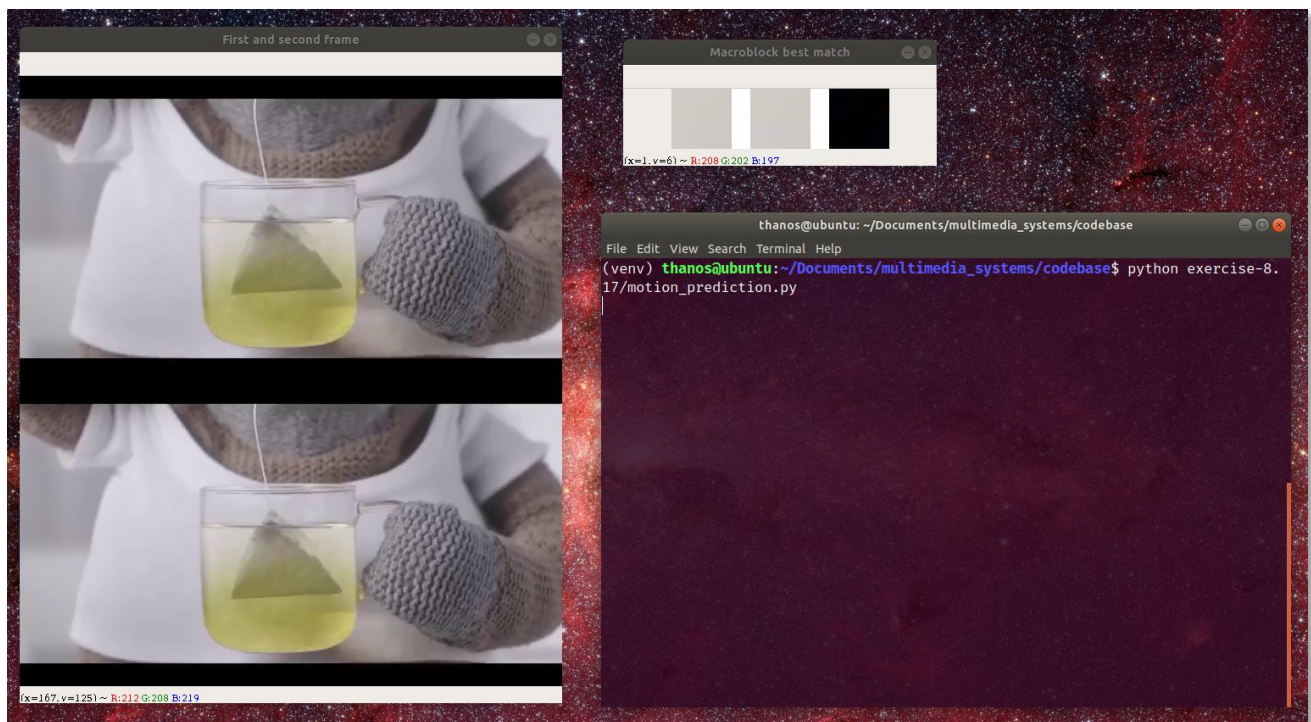
Μόλις προβληθούν όλα τα πλαίσια σφαλμάτων το πρόγραμμα τερματίζει.

Το δεύτερο αρχείο **motion_prediction.py** υλοποιεί μια τεχνική πρόβλεψης κίνησης με τη βοήθεια της μετρικής SAD και λογαριθμικής αναζήτησης.

Για να εκτελέσουμε τον κώδικα ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε την γραμμή εντολών και μεταβαίνουμε στο φάκελο **multimedia_systems/exercise-8.17**
2. **python motion_prediction.py**

Αμέσως εμφανίζονται δυο παράθυρα, το πρώτο δείχνει τα δυο συγκρινόμενα πλαίσια και



Εικόνα 5.2

το δεύτερο τα μακρομπλόκ που συγκρίνονται και τη διαφορά τους:

Μόλις ο αλγόριθμος συγκρίνει όλα τα μακρομπλόκ του δεύτερου πλαισίου με το καλύτερο μακρομπλόκ του πρώτου, το πρόγραμμα τερματίζει.

5.3. Επεξήγηση

Για την επίλυση της άσκησης χρησιμοποιούμε ένα διαφημιστικό βίντεο 5 δευτερολέπτων πάνω στο οποίο βασίζουμε όλη την επεξεργασία που γίνεται. Το αρχείο του βίντεο βρίσκεται στη τοποθεσία **videos/video1.mp4**. Στο πρώτο ερώτημα όσον αφορά τα πλαίσια σφαλμάτων, φορτώνουμε με τη βοήθεια της βιβλιοθήκης **opencv-python** όλα τα πλαίσια του βίντεο, αποθηκεύοντας πάντα το προηγούμενο (εκτός του πρώτου που είναι τύπου I). Η χρήσιμη συνάρτηση **cv2.absdiff** μας δίνει τη δυνατότητα να βρούμε τη διαφορά των δυο πλαισίων άμεσα και έπειτα γίνεται η προβολή τους.

Στο δεύτερο ερώτημα, παίρνουμε τα πρώτα δυο πλαίσια του βίντεο και τα χωρίζουμε σε μακρομπλόκ 16×16 . Για να διαμορφωθούν όλα τα επιθυμητά μακρομπλόκ είναι αναγκαίο να μεταβάλλουμε το πλάτος και ύψος του βίντεο με τέτοιο τρόπο ώστε να διαιρούνται με το 16. Μόλις τελειώσει αυτή η διαδικασία, εφαρμόζουμε τον αλγόριθμο λογαριθμικής αναζήτησης με $k = 16$ για κάθε μακρομπλόκ του πλαισίου-στόχου ώστε να βρούμε το κατάλληλο πλαίσιο-αναφοράς χρησιμοποιώντας τη μετρική SAD. Τέλος, προβάλλουμε τα δυο πλαίσια σε ένα παράθυρο και όλα τα πλαίσια στόχων, αναφοράς και σφαλμάτων που προέκυψαν.

6. Άσκηση 8.18

6.1. Εκφώνηση

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 ένα 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 αντικειμένων για διάφορα μεγέθη μακρομπλόκ.

- Κατ' αρχήν, φορτώστε ένα σύνολο πλαισίων βίντεο. Υποθέστε ότι το πρώτο πλαίσιο είναι αποκλειστικά πλαίσιο παρασκήνιου και δεν περιέχει αντικείμενα σε κίνηση. Δοθέντος ενός πλαισίου, n , προχωρήστε στη διαίρεσή του σε μπλοκ. Υπολογίστε ένα διάνυσμα κίνησης ανά μπλοκ με βάση το προηγούμενο πλαίσιο (αναφοράς). Για τα μπλοκ παρασκήνιου, θα πρέπει να προκύψουν μη μηδενικά διανύσματα κίνησης. Παρακολουθήστε όλα αυτά τα διανύσματα κίνησης. Μπορείτε ακόμη και να τα απεικονίσετε για κάθε μπλοκ, καθώς προβάλετε το βίντεο.
- Στη συνέχεια, βρείτε τα μπλοκ που αντιστοιχούν σε μη μηδενικά διανύσματα κίνησης. Αντικαταστήστε κάθε τέτοιο μπλοκ με το αντίστοιχο μπλοκ παρασκήνιου. Τα εικονοστοιχεία παρασκήνιου θα πρέπει να προέρχονται από προηγούμενο πλαίσιο, στο οποίο δεν υπήρχε αντικείμενο σε κίνηση. Η αντικατάσταση όλων αυτών των μπλόκ θα οδηγήσει στην απομάκρυνση των αντικειμένων, διότι τα

αντικείμενα αντικαθίστανται από το παρασκήνιο. Επαναλάβετε τη διαδικασία, χρησιμοποιώντας μακρομπλόκ διαφορετικού μεγέθους.

- Είναι πιθανό να αντιμετωπίσετε ασυνέχειες και παρενέργειες στα όρια των μπλοκ που αντικαθίστανται. Πως θα ελαχιστοποιήσετε αυτά τα φαινόμενα;

Η προτεινόμενη λύση θα λειτουργήσει ικανοποιητικά, μόνο όταν υποθέστε ότι η κάμερα δεν κινείται, όταν τα αντικείμενα που κινούνται υπόκεινται σε λεία κίνηση και ότι δεν παρατηρούνται αλλαγές κατεύθυνσης. Όμως στη γενική περίπτωση, η κάμερα κινείται, τα αντικείμενα κινούνται με όχι αυστηρό τρόπο και επιπλέον, παρατηρούνται αλλαγές στον φωτισμό, καθώς η κάμερα κινείται.

- Αλλά ας υποθέσουμε ότι μπορείτε να εντοπίσετε όλα τα μακρομπλόκ ενός πλαισίου που αντιστοιχούν στο παρασκήνιο. Πως μπορείτε να αξιοποιήσετε αυτό το γεγονός, πλέον της χρήσης αντιστάθμισης κίνησης σε επίπεδο μακρομπλόκ;

6.2. Εκτέλεση

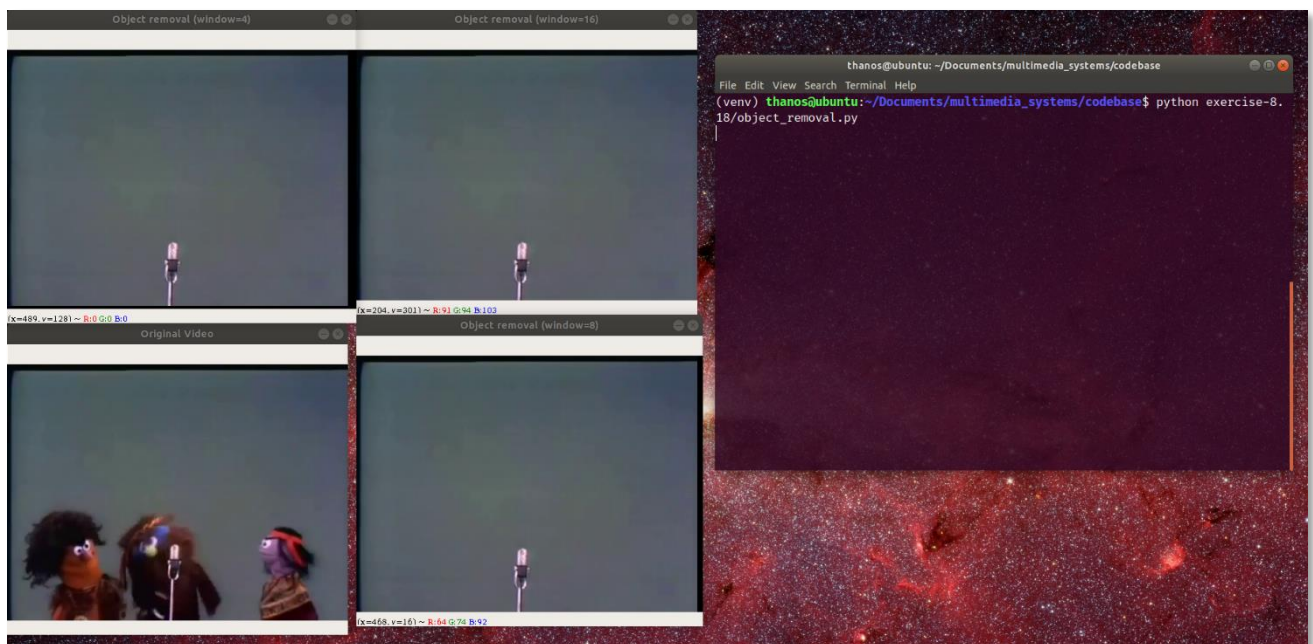
Η επίλυση της άσκησης βρίσκεται στον φάκελο **exercise-8.18** και συγκεκριμένα στο αρχείο **object_removal.py**

Για να εκτελέσουμε τον κώδικα ακολουθούμε τα παρακάτω βήματα:

1. Ανοίγουμε την γραμμή εντολών και μεταβαίνουμε στο φάκελο **multimedia_systems/exercise-8.18**
2. **python object_removal.py**

Παρατηρούμε αμέσως ότι εμφανίζονται τέσσερα παράθυρα:

- Το πρώτο παράθυρο απεικονίζει το αρχικό βίντεο με τους χαρακτήρες.
- Το δεύτερο παράθυρο απεικονίζει το βίντεο με τους εξαφανισμένους χαρακτήρες στο οποίο το μέγεθος μακρομπλόκ είναι ίσο με 16.
- Το τρίτο παράθυρο απεικονίζει το βίντεο με τους εξαφανισμένους χαρακτήρες στο οποίο το μέγεθος μακρομπλόκ είναι ίσο με 8.
- Το τέταρτο παράθυρο απεικονίζει το παράθυρο με τους εξαφανισμένους χαρακτήρες στο οποίο το μέγεθος μακρομπλόκ είναι ίσο με 4.



Εικόνα 6.1

Μόλις προβληθούν όλα τα πλαίσια του βίντεο το πρόγραμμα τερματίζει.

6.3. Επεξήγηση

Σε αυτή την άσκηση χρησιμοποιήσαμε ένα βίντεο με καρτούν 10 δευτερολέπτων. Στόχος μας είναι η απομάκρυνση των τριών χαρακτήρων που κινούνται στο παρασκήνιο με μακρομπλόκ από το πρώτο πλαίσιο του βίντεο, πριν εμφανιστούν. Επιλέξαμε το πρώτο πλαίσιο επειδή τα διανύσματα κίνησης είναι σχεδόν μηδενικά λόγω του ότι το παρασκήνιο είναι στατικό. Αφού χωρίσαμε το βίντεο σε μακρομπλόκ μεγέθους 16, 8 και 4, το κάθε επόμενο πλαίσιο δανείζεται τα μακρομπλόκ του προηγούμενου από τη μέση και κάτω. Εφόσον το δεύτερο πλαίσιο παίρνει τα μακρομπλόκ παρασκήνιου του πρώτου, το τρίτο του δεύτερου κ.ο.κ. έχουμε ως αποτέλεσμα την αφαίρεση των κινούμενων χαρακτήρων από το βίντεο.

Αξίζει να σημειωθεί ότι η τεχνική αυτή έχει κάποιες παρενέργειες στο βίντεο, ένας προσεκτικός παρατηρητής μπορεί να διαπιστώσει ότι από τη μέση και κάτω υπάρχει μια απότομη αλλαγή στα εικονοστοιχεία του παρασκήνιου. Επίσης, εάν το παρασκήνιο είχε κάποια επιπλέον στατικά χαρακτηριστικά, αυτό θα απαιτούσε μια πιο προσεκτική αντικατάσταση των μακρομπλόκ. Τέλος, σε ένα πραγματικό παράδειγμα, μπορεί να υπήρχαν αλλαγές στη φωτεινότητα πράγμα που δεν μας απασχολεί σε αυτή τη περίπτωση.

Για να αντιμετωπίσουμε τις παρενέργειες στα όρια των μακρομπλόκ προτείνουμε τις εξής λύσεις:

- Έξυπνη αντικατάσταση από μακρομπλόκ του ίδιο και όχι προηγούμενου πλαισίου. Αυτό πρακτικά σημαίνει ότι εφόσον οι χαρακτήρες δεν καλύπτουν όλο το παρασκήνιο, μπορούμε να πάρουμε μέρους του από το ίδιο στιγμιότυπο.
- Συνδυασμός μακρομπλόκ από το ίδιο και προηγούμενα πλαίσια ώστε να λαμβάνεται υπ' όψη ο φωτισμός και άλλα χαρακτηριστικά του παρασκήνιου που μπορεί να μην είναι διακριτά σε ένα διάστημα πλαισίων της κίνησης των χαρακτήρων.

7. Βοηθητικές συναρτήσεις

Το αρχείο **functions.py** που βρίσκεται στους φακέλους **exercise-8.17** και **exercise-8.18** περιέχει χρήσιμες συναρτήσεις που χρησιμοποιούνται στον κώδικα των ασκήσεων. Ας δούμε λεπτομερώς πως λειτουργούν.

7.1. `frame_to_macroblocks(frame, window = 16)`

Η συνάρτηση δέχεται ένα οποιοδήποτε πλαίσιο (frame) και επιστρέφει έναν πίνακα που περιέχει όλα τα μακρομπλόκ. Η προεπιλεγμένη τιμή των διαστάσεών τους είναι 16 (window).

Βήματα για την δημιουργία των μακρομπλόκ:

- Μεταβολή του πλάτους και ύψους του εκάστοτε πλαισίου ώστε να μπορέσουμε να διαιρέσουμε την εικόνα σε 16×16 μακρομπλόκ (προεπιλογή).
- Εάν το πλαίσιο που προκύψει έχει μεγαλύτερη διάσταση από το αρχικό, συμπληρώνεται με μαύρα εικονοστοιχεία.
- Διατρέχουμε όλες τις γραμμές και στήλες του νέου πλαισίου με βήμα ανάλογο με το μέγεθος του μακρομπλόκ.
- Αποθηκεύουμε το μακρομπλόκ στην εκάστοτε γραμμή και στήλη του πίνακα αποτελεσμάτων.

Τέλος επιστρέφουμε τον πίνακα αποτελεσμάτων της μορφής: (**γραμμή, στήλη, μακρομπλόκ, 16, 16, 3**).

7.2. `macroblocks_to_frame(macroblocks)`

Η συνάρτηση δέχεται ένα αποτέλεσμα της συνάρτησης `frame_to_macroblocks` και επιστρέφει τον ανακατασκευασμένο πλαίσιο.

Διατρέχουμε τον πίνακα αποτελεσμάτων των μακρομπλόκ ανά γραμμή και στήλη και τοποθετούμε όλα τα pixel ξανά σε διαστάσεις πλαισίου.

Επιστρέφουμε το νέο πλαίσιο που μπορούμε έπειτα να το προβάλλουμε σαν εικόνα.

7.3. `fit_size(x, window = 16)`

Επιστρέφει έναν ακέραιο αριθμό y που διαιρείται από το μέγεθος του μακρομπλόκ (`window`) και είναι ο πλησιέστερος μεγαλύτερος αριθμός στον x .

7.4. `get_sad(m_prev, m_next, window = 16)`:

Η συνάρτηση δέχεται δυο μακρομπλόκ, ενός πλαισίου αναφοράς (`m_prev`) και ενός πλαισίου στόχου (`m_next`) αντίστοιχα. Επιστρέφει την μετρική τιμή SAD για αυτά τα δυο μακρομπλόκ. Το προεπιλεγμένο μέγεθος των μακρομπλόκ είναι 16 (`window`).

Διατρέχουμε τους δυο πίνακες μακρομπλόκ ανά εικονοστοιχείο και προσθέτουμε στο σκορ την απόλυτη διαφορά ανά χρωματική συνιστώσα.

Το αποτέλεσμα που επιστρέφουμε είναι το άθροισμα των απόλυτων διαφορών από όλα τα εικονοστοιχεία.

7.5. `get_best_match(ms_prev, next_row, next_col, m_next, k = 16)`

Εφαρμόζει λογαριθμική αναζήτηση για ένα μακρομπλόκ (`m_next`) που βρίσκεται στη θέση (`next_row`, `next_col`) με βάση τα μακρομπλόκ του πλαισίου αναφοράς (`ms_prev`).

Τα βήματα αντιστοίχισης μακρομπλόκ:

- Συλλέγουμε τις 8 γειτονικές θέσεις γύρω από το μακρομπλόκ του πλαισίου στόχου εφόσον υπάρχουν.
- Υπολογίζουμε τις μετρικές SAD για τα 8 γειτονικά μακρομπλόκ.
- Επαναπροσδιορίζουμε το κέντρο αναζήτησης στο γειτονικό μακρομπλόκ με το μικρότερο σκορ SAD.
- Επαναλαμβάνουμε τη διαδικασία με $k = 16$ (προεπιλογή) μειώνοντας κάθε φορά το k στο μισό, μέχρι το k να γίνει 1.

Τέλος, επιστρέφουμε το μακρομπλόκ που επιλέχθηκε μετά από αλληπάλληλες αλλαγές του κέντρου αναζήτησης.

8. Αναφορές - Βιβλιογραφία

Sesame Street - Mad. (n.d.). Retrieved from YouTube:

<https://www.youtube.com/watch?v=D6oINRXLt-8>

Tips, P. (n.d.). *PG tips Green Tea*. Retrieved from YouTube:

https://www.youtube.com/watch?v=mEh5GiQ4_Yk

Συστήματα Πολυμέσων - Αλγόριθμοι, Πρότυπα και Εφαρμογές

Parag Havaladar, Gerard Medioni -

Επιμέλεια Ελληνικής έκδοσης Άγγελος Πικράκης, Ph.D

Σημείωσης μαθήματος "Συστήματα Πολυμέσων"

Διδάσκον Άγγελος Πικράκης, Ph.D