Συστήματα Πολυμέσων και Εικονική Πραγματικότητα

Κωδικοποιητής/αποκωδικοποιητής MPEG-1

Ομάδα Κατανόησης Πολυμέσων
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Εργασία 2017-2018

Εισαγωγικές Παρατηρήσεις

Η παρακάτω εργασία αποτελεί *προαιρετικό* μέρος του μαθήματος 987Η-Συστήματα Πολυμέσων και Εικονική Πραγματικότητα και η εκτέλεσή της συνεισφέρει 2, 4 ή και 5 επιπλέον μονάδες στην τελική βαθμολογία. Η εργασία θα πρέπει να εκτελεστεί σε ομάδες μέχρι και δύο ατόμων και αποτελείται από 3 ενότητες. Η υλοποίηση της εργασίας μπορεί κατ' επιλογή να περιλάβει την πρώτη ενότητα, την πρώτη και τη δεύτερη, κ.ο.κ. συνεισφέροντας τις αντίστοιχες μονάδες για κάθε ενότητα. Δεν μπορεί όμως να εκτελεστεί μία ενότητα χωρίς να έχει ορθά εκτελεστεί η προηγούμενή της.

Η προτεινόμενη εργασία στοχεύει στην υλοποίηση ενός κωδικοποιητή/ αποκωδικοποιητή video κατά το πρότυπο MPEG-1 Video (ISO/IEC 11172-2). Η εργασία θα περιλαμβάνει τις συναρτήσεις που περιγράφονται στη συνέχεια, υλοποιημένες σε MATLAB. Θα συνοδεύεται υποχρεωτικά από γραπτή αναφορά η οποία θα περιγράφει τον τρόπο χρήσης των προγραμμάτων και θα επιδεικνύει ενδεικτικά αποτελέσματα. Για τον έλεγχο του κωδικοποιητή/αποκωδικοποιητή που θα κατασκευαστεί θα χρησιμοποιηθεί ενδεικτική ακολουθία ασυμπίεστων εικόνων σε μορφή RGB-CCIR.

1 MPEG Library (2 μονάδες)

Σκοπός του πρώτου παραδοτέου είναι η δημιουργία μιας βιβλιοθήκης συναρτήσεων που θα χρησιμοποιηθούν αργότερα για την κατασκευή του κωδικοποιητή. Κάθε συνάρτηση της βιβλιοθήκης συνοδεύεται και από την αντίστροφή της που ανακατασκευάζει την είσοδο της πρώτης και που θα αποτελέσει μέρος του αποκωδικοποιητή.

1.1 Προεπεξεργασία

Η ακολουθία των εικόνων εισόδου περνά αρχικά από ένα στάδιο προεπεξεργασίας με σκοπό τη μετατροπή της στη μορφή του MPEG SIF.

Κατασκευάστε τη συνάρτηση

```
function [frameY, frameCr, FrameCb] = ccir2ycrcb(frameRGB)
```

και την αντίστροφή της

```
function [frameRGB] = ycrcb2ccir(frameY, frameCr, FrameCb)
```

όπου

`frameY`: Το Y κανάλι του frame, με διάσταση 352×288.

`frameCr`: Το Cr κανάλι του frame, με διάσταση 176×144.

`frameCb`: Το Cb κανάλι του frame, με διάσταση 176×144.

`frameRGB`: Το frame προς επεξεργασία [1].

Αποτέλεσμα της `ccir2ycrcb` είναι η δημιουργία μιας ακολουθίας εικόνων με ανάλυση 352×288, σε $YCrCb$ $4:2:0$. Πληροφορίες για τη διαδικασία αυτή υπάρχουν στην ενότητα `2-D.3.1` του προτύπου, όπου δίνονται και ενδεικτικά FIR φίλτρα.

Αντίστοιχα, η `ycrcb2ccir` ανακατασκευάζει την είσοδο σε μορφή RGB CCIR601 με ανάλυση 720×576 pixels. Η διαδικασία περιγράφεται στην ενότητα `2-D.8.2` του προτύπου.

Σημείωση: Οι εικόνες που σας δίνονται είναι RGB και θεωρήστε ότι παράγονται 25 τέτοιες εικόνες ανά δευτερόλεπτο. Θα πρέπει να μετατρέψετε πρώτα τις εικόνες που σας δίνονται σε μορφή CCIR 601 (η οποία είναι τύπου $4:2:2$) πριν εφαρμόσετε την διαδικασία που περιγράφεται στην ενότητα `2-D.3.1`. Κατά την μετατροπή θα πρέπει να σιγουρευτείτε ότι δεν θα προκύψει aliasing.

1.2 Εκτίμηση και Αντιστάθμιση Κίνησης

Για την εκτίμηση κίνησης στα P-frames, κατασκευάστε τη συνάρτηση

```
function [eMBY, eMBCr, eMBCb, mV] = motEstP(frameY, frameCr, frameCb, mBIndex, ...
                                            refFrameY, refFrameCr, refFrameCb)
```

και την αντίστροφή της

```
function [mBY, mBCr, mBCb] = iMotEstP(eMBY, eMBCr, eMBCb, mBIndex, mV, ...
                                      refFrameY, refFrameCr, refFrameCb)
```

όπου

`eMBY`: Το Y κανάλι του macroblock σφάλματος πρόβλεψης, με διαστάσεις 16×16.

`eMBCr`: Το Cr του macroblock σφάλματος πρόβλεψης, με διαστάσεις 8×8.

`eMBCb`: Το Cb κανάλι του macroblock σφάλματος πρόβλεψης, με διαστάσεις 8×8.

`mV`: Ο πίνακας με τα motion vectors του συγκεκριμένο macroblock της εικόνας. Είναι ένας πίνακας με δύο γραμμές και δύο στήλες που περιέχει στην πρώτη στήλη το motion vector που αφορά την εκτίμηση κίνησης σε σχέση με το προηγούμενο frame και στη δεύτερη το motion vector σε σχέση με το επόμενο. Εφόσον στα P-frames δε χρησιμοποιείται επόμενο frame ως αναφορά, η δεύτερη στήλη περιέχει το στοιχείο `NaN` (not a number) του MATLAB.

`frameY`: Το Y κανάλι του τρέχοντος P-frame

`frameCr`: Το Cr κανάλι του τρέχοντος P-frame

`frameCb`: Το Cb κανάλι του τρέχοντος P frame

[1] Η ενδεικτική ακολουθία που σας δόθηκε έχει διαστάσεις 720×576 και είναι σε RGB

`mBIndex`: Ο αύξων αριθμός του macroblock στην εικόνα. Θεωρείστε ότι τα macroblocks είναι διατεταγμένα με αύξοντα αριθμό από αριστερά προς τα δεξιά και από πάνω προς τα κάτω. Το πάνω αριστερά macroblock θεωρούμε ότι έχει αύξοντα αριθμό 0.

`refFrameY`: Το Y κανάλι του frame αναφοράς
`refFrameCr`: Το Cr κανάλι του frame αναφοράς
`refFrameCb`: Το Cb κανάλι του frame αναφοράς
`mBY`: Το Y κανάλι του ανακατασκευασμένου macroblock, με διάσταση 16×16.
`mBCr`: Το Cr κανάλι του ανακατασκευασμένου macroblock, με διάσταση 8×8.
`mBCb`: Το Cb κανάλι του ανακατασκευασμένου macroblock, με διάσταση 8×8.

Να κατασκευαστεί η αντίστοιχη συνάρτηση για τα B-frames

```
function [eMBY, eMBCr, eMBCb, mV] = motEstB(frameY, frameCr, frameCb, mBIndex, ...
                                            backwFrameY, backwFrameCr, backwFrameCb, ...
                                            forwFrameY, forwFrameCr, forwFrameCb)
```

και η αντίστροφή της

```
function [mBY, mBCr, mBCb] = iMotEstB(eMBY, eMBCr, eMBCb, mBIndex, mV, ...
                                      backwFrameY, backwFrameCr, backwFrameCb, ...
                                      forwFrameY, forwFrameCr, forwFrameCb)
```

όπου

`eMBY`: Το Y κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 16×16.
`eMBCr`: Το Cr κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 8×8.
`eMBCb`: Το Cb κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 8×8.
`mV`: Ο πίνακας με τα motion vectors. Η δομή του είναι ίδια με την αντίστοιχη της προηγούμενης συνάρτησης.
`frameY`: Το Y κανάλι του τρέχοντος P-frame.
`frameCr`: Το Cr κανάλι του τρέχοντος P-frame.
`frameCb`: Το Cb κανάλι του τρέχοντος P-frame.
`mBIndex`: Ο αύξων αριθμός του macroblock στην εικόνα.
`backwFrameY`: Το Y κανάλι του προηγούμενου frame αναφοράς.
`backwFrameCr`: Το Cr κανάλι του προηγούμενου frame αναφοράς.
`backwFrameCb`: Το Cb κανάλι του προηγούμενου frame αναφοράς.
`forwFrameY`: Το Y κανάλι του επόμενου frame αναφοράς.
`forwrameCr`: Το Cr κανάλι του επόμενου frame αναφοράς.
`forwFrameY`: Το Cb κανάλι του επόμενου frame αναφοράς.
`mBY`: Το Y κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 16×16.
`mBCr`: Το Cr κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 8×8.
`mBCb`: Το Cb κανάλι του macroblock σφάλματος πρόβλεψης, με διάσταση 8×8.

1.3 Μετασχηματισμός DCT

Κατασκευάστε τη συνάρτηση

```
function [dctBlock] = blockDCT(block)
```

και την αντίστροφή της

```
function [block] = iBlockDCT(dctBlock)
```

όπου

`dctBlock`: Τα DCT coefficients του block 8×8.
`block`: Το block της εισόδου.

Για την υλοποίηση αυτού του βήματος συνίσταται η χρήση της συνάρτησης `dct2` της MATLAB. Υπενθυμίζεται ότι ο μετασχηματισμός εφαρμόζεται σε επίπεδο block 8×8. Η διαδικασία περιγράφεται στήν ενότητα `2-D.6.3.3`.

1.4 Κβαντισμός

Για τον κβαντισμό των I και P και B-blocks αντίστοιχα, κατασκευάστε τις συναρτήσεις

```
function [qBlock] = quantizeI(dctBlock, qTable, qScale)
function [qBlock] = quantizePB(dctBlock, qTable, qScale)
```

και τις αντίστροφές τους

```
function [dctBlock] = dequantizeI(qBlock, qTable, qScale)
function [dctBlock] = dequantizePB(qBlock, qTable, qScale)
```

όπου

`qBlock`: Τα σύμβολα κβαντισμού των DCT coefficients του block.
`dctBlock`: Τα DCT coefficients του block.
`qTable`: Ο πίνακας κβαντισμού (ενότητα `2-D.6.3.4`).
`qScale`: Η κλίμακα κβαντισμού (ενότητα `2-D.6.4.5`).

Εδώ χρησιμοποιούνται ξεχωριστές συναρτήσεις διότι ο κβαντιστής στην περίπτωση των I-blocks κβαντίζει με διαφορετικό (και ανεξάρτητο από την κλίμακα κβάντισης) τρόπο τα DC coefficients, σε αντίθεση με τον κβαντιστή των P και B-blocks.

Υπενθυμίζεται ότι κατά τον κβαντισμό, το αποτέλεσμα που προκύπτει από το ακέραιο μέρος της διαίρεσης με το βήμα κβαντισμού είναι σύμβολο και όχι νούμερο (τιμή φωτεινότητας ή χρωματικότητας).

1.5 Zig-zag scanning και RLE

Κατασκευάστε τη συνάρτηση υπολογισμού των συμβόλων μήκους διαδρομής για τους κβαντισμένους συντελεστές DCT

```
function [runSymbols] = runLength(qBlock)
```

και την αντίστροφή της

```
function [qBlock] = iRunLength(runSymbols)
```

όπου

`runSymbols`: Ο πίνακας που περιέχει τα σύμβολα μήκους διαδρομής τα οποία είναι δυάδες της μορφής `(precedingZeros, quantSymbol)`. Ο πίνακας έχει συνεπώς διαστάσεις $R \times 2$, όπου R τα μήκη διαδρομής που εντοπίστηκαν στο συγκεκριμένο block.
`qBlock`: Τα σύμβολα κβαντισμού των DCT coefficients.

Τα σύμβολα δεν κωδικοποιούνται με τη σειρά 'γραμμή-στήλη', αλλά χρησιμοποιώντας fzig-zag scanning. Σύμφωνα με το πρότυπο, η διαδικασία κωδικοποίησης RLE ακολουθείται μόνο για τους AC συντελεστές κάθε block. Εδώ όμως, για λόγους απλούστευσης, ο DC συντελεστής κάθε block θα περιληφθεί, έτσι ώστε το πρώτο run να έχει τη μορφή `[0, DC_value]`. Η διαδικασία περιγράφεται λεπτομερώς στις ενότητες `2-D.6.3.4` και `2-D.6.4.6`.

1.6 Κωδικοποίηση Μεταβλητού Μήκους Λέξης (Variable Length Coding)

Κατασκευάστε τη συνάρτηση

```
function [vlcStream] = vlc(runSymbols)
```

και την αντίστροφή της

```
function [runSymbols] = ivlc(vlcStream)
```

όπου

`vlcStream`: Πίνακας από bits, κωδικοποιημένος με λέξεις μεταβλητού μήκους.
`runSymbols`: Τα σύμβολα μήκους διαδρομής.

Η συνάρτηση αυτή πρέπει να κωδικοποιεί τα σύμβολα μήκους διαδρομής σύμφωνα με τους κώδικες που περιλαμβάνονται στους πίνακες `2-D.15`, `2-D.16` και `2-D.17` του προτύπου. Η πληροφορία αυτών των πινάκων θα είναι

αποθηκευμένη σαν `global` μεταβλητή. Χρησιμοποιήστε τη συνάρτηση `getTheGlobals` που ορίζει τους παραπάνω πίνακες `d15a, d15b, d16a, d16b, d17a, d17b` ως `global` μεταβλητές. Η ίδια συνάρτηση ορίζει τους πίνακες `h, h16, h17` που δημιουργήθηκαν με τη συνάρτηση `vlcTable` και, εφόσον το επιθυμείτε μπορούν να χρησιμοποιηθούν για την αποκωδικοποίηση των vlc codes. Τόσο η `vlcTable`, όσο και οι `h, h16, h17` παρέχονται, αλλά χωρίς υποστήριξη.

2 MPEG Processing (+2 μονάδες)

Σε αυτό το βήμα θα χρησιμοποιηθούν οι προηγούμενες συναρτήσεις για την κατασκευή ενός codec, Ο οποίος θα είναι δομημένος σε επίπεδα, όπως αυτά ορίζονται από το πρότυπο. Συγκεκριμένα, υπάρχουν τα επίπεδα κωδικοποίησης Sequence, Group of Pictures, Picture, Slice, MacroBlock και Block, τα οποία θα χειρίζονται οι αντίστοιχες συναρτήσεις.

2.1 Δομή Αποθήκευσης

Το αποτέλεσμα της διαδικασίας encodeMPEG θα αποθηκευθεί σε δομή τύπου struct του matlab. Αυτή παρουσιάζεται στον ακόλουθο πίνακα.

Δομή	Στοιχείο	Σχόλια
SeqEntity	SeqHeader (1) GoPEntityArray (1,*) SeqEnd (1)	Ο Header της ακολουθίας σε δυαδική μορφή Πίνακας από δομές τύπου GoPEntity
GoPEntity	GoPHeader (1) PicSliceEntityArray(GoP size)	Πίνακας από δομές τύπου PicSliceEntity
PicSliceEntity	PicSliceHeader (1) MBEntityArray (n)	Πίνακας από δομές τύπου MBEntity
MBEntity	MBHeader (1) MotionVetors (1) BlockEntityArray (6)	Τα motion vectors του macroblock. Πίνακας 2×2 Δομή τύπου BlockEntity. Τα πρώτα 4 αντιστοιχούν στο Y macroblock, και τα επόμενα 2 στα Cb και Cr blocks.
BlockEntity	VLCodes	Τα κωδικοποιημένα blocks σε δυαδική μορφή
SeqHeader	sequence_header_code horizontal_size vertical_size	32 bslbf 12 uimsbf 12 uimsbf
SeqEnd	sequence_end_code	32 bslbf
GoPHeader	group_start_code	32 bslbf
PicSliceHeader	picture_start_code temporal_reference picture_coding_type slice_start_code quantizer_scale	32 bslbf 10 uimsbf 3 uimsbf (1, 2, 3 για I, P, B frames αντίστοιχα) 32 bslbf 5 uimsbf
MBHeader	macroblock_type quantizer_scale	3 uimsbf (1, 2, 3 για I, P, B frames αντίστοιχα) 5 uimsbf
MotionVetors	Πίνακας 2×2 από motion vectors	για την υλοποίηση του επιπέδου 2, τα motion vectors θα αναπαρασταθούν όπως προκύπτουν από τις εξόδους των συναρτήσεων motestP και motestB, σαν πίνακες που περιέχουν τις υπολογισμένες αλγεβρικές τιμές ή NaN. Στην υλοποίηση του επιπέδου 3 τα motion vectors κάθε PicSlice θα κωδικοποιηθούν με DPCM και θα χρησιμοποιηθούν οι vlc κώδικες της ενότητας 2-D.6.2.3 για την αναπαράστασή τους.

Να σημειωθεί ότι όπου αναφέρεται δυαδική μορφή υπονοείται ένας πίνακας από char ο οποίος περιέχει στοιχεία

'0' και '1'. Η συνάρτηση treelist, η οποία διατείθεται, τυπώνει τα στοιχεία μιας δομής σε μορφή δέντρου και μπορείτε να τη χρησιμοποιήσετε για έλεγχο της κατασκευασμένης δομής.

2.2 Ορισμοί Συναρτήσεων και Δομών Δεδομένων

2.2.1 Κυρίες Συναρτήσεις

```
function encodeMPEG(bName, fExtension, startFrame, numOfGoPs, qScale)
```

bName: Το πρόθεμα της σειράς των αρχείων εικόνων.
fExtension: Η επέκταση της σειράς των αρχείων εικόνων.
startFrame: Ο αύξων αριθμός της πρώτης εικόνας.
GoP: Η μορφή του GoP. Π.χ. GoP='IBBPBBP'.
numOfGoPs: Ο αριθμός των GoPs.
qScale: Η κλίμακα κβάντισης (βλ. ενότητα 2-D.6.4.5 του προτύπου).

Παρατηρήσεις/απλουστεύσεις:
1. Οι τιμές των GoP και qScale θα παραμένουν σταθερές σε όλη την κωδικοποίηση.
2. Τα GoP θεωρούνται κλειστά (δηλ. αρχίζουν με I και τελειώνουν σε P ή I).

```
function GoPEntityArray = encodeSeq(bName, fExtension, startFrame, GoP, numOfGoPs, qScale)
```

bName: Το πρόθεμα της σειράς των αρχείων εικόνων.
fExtension: Η επέκταση της σειράς των αρχείων εικόνων.
startFrame: Ο αύξων αριθμός της πρώτης εικόνας.
GoP: Η μορφή του GoP.
numOfGoPs: Ο αριθμός των GoPs.
qScale: Η κλίμακα κβάντισης.
GoPEntityArray: Πίνακας από δομές τύπου GoPEntity.

```
function [t, PicSliceEntityArray] = encodeGoP(frameNumber, bName, fExtension, ...
                                               startFrame, GoP, qScale)
```

frameNUmber: Ο αύξων αριθμός του πρώτου frame του GoP.
bName: Το πρόθεμα της σειράς των αρχείων εικόνων.
fExtension: Η επέκταση της σειράς των αρχείων εικόνων.
startFrame: Ο αύξων αριθμός της πρώτης εικόνας.
GoP: Η μορφή του GoP.
qScale: Η κλίμακα κβάντισης.
t: Ακέραιος αριθμός που συμβολίζει επιτυχία (1) ή αποτυχία (0) στην ανάγνωση εικόνας.
PicSliceEntityArray: Πίνακας από δομές τύπου PicSliceEntity.

Παρατηρήσεις/απλουστεύσεις:
1. Εδώ να περιλάβετε την κλήση της συνάρτησης getNextPicture που θα φορτώνει την επόμενη εικόνα προς κωδικοποίηση, σύμφωνα με τη μορφή του GoP.

```
function MBEntityArray = encodePicSlice(pic, picType, qScale)
```

pic: Η προς κωδικοποίηση εικόνα, όπως την επιστρέφει η getNextPicture. Δομή της MATLAB που περιέχει 3 στοιχεία (decPic.frameY, decPic.frameCr, decPic.frameCb) που αντιστοιχούν στις συνιστώσες της εικόνας. Καθεμία από αυτές είναι ένας διδιάστατος αριθμητικός πίνακας με διαστάσεις 352×288, 176×144 και 176×144 αντίστοιχα.
picType: Ο τύπος της εικόνας 'I', 'P', ή 'B'.
qScale: Η κλίμακα κβάντισης.
MBEntityArray: Πίνακας από δομές τύπου MBEntity.

```
function [MotionVectors, BlockEntityArray] = encodeMB(pic, picType, qScale, mBIndex)
```

`pic`: Η εικόνα προς κωδικοποιήση, όπως περιγράφεται στην `encodePicSlice`.
`picType`: Ο τύπος της εικόνας `'I'`, `'P'`, ή `'B'`.
`qScale`: Η κλίμακα κβάντισης.
`mBIndex`: Ο αύξων αριθμός του τρέχοντος macroblock.
`BlockEntityArray`: Πίνακας από δομές τύπου `BlockEntity`.
`MotionVectors`: Πίνακας με τα motion vectors, όπως περιγράφεται στην ενότητα 2 για τις συναρτήσεις `motEstP` και `motEstB`.

```
function VLCodes = encodeBlock(blockMatrix, mBType, qScale)
```

`blockMatrix`: Πίνακας με αριθμητικές τιμές του block.
`mBType`: Ο τύπος του macroblock.
`qScale`: Η κλίμακα κβάντισης.
`VLCodes`: Δυαδική συμβολοσειρά με την ακολουθία του κωδικοποιημένου block.

Παρατηρήσεις/απλουστεύσεις:
1. Σε αυτή τη συνάρτηση να ορίσετε και τους πίνακες κβαντισμού για τις συναρτήσεις `quantizeI` και `quantizePB`, χρησιμοποιώντας τις default τιμές του προτύπου.

```
function decodeMPEG(fName, outFName)
```

`fName`: Το όνομα του αρχείου τύπου `mat` όπου είναι αποθηκευμένη η δομή του βίντεο.
`outFName`: Η βάση του ονόματος με το οποίο θα αποθηκευτούν οι αποκωδικοποιημένες εικόνες. Το όνομα θα έχει τη μορφή `[base_name, frame_number, '.tiff']`.

```
function decodeSeq(GoPEntityArray, outFName)
```

`outFName`: Το πρόθεμα του ονόματος των αποκωδικοποιημένων εικόνων.
`GoPEntityArray`: Πίνακας από δομές τύπου `GoPEntity`.

Παρατηρήσεις/απλουστεύσεις:
1. Εδώ φορτώνεται η αποθηκευμένη δομή του κωδικοποιημένου βίντεο.

```
function decodeGoP(PicSliceEntityArray, outFName)
```

`outFName`: Το πρόθεμα του ονόματος των αποκωδικοποιημένων εικόνων.
`PicSliceEntityArray`: Πίνακας από δομές τύπου `PicSliceEntity`.

Παρατηρήσεις/απλουστεύσεις:
1. Εδώ αποθηκεύεται η αποκωδικοποιημένη εικόνα που επιστρέφει η `decodePicSlice` (αφού πρώτα κληθεί η `ycrcb2ccir` για να τη μετατρέψει στην κατάλληλη μορφή.
2. Για το σκοπό αυτό καλείται η βοηθητική συνάρτηση `pushFlushPic`, η οποία περιγράφεται παρακάτω.

```
function decPic = decodePicSlice(MBEntityArray)
```

`MBEntityArray`: Πίνακας από δομές τύπου `MBEntity`.
`decPic`: Η αποκωδικοποιημένη εικόνα. Δομή, όπως περιγράφεται στην `encodePicSlice`.

```
function [decMBY, decMBCr, decMBCb] = decodeMB(BlockEntityArray, MotionVectors, ...
                                               mBType, mBIndex, qScale)
```

`BlockEntityArray`: Πίνακας από δομές τύπου `BlockEntity`.
`MotionVectors`: Πίνακας με τα motion vectors.
`mBType`: Ο τύπος του macroblock.
`mBIndex`: Ο αύξων αριθμός του τρέχοντος macroblock.
`qScale`: Η κλίμακα κβάντισης.
`decMBY`, `decMBCr`, `decMBCb`: Τα Y, Cr, Cb κανάλια του αποκωδικοποιημένου macroblock. Είναι αριθμητικοί πίνακες με διαστάσεις 16×16, 8×8 και 8×8 αντίστοιχα.

```
function decBlock = decodeBlock(encBlock, mBType, qScale)
```

`BlockEntity`: Το κωδικοποιημένο block.
`mBIndex`: Ο τύπος του macroblock.
`qScale`: Η κλίμακα κβάντισης.
`decBlock`: Το αποκωδικοποιημένο block. Αριθμητικός πίνακας με 8×8 στοιχεία.

2.2.2 Βοηθητικές Συναρτήσεις

```
function pushFlushPic(pic, tempRef, outFName)
```

`pic`: Η εικόνα προς αποθήκευση. Δομή με στοιχεία `frameY, frameCr, frameCb`, όπως περιγράφεται στη συνάρτηση `decodePicSlice` για την `decPic`.

`tempRef`: Ο αύξων αριθμός της εικόνας. Αναφέρεται στη σειρά προβολής της και όχι στη σειρά κωδικοποίησης της εικόνας. Ο αριθμός αυτός περιέχεται στο `PicSliceEntityHeader.temporal_reference`.

`outFName`: Η βάση του ονόματος του αρχείου προς αποθήκευση.

Παρατηρήσεις:
1. Αυτή η συνάρτηση χρησιμοποιείται από την `decodeGoP` για να αποθηκεύσει την αποκωδικοποιημένη εικονοσειρά σαν αρχείο εικόνας στο δίσκο.

```
function pushPic(pic)
```

`pic`: Η εικόνα προς αποθήκευση στον buffer.

Παρατηρήσεις:
1. Αυτή η συνάρτηση χρησιμοποιείται για την αποθήκευση ενός καρέ στον buffer. Ο τελευταίος μπορεί να θεωρηθεί σαν μια `global` μεταβλητή η οποία περιέχει δύο εικόνες αποθηκευμένες σε δομές, όπως περιγράφεται στη συνάρτηση `encodePicSlice`.

```
function [pic, picType, tempRef] = getNextPicture(bName, fExtension, frameNumber, ...
                                                  startFrame, GoP)
```

`bName`: Το πρόθεμα της σειράς των αρχείων εικόνων.
`fExtension`: Η επέκταση της σειράς των αρχείων εικόνων.
`startFrame`: Ο αύξων αριθμός της πρώτης εικόνας.
`frameNumber`: Ο αύξων αριθμός του τρέχοντος frame.
`GoP`: Η μορφή του GoP.
`pic`: Η εικόνα που διαβάστηκε, αποθηκευμένη σαν δομή.
`picType`: Ο τύπος της εικόνας.
`tempRef`: Ο αύξων αριθμός της εικόνας `pic`.

2.3 Γενικές Παρατηρήσεις

- Οι συναρτήσεις `genSeqHeader`, `genGoPHeader`, `genPicSliceHeader`, `genMBHeader` και `genBHeader` δημιουργούν τα `header structures` ενώ οι συναρτήσεις `writeSeqHeader`, `writeGoPHeader`, `writePicSliceHeader`, `writeMBHeader` και `writeBHeader` τα τοποθετούν σε ένα `structure`. Στο επόμενο βήμα θα χρησιμοποιηθούν εμπλουτισμένες εκδόσεις των τελευταίων για να αποθηκευτούν τα αποτελέσματα σε ένα δυαδικό `stream`.
- Τα motion vectors που υπολογίζονται από τις συναρτήσεις `motestP` και `motestB` θα βρίσκονται στο διάστημα $[-7, 7]$ και θα κωδικοποιούνται σύμφωνα με τον κώδικα του πίνακα `2-D.9` της ενότητας `2-D.6.3` του προτύπου.
- Για λόγους απλότητας θεωρούμε ότι κάθε εικόνα περιέχει μόνο ένα Slice και η κωδικοποίηση στο επίπεδο αυτό γίνεται από κοινή συνάρτηση (`encodePicSlice`).
- Η υλοποίησή σας θα πρέπει να δουλεύει για οποιδήποτε GoP. Στα πλαίσια της εργασίας, θα χρησιμοποιήσετε τα IPP και $IBPBBP$.

- Μία καλή τιμή για την `qScale` είναι 8.
- Θα σας δοθούν συναρτήσεις (`pushstream( )` και `readstream( )`) που γράφουν και διαβάζουν δυαδικά δεδομένα σε streams.

3 MPEG Syntax (+1 μονάδα)

Στο σημείο αυτό η δομή που κατασκευάζεται στο προηγούμενο βήμα μετατρέπεται σε bitstream το οποίο περιλαμβάνει το κωδικοποιημένο video αλλά και τους κατάλληλους headers σε δυαδική μορφή. Το αποτέλεσμα της κωδικοποίησης θα είναι πίνακας από unsigned integers (`uint8`) που περιέχουν τα αντίστοιχα bits ανά οκτάδες σε δεκαδική μορφή. Για παράδειγμα το δυαδικό stream `'0110000001000001'` αντιστοιχεί σε πίνακα με 2 στοιχεία $[96, 65]$. Για την αποθήκευση και την ανάγνωση bits από τον πίνακα αυτον, μπορείτε να χρησιμοποιήσετε τις συναρτήσεις `pushstream` και `readstream`, που διατείθενται χωρίς υποστήριξη.

Πιο συγκεκριμένα, κατά την κωδικοποίηση αποθηκεύουμε τον πίνακα που μόλις περιγράφηκε σε μια `global` μεταβλητή με όνομα `encodedStream`. Θεωρούμε μια ακόμα `global` μεταβλητή, την `pos`, που εκφράζει το σημείο (bit) του τρέχοντος στοιχείου (byte) του πίνακα, στο οποίο θα προστεθεί το επόμενο bit που θα προκύψει από την κωδικοποίηση. Περισσότερες πληροφορίες για τη χρήση των μεταβλητών αυτών θα βρείτε στα σχόλια των αρχείων `pushstream` και `readstream`.

Για την υλοποίηση του βήματος, πρέπει να τροποποιηθούν ελαφρώς οι προηγούμενες συναρτήσεις, έτσι ώστε να μην δέχονται στην είσοδό (και να επιστρέφουν στην έξοδο τις δομές της ενότητας 2.1), αλλά να επιδρούν κατευθείαν πάνω στις αντίστοιχες `global` μεταβλητές. Εδώ παρουσιάζονται οι καινούργιες εκδόσεις των συναρτήσεων που θα χρειαστεί να αλλάξουν. Στο όνομα κάθε συνάρτησης η οποία πρέπει να τροποποιηθεί προστίθεται ο χαρακτήρας '3'.

```
function encodeSeq3(bName, fExtension, startFrame, GoP, numOfGoPs, qScale)
function [t] = encodeGoP3(frameNumber, bName, fExtension, startFrame, GoP, qScale)
function encodePicSlice3(pic, picType, qScale)
function encodeMB3(pic, picType, qScale, mBIndex, prevMV)
function encodeBlock3(blockMatrix, mBType, qScale)
function decodeSeq3(outFName)
function decodeGoP3(outFName)
function decPic = decodePicSlice3()
function [decMBY, decMBCr, decMBCb] = decodeMB3(MotionVectors, mBType, mBIndex, prevMV, qScale)
function decBlock = decodeBlock3(mBType, qScale)
```

Στο επίπεδο αυτό πρέπει επίσης να προσαρμοστεί και η διαδικασία αναπαράστασης των motion vectors με την κατασκευή των συναρτήσεων που θα κωδικοποιούν/αποκωδικοποιούν τα motion vectors σε επίπεδο PicSlice. Υπενθυμίζεται ότι για απλούστευση θεωρούμε ότι σε κάθε P (B) frame όλα τα blocks είναι τύπου P (B). Επίσης, η encodeMB3 πρέπει να γράφει στο stream *πρώτα* τα motion vectors και στη συνέχεια τα σφάλματα πρόβλεψης.

```
function dpcmMV = mvDPCM(currMV, prevMV, picType)
function decMV = imvDPCM(dpcmMV, prevMV, picType)
```

όπου

`currMV`: Πίνακας με τα motion vectors, του τρέχοντος macroblock.

`prevMV`: Πίνακας με τα motion vectors, του προηγούμενου macroblock.

`picType`: Ο τύπος της εικόνας.

`dpcmMV`: Τα differential motion vectors.

Στη συνέχεια, τα differential MVs που θα βρίσκονται στο διάστημα $[-16, 16]$ πρέπει να κωδικοποιούνται σε bitsream με χρήση των vlc codes του πίνακα `2-D.9`. Αυτό γίνεται με τη συνάρτηση `mvVLC`, ενώ την αντίστροφη εργασία κάνει η `imvVLC`.

```
function vlcMV = mvVLC(dpcmMV)
function dpcmMV = imvVLC(vlcMV, picType)
```

όπου

`vlcMV`: Τα κωδικοποιημένα κατά VLC motion vectors του τρέχοντος macroblock. Είναι πίνακας από χαρακτήρες (`'0'` ή `'1'`), μεταβλητού μήκους.

`dpcmMV`: Πίνακας με τα κωδικοποιημένα κατά dpcm motion vectors του τρέχοντος macroblock.

`picType`: Ο τύπος της εικόνας.

Πίνακας 1: Διάρθρωση των αρχείων κώδικα

Φάκελος	**Αρχεία**
`step1`	`ccir2ycrcb.m`, `ycbcr2ccir.m`, `motEstP.m`, `iMotEstP.m`, `motEstB.m`, `iMotEstB.m`, `blockDCT.m`, `iBlockDCT.m`, `quantizeI.m`, `quantizePB.m`, `dequantizeI.m`, `dequantizePB.m`, `runLength.m`, `iRunLength.m`, `vlc.m`, `ivlc.m`, `demo1.m`
`step2`	`encodeMPEG.m`, `encodeSeq.m`, `encodeGoP.m`, `encodePicSlice.m`, `encodeMB.m`, `encodeBlock.m`, `decodeMPEG.m`, `decodeSeq.m`, `decodeGoP.m`, `decodePicSlice.m`, `decodeMB.m`, `decodeBlock.m`, `pushFlushPic.m`, `pushPic.m`, `getNextPicture.m`, `demo2.m`
`step3`	`encodeSeq3.m`, `encodeGoP3.m`, `encodePicSlice3.m`, `encodeMB3.m`, `encodeBlock3.m`, `decodeSeq3.m`, `decodeGoP3.m`, `decodePicSlice3.m`, `decodeMB3.m`, `decodeBlock3.m`, `mvDPCM.m`, `imvDPCM.m`, `mvVLC.m`, `imvVLC.m`, `demo3.m`

Αξιολόγηση & Παραδοτέα

Κατά την υποβολή της εργασίας θα πρέπει να παραδώσετε τα αρχεία του παραπάνω πίνακα, καθώς και μία αναφορά. Ανά ενότητα υπάρχει και ένα script με όνομα `demo1.m`, `demo2.m`, και `demo3.m` αντίστοιχα, το οποίο επιδεικνύει την λειτουργικότητα της ενότητας. Στην αναφορά θα πρέπει να παρουσιάσετε όποιες σχεδιαστικές επιλογές έχετε κάνει κατά την υλοποίηση του κωδικοποιητή και του αποκωδικοποιητή. Επίσης θα πρέπει να παρουσιάσετε ενδεικτικά αποτελέσματα της λειτουργίας και των επιδόσεων. Επιπλέον, η αναφορά θα πρέπει να περιέχει και τα ακόλουθα ανά ενότητα.

Ενότητα 1

Παρουσιάστε την λειτουργία της `motEstP`. Εφαρμόστε την στην εικόνα "coastguard003.tiff", και συγκεκριμένα στο macroblock 0. Το frame αναφοράς θα είναι η εικόνα "coastguard001.tiff". Παρουσιάστε (μόνο για το κανάλι Y) το microblock 0 της εικόνας "coastguard003.tiff", το χωρίο της εικόνας "coastguard001.tiff" που διαλέξατε βάσει της `motEstP`, το motion vector, και το σφάλμα πρόβλεψης, και δείξτε το αποτέλεσμα με βάση τις αρχικές εικόνες.

Ενότητα 2

Για την δεύτερη ενότητα θα πρέπει να παρουσιάσετε τη συμπίεση που πετυχαίνει ο κωδικοποιητής σας, καθώς και τα απόλυτα μεγέθη (σε bytes) των δύο αναπαραστάσεων: (α) σε tiff, και (β) στην κωδικοποιημένη μορφή. Για την αποτίμηση του μέγεθους της κωδικοποιημένης αναπαράστασης δεν μπορείτε να χρησιμοποιήσετε το μέγεθος του αρχείου mat που παράγει ο κωδικοποιητής σας. Χρησιμοποιείστε τη συνάρτηση `getSeqEntityBits` που σας δίνεται η οποία υπολογίζει τον αριθμό των απαιτούμενων bits για ένα struct τύπου `SeqEntity`.

Επίσης παρουσιάστε το μέσο απόλυτο σφάλμα ανά εικόνα. Συγκεκριμένα, έστω x μία εικόνα όπως σας δίνεται, και έστω y η ίδια εικόνα όπως παράγεται από τον κωδικοποιητή. Και οι δύο εικόνες είναι πίνακες διαστάσεων $[720 \times 576 \times 3]$, με τιμές ακέραιους στο διάστημα $[0, 255]$. Το μέσο απόλυτο σφάλμα μπορεί να υπολογισθεί ως ο μέσος όρος των απόλυτων τιμών των στοιχείων του πίνακα $x - y$.

Τέλος, παρουσιάστε το κανάλι Y της εικόνας "coastguard002.tiff" και το σφάλμα πρόβλεψης.

Ενότητα 3

Μετρήστε το μήκος του stream (σε bits) που δημιουργεί η ενότητα 3. Θα παρατηρήσετε ότι είναι μικρότερο από αυτό που υπολογίσατε στην ενότητα 2 από το struct `SeqEntity`. Πού οφείλεται η διαφορά αυτή;

Παραδοτέα

Κατά την υποβολή της εργασίας θα πρέπει να παραδώσετε:

1. Τα αρχεία κώδικα MATLAB που υλοποιήσατε με αναλυτικά σχολιασμένο κώδικα στα αγγλικά, χωρισμένα σε 3 φακέλους, κατ' αντιστοιχία με τα 3 επίπεδα της εργασίας. Συμβουλευτείτε τον Πίνακα 1.
2. Αναφορά σε PDF που θα περιγράφει τον τρόπο υλοποίησης του προβλήματος (μαθηματικά και προγραμματιστικά) και θα *παρουσιάζει - σχολιάζει* τα αποτελέσματα σε χωριστή ενότητα για κάθε σύστημα.

Επιπλέον

- Υποβάλετε ένα και μόνο αρχείο, τύπου zip.
- Το όνομα του αρχείου πρέπει να είναι `AEM1.zip` για τις ομάδες ενός ατόμου και `AEM1_AEM2.zip` για τις ομάδες δύο ατόμων, όπου `AEM1` είναι το ΑΕΜ του πρώτου φοιτητή της ομάδας, και `AEM2` είναι το ΑΕΜ του δεύτερου φοιτητή της ομάδας, αν υπάρχει.
- Το προς υποβολή αρχείο πρέπει να περιέχει τους 3 φακέλους που αναφέρονται στον Πίνακα 1 και το αρχείο `report.pdf` το οποίο θα είναι η αναφορά της εργασίας.
- Αν κατασκευάσετε και χρησιμοποιήσετε επιπλέον συναρτήσεις από αυτές που ζητούνται, τοποθετήστε τα αντίστοιχα αρχεία κώδικα στον κατάλληλο φάκελο.
- Μην υποβάλετε αρχεία που δεν χρειάζονται για την λειτουργία του κώδικά σας, ή φακέλους/αρχεία που δημιουργεί το λειτουργικό σας, πχ "Thumbs.db", ".DS_Store", ".directory".
- Μην υποβάλετε τα εργαλεία που σας δίνονται (`testQ*`, `setup.p`, κλπ)
- Μην υποβάλετε τα δείγματα ήχου που σας δίνονται.
- Η αναφορά πρέπει να είναι ένα αρχείο τύπου PDF, και να έχει όνομα `report.pdf`.
- Για την ονομασία των αρχείων, φακέλων, κλπ που περιέχονται στο προς υποβολή αρχείο, χρησιμοποιείτε μόνο αγγλικούς χαρακτήρες, και όχι ελληνικούς ή άλλα σύμβολα, πχ "#", "$", "%" κλπ.
- Όλα τα αρχεία εκτός της αναφοράς πρέπει να είναι αρχεία κειμένου τύπου UTF-8.

Καλή επιτυχία!

Παράρτημα

Στη συνέχεια δίνεται μια περιγραφή των συναρτήσεων που θα υλοποιήσετε, σε μορφή ψευδοκώδικα. Παρόλα αυτά, ο κώδικας υλοποίησης του παραδοτέου απαιτείται να είναι συμβατός *μόνο ως προς τις εισόδους και τις εξόδους αυτών των συναρτήσεων*. Η εσωτερική υλοποίησή τους είναι δική σας επιλογή (και η ευθύνη μας περιορισμένη).

```
function encodeMPEG
    genSeqHeader % Generate the header of the sequence
    writeSeqHeader
    encodeSeq
    writeSeqEnd
```

```
function encodeSeq
    while true
        genGoPHeader
        writeGoPHeader
        t = encodeGoP
        if t == 0
            break % Unexpected end of GoP found, assuming end of sequence
        end
    end
```

```
function encodeGoP
    for picIndex = 1:sizeOfGoP
        pic = getNextPic % Get next picture
        if isempty(pic)
            return(0) % No picture found
        end
        genPicSliceHeader % Generate headers for picture and slice
        writePicSliceHeader
        encodePicSlice
        if picType == I || P
            decodePicSlice
            pushPic % Save to picture buffer for future reference
        end
    end
    return(1)
```

```
function encodePicSlice
    for mBIndex = 1:numOfMB
         genMBHeader
         writeMBHeader
         encodeMB
         writeMV
         writeMBEnd
    end
```

```
function encodeMB
    switch (picType) % Determine picture type. Perform motion estimation,
                     % if necessary, and code the corresponding motion vectors
        case P
            motEstP
            encodeMV
        case B
```

```
            motEstB
            encodeMV
    end
    for bIndex = 1:numOfBlocks
         encodeBlock
    end
```

```
function encodeBlock
    C = blockDCT(I)
    quantize(c, qTable, qScale)
    runLength(c)
    vlc(c)
    writeVLC
```

```
function decodeMPEG
    readSeqHeader
    decodeSeq
```

```
function decodeSeq}
    while true
        readGoPHeader
        t = decodeGoP
        if t == 0
            break % Unexpected end of GoP found, assuming end of sequence
        end
    end
```

```
function decodeGoP
    for picIndex = 1:sizeOfoP
         readPicSliceHeader
         dPic = decodePicSlice
         if isempty(dPic)
             return(0) % No picture found
         end
         pushFlushPic % Send picture to output buffer
         if picType == I || P
             pushPic % Save to picture buffer for future reference
         end
    end
```

```
function decodePicSlice
    for mBIndex = 1:numOfMB
         readMBHeader
         decodeMB
    end
```

```
function decodeMB
    if picType == P || B % Read motion vectors if necessary
        readMV
    end
    for bIndex = 1:numOfBlocks
         readBHeader
         x(bIndex) = decodeBlock
```

```
    end
    switch (picType)
        case P
            mB = imotestP(x, mV, buffer)
        case B
            mB = imotestB(x, mV, buffer)
        case I
            mB = x
    end
```

```
function decodeBlock
    c = readVLDC % Read VL Coded block from file
    iVLC(c) % Decode VLC
    iRunLength(c)
    dequantize(c, qTable, qScale)
    x = iBlockDCT(c)
```