ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ

Τμήμα μηχανικών πληροφορικής

Πτυχιακή Εργασία

Ανάπτυξη μη απωλεστικού αλγορίθμου συμπίεσης για ιατρικά δεδομένα, για εφαρμογή σε ενσωματωμένα σε συστήματα.

Ντάγκινη Νεκταρία Κυριακή (ΑΜ :1060)

Επιβλέποντες Καθηγητές:
Αντωνόπουλος Χρήστος, Βώρος Νικόλαος

Αντίρριο 2016

Ευχαριστίες

Abstract

Medical data compression is very important process today for reducing data archival volume and the transmission(speed, data rate). Lossless compression achieves that by removing redundant information while maintaining the quality of the data, and guaranteeing that the reconstructed data will be the same as the original.

LEC algorithm (Lossless entropy compression) is a popular compression method that

Counts the frequency of occurrence of each unique symbol in the given text and then it is replaced by the unique symbol generated by the algorithm.

Programmable devices are used widely for the implementation of compression algorithms. Embedded systems are based on programmable devices like fpga which can be programmed by keil uvision.

The proposed work is to compress medical data without any loss using LEC algorithm. The code is written in C and for reference and verification we have implemented it in MATLAB. It has been implemented on Microsoft Visual Studio Community.

Περίληψη

Η συμπίεση ιατρικών δεδομένων είναι μια πολύ σημαντική διαδικασία σήμερα τόσο γιατί επιτυγχάνεται μείωση του όγκου αποθήκευσης όσο και την βελτίωση της μετάδοσης τους(ταχύτητα, data rate). Η μη απωλεστική συμπίεση επιτυγχάνει τα παραπάνω με την αφαίρεση της περιττής πληροφορία ενώ διατηρεί την ποιότητα των δεδομένων και εγγυάται ότι τα ανακατασκευασμένα δεδομένα θα είναι τα ίδια με τα αρχικά.

Ο αλγόριθμος LEC (Μη απωλεστική συμπίεση εντροπίας) είναι δημοφιλής μέθοδος συμπίεσης που μετράει τη συχνότητα εμφάνισης κάθε μοναδικού συμβόλου στο δοσμένο κείμενο και στη συνέχεια το αντικαθιστά με ένα σύμβολο που δημιουργείται από τον αλγόριθμο.

Οι προγραμματιζόμενες συσκευές χρησιμοποιούνται ευρέως για την εφαρμογή αλγόριθμων συμπίεσης. Τα ενσωματωμένα συστήματα βασίζονται σε προγραμματιζόμενες συσκευές όπως FPGA που μπορούν να προγραμματιστούν από τη γλώσσα C.

Η προτεινόμενη εργασία είναι η μη απωλεστική συμπίεση ιατρικών δεδομένων με τη χρήση του αλγορίθμου LEC. Ο κώδικας είναι γραμμένος σε γλώσσα C και για αναφορά και επαλήθευση έχει εφαρμοστεί ο ίδιος αλγόριθμος σε MATLAB. Το προγραμματιστικό περιβάλλον ήταν το Microsoft Visual Studio Community

Περιεχόμενα

Κεφάλαιο 1

1.1 Εισαγωγή στη συμπίεση δεδομένων

Στην σύγχρονη εποχή η αξία των πληροφοριών γίνεται όλο και πιο σημαντική. Οι άνθρωποι έχουν την τάση να συσσωρεύουν δεδομένα. Επίσης η μεταφορά των δεδομένων πρέπει να είναι ταχύτατη ή τουλάχιστον ικανοποιητική, διάρκειας λίγων δευτερολέπτων ώστε να μην προκαλείται δυσαρέσκεια στον συνηθισμένο στον φρενήρη ρυθμό της εποχής μας άνθρωπο. Οι πληροφορίες πλέον όλο και περισσότερο χρησιμοποιούνται σε ψηφιακή μορφή δηλαδή σε μορφή αριθμών που αναπαρίστανται από byte δεδομένων. Η διατήρηση των δεδομένων όμως απαιτεί αποθηκευτικούς πόρους, πόρους στη μετάδοση καθώς και ενέργεια για τη μεταφορά τους. Για τους παραπάνω λόγους έχει γίνει δημοφιλής η διαδικασία της συμπίεσης δεδομένων.

Για παράδειγμα, για να παρουσιάσουμε ψηφιακά ένα βίντεο ένα δευτερόλεπτο χωρίς συμπίεση (με CCIR 601 format), χρειαζόμαστε περισσότερο από 20 megabytes, ή 160 megabits. Αν σκεφτούμε ότι τη διάρκεια μιας ταινίας θα καταλάβουμε γιατί μας είναι απαραίτητης η συμπίεση. Επίσης ένα δίλεπτο ασυμπίεστης μουσικής ποιότητας CD (44,100 δείγματα ανά δευτερόλεπτο, 16 bits το δευτερόλεπτο) απαιτεί περισσότερα από 84 εκατομμύρια bits. Έτσι το να κατεβάσουμε μουσική από έναν ιστότοπο σε αυτούς τους ρυθμούς θα έπαιρνε πολύ ώρα.

Καθώς η δραστηριότητα του ανθρώπου έχει όλο και μεγαλύτερο αντίκτυπο στο περιβάλλον, υπάρχει η αυξανόμενη ανάγκη για περισσότερη πληροφορία γι αυτό, πως λειτουργεί, και τι του προκαλούμε. Διάφορα διαστημικά πρακτορεία από όλο τον κόσμο, συμπεριλαμβανομένου και του ευρωπαϊκού, παγκόσμιου, καναδικού, ιαπωνικού διαστημικού πρακτορείου (αντίστοιχα ESA,NASA,CSA,STA), συνεργάζονται σε ένα πρόγραμμα για την παρακολούθηση της παγκόσμιας αλλαγής του περιβάλλοντος και παράγεται μισό terabyte δεδομένων* την ημέρα.

Τα παραπάνω παραδείγματα δίνουν μια εικόνα για την αναγκαιότητα της συμπίεσης δεδομένων. Το ερώτημα που τίθεται είναι γιατί να επικεντρωθούμε στη συμπίεση δεδομένων και να μην ασχοληθούμε με το να εξελίξουμε την τεχνολογία στον τομέα της μεταφοράς και της αποθήκευσης δεδομένων. Η τεχνολογία εξελίσσεται και σε αυτό όμως δεν αρκεί. Έχουν γίνει σημαντικά βήματα που επιτρέπουν την αποθήκευση και τη μετάδοση όλο και περισσότερων πληροφοριών χωρίς συμπίεση, όπως τα CD-ROMs, οπτικές ίνες, ADSL, καλωδιακά μόντεμ. Παρόλα αυτά, ενώ η ικανότητα αποθήκευσης και μετάδοσης βελτιώνεται με τις καινοτομίες της τεχνολογίας, σύμφωνα με το πόρισμα του πρώτου νόμου του Parkinson, φαίνεται ότι η ανάγκη για αποθήκευση και μετάδοση αυξάνεται τουλάχιστον δύο φορές γρηγορότερα. Υπάρχουν και περιπτώσεις που δεν δύναται βελτίωση στη χωρητικότητα για παράδειγμα μέσω του αέρα, λόγω των χαρακτηριστικών της ατμόσφαιρας, η πληροφορία που μπορούμε να μεταδώσουμε είναι περιορισμένη.

Ένα πρώιμο παράδειγμα συμπίεσης δεδομένων είναι ο κώδικας του Morse, που αναπτύχθηκε από τον Samuel Morse στα μέσα του 19ου αιώνα. Τα γράμματα του αλφάβητου που στέλνονταν από τον τηλέγραφό κωδικοποιούνταν με τη χρήση μόνο τελειών και παυλών. Ο Morse παρατήρησε ότι συγκεκριμένα γράμματα εμφανίζονται συχνότερα από άλλα. Έτσι προκειμένου να μειώσει το μέσο χρόνο που απαιτείται για να σταλεί ένα μήνυμα ανέθεσε μικρότερες ακολουθίες σε γράμματα που εμφανίζονται πιο συχνά, όπως το e (.) και το a(. -), και μεγαλύτερες ακολουθίες σε γράμματα που εμφανίζονται λιγότερο συχνά, όπως το q(- - . -) και το j(. - - -). Η ιδέα αυτή της χρήσης δηλαδή μικρότερων ακολουθιών για πιο συχνά εμφανιζόμενους χαρακτήρες χρησιμοποιείται στην κωδικοποίηση Huffman.

Ενώ η κωδικοποίηση Morse χρησιμοποιεί τη συχνότητα εμφάνισης κάθε χαρακτήρα, μια ευρέως χρησιμοποιούμενη μορφή αύτη της κωδικοποίησης Braille, η οποία επίσης αναπτύχθηκε στα μέσα του 19ου αιώνα, χρησιμοποιεί τη συχνότητα εμφάνισης λέξεων. Στην κωδικοποίηση Braille, χρησιμοποιείτε ένας 2x3 πίνακας ανάγλυφων τελειών για να αναπαρασταθεί το κείμενο. Μ' αυτές γίνονται 64 συνδυασμοί, που αντιστοιχούν στα γράμματα του αλφάβητου κάθε χώρας και τους αριθμούς αλλά και κάποιες συχνά χρησιμοποιούμενες λέξεις όπως το "και". Αυτές οι τροποποιήσεις μαζί με τη δομή των λέξεων(συχνότητα εμφάνισης) έχουν ως αποτέλεσμα μια μείωση χωρητικότητας, η συμπίεση, 20%.

Με τη βοήθεια της στατιστικής δομής μπορούμε να έχουμε συμπίεση στα παραπάνω παραδείγματα όμως υπάρχει και άλλου είδους δομή στα δεδομένα πέραν αυτής. Ας σκεφτούμε την ομιλία. Όταν μιλάμε ο μηχανισμός της παραγωγής της φωνής καθορίζει τα είδη των ήχων που μπορούμε να παράγουμε. Έτσι, αντί να μεταδώσουμε μια ομιλία μπορούμε να στείλουμε πληροφορίες για τη διάπλαση του λάρυγγα οι οποίες θα χρησιμοποιηθούν από τον δέκτη ώστε να συνθέσει την ομιλία. Οι πληροφορίες για τη διάπλαση του λάρυγγα μπορούν να παρασταθούν καλύτερα από τους αριθμούς που είναι οι τιμές της φωνής που έχουν υποστεί δειγματοληψία. Επομένως, έχουμε συμπίεση. Αυτού του είδους η συμπίεση χρησιμοποιείτε σε ένα αριθμό εφαρμογών, συμπεριλαμβανομένου τη μετάδοση της ομιλίας στα ασύρματα ραδιόφωνα και στη συνθετική φωνή των παιχνιδιών που μιλούν. Ο Homer Dudley στα εργαστήρια της Bell το 1936 ανέπτυξε μια πρώιμη εκδοχή αυτής της προσέγγισης συμπίεσης που ονομαζόταν vocoder (voice coder).

Η συμπίεση μπορεί να επιτευχθεί με τη χρήση της δομής των δεδομένων όπως στα παραπάνω παραδείγματα αλλά και με τη χρήση άλλων στοιχείων όπως για παράδειγμα λαμβάνοντας υπόψη τα χαρακτηριστικά του χρήστη των δεδομένων. Η ομιλία και οι εικόνες είναι δεδομένα που προορίζονται για τον άνθρωπο αλλά η ικανότητα αντίληψης του ανθρώπου είναι περιορισμένη. Για παράδειγμα, ο άνθρωπος δεν μπορεί να ακούσει τους ήχους με πολύ υψηλή συχνότητα που μπορεί ένας σκύλος. Επομένως, μπορούμε να παραβλέψουμε τις πληροφορίες τέτοιου είδους επιτυγχάνοντας και πάλι συμπίεση

Η συμπίεση δεδομένων είναι η επιστήμη που ασχολείται με την εξέλιξη μεθόδων που μπορούν να εφαρμοστούν στα δεδομένα προκειμένου να καταλαμβάνουν όλο και λιγότερο χώρο. Η συμπίεση δεδομένων πιο συγκεκριμένα είναι η διαδικασία μετατροπής ενός data stream (ρεύμα δεδομένων) εισόδου (το source stream ή τα αρχικά ανεπεξέργαστα δεδομένα) σε ένα άλλο data stream (έξοδος ή συμπιεσμένο data stream) που έχει μικρότερο μέγεθος. Ένα stream είναι είτε ένα αρχείο είτε ένας buffer στη μνήμη.

Για να γίνει συμπίεση:

Μια συνάρτηση f εφαρμόζεται σε κάθε ενότητα δεδομένων di ώστε να παραχθούν δεδομένα $mi \rightarrow f(di) = mi$

Για να γίνει αποσυμπίεση:

Μια συνάρτηση f_r εφαρμόζεται στα δεδομένα mi ώστε να παραχθούν τα δεδομένα $dr \rightarrow f_r(mi) = d_r$

Ο τομέας της συμπίεσης δεδομένων(data compression) καλείται συχνά κωδικοποίηση πηγαίου κώδικα (source coding).

Η συμπίεση δεδομένων έκτος από τα πλεονεκτήματα που προαναφέρθηκαν έχει και μειονεκτήματα και αυτό γιατί τα συμπιεσμένα δεδομένα για να χρησιμοποιηθούν πρέπει πρώτα να αποσυμπιεστούν. Η διαδικασία όμως της αποσυμπίεσης αποτελεί μια επιπλέον επεξεργασία που επιβάλλει υπολογιστικό ή άλλου είδους κόστος.

Γενικότερα η συμπίεση δεδομένων είναι μια διαδικασία που περιλαμβάνει συμβιβασμούς (trade-off) μεταξύ των διάφορων παραγόντων συμπεριλαμβανομένου του βαθμού συμπίεσης, του ποσού της παραμόρφωσης που εισάγεται (για lossy συμπίεση δεδομένων), καθώς και των υπολογιστικών πόρων που απαιτούνται για τη συμπίεση και αποσυμπίεση των δεδομένων.

Υπάρχουν πολλές μέθοδοι για να επιτευχθεί συμπίεση δεδομένων. Όλες οι υπάρχουσες μέθοδοι είναι διαφορετικές, βασίζονται σε διαφορετικές ιδέες, κατάλληλες για διαφορετικού είδους δεδομένα και παράγοντας διαφορετικά αποτελέσματα αλλά όλες έχουν την ίδια αρχή στην οποία βασίζονται η οποία είναι η συμπίεση δεδομένων εξαλείφοντας τον πλεονασμό από τα αρχικά δεδομένα στο αρχείο προέλευσης. Η ιδέα της συμπίεσης αφαιρώντας τον πλεονασμό αποτελεί τον γενικό νόμο της συμπίεσης δεδομένων , δηλαδή το να αναθέτουμε μικρούς κωδικούς σε κοινά σύμβολα ή φράσεις και μεγάλους κωδικούς σε σπάνια σύμβολα ή φράσεις. Δεν υπάρχει αλγόριθμος συμπίεσης δεδομένων που να εφαρμόζεται σε όλους τους τύπους δεδομένων και να είναι εξίσου αποδοτικός. Για να γίνει η διαδικασία της συμπίεσης, ο αλγόριθμος πρέπει να εξετάσει τα δεδομένα, να βρει τους πλεονασμούς σε αυτά και να προσπαθήσει να τους αφαιρέσει και καθώς οι πλεονασμοί εξαρτώνται από το είδος δεδομένων(κείμενο, εικόνες, ήχος, κτλ.) ο αλγόριθμος που χρησιμοποιείται πρέπει να είναι σχεδιασμένος για το συγκεκριμένο είδος ώστε να αποδίδει καλύτερα σε αυτό. Οι περισσότεροι μέθοδοι κατηγοριοποιούνται στις: run length encoding (RLE), statistical methods, dictionary-based.

1.2 Συμπίεση ιατρικών δεδομένων αισθητήρα

Πριν προχωρήσω όμως σε μια γενικότερη κατηγοριοποίηση(και με χρήση απλών παραδειγμάτων για κατανόηση) κάποιον μεθόδων συμπίεσης, θα αναφερθώ στη σημασία της χρήσης αλγορίθμων συμπίεσης σε ιατρικά δεδομένα αισθητήρων που αποτελεί και το θέμα της εργασίας.

Γρήγορα συστήματα ψηφιακής επεξεργασίας σημάτων (high-speed digital signal processing systems- DSP) όπως ασύρματων υποδομών (wireless infrastructure), επεξεργασία ραντάρ και υποσυστήματα αισθητήρων ιατρικής απεικόνισης υιοθέτησαν μόλις πρόσφατα την χρήση αλγορίθμων συμπίεσης παρά την επικράτηση και η επιτυχία των αλγορίθμων συμπίεσης φωνής, ήχου, εικόνας και βίντεο στον τομέα των καταναλωτικών ηλεκτρονικών προϊόντων (consumer electronics). Η συμπίεση των δειγμάτων από ιατρικούς αισθητήρες μειώνει το κόστος μεταφοράς και αποθήκευσης πριν την ανακατασκευή των δειγμάτων (κυρίως για δείγματα ιατρικής απεικόνισης καθώς εκεί έγκειται το πρόβλημα, έχουν μεγάλες απαιτήσεις χωρητικότητας).

Τα δείγματα ιατρικών αισθητήρων διαφέρουν από τα δείγματα ηχητικών και οπτικών σημάτων στα εξής:

1) οι αισθητήρες ιατρικής απεικονίσεις παράγουν εκατοντάδες-χιλιάδες stream ενώ τα σήματα ήχου-βίντεο παράγουν λίγα.

2)το bandwidth και η διακύμανση των απαιτήσεων των ιατρικών αισθητήρων είναι μεγάλες για παράδειγμα 10 ksamp/s στα 20 b/sample για την αξονική τομογραφία. Ενώ η συμπίεση ενός ηχητικού σήματος καθορίζεται από την ακουστική ικανότητα του ανθρώπου η εμβέλεια της οποίας είναι 130 dB και 20 KHz bandwidth. Γι αυτό τέτοιου είδους σήματα δε χρειάζονται αποσυμπίεση με περισσότερα από 24 b/sample ή με περισσότερα από 2 κανάλια (stereo)

3)Ο ήχος και το βίντεο αξιολογούνται από τον άνθρωπο ενώ τα ιατρικά δεδομένα από μηχανήματα που λειτουργούν με συγκεκριμένους απαιτητικούς αλγορίθμους που χρειάζονται μεγάλο bandwidth και dynamic range ώστε να κάνουν σωστή ανακατασκευή σημάτων/ εικόνων.

Τα ιατρικά σήματα αισθητήρων και κυρίως των μεθόδων απεικόνισης αντιμετωπίσουν πρόβλημα διότι όλο και αυξάνονται τα δεδομένα που λαμβάνουν οι αισθητήρες. Συγκεκριμένα :

- ο αριθμός των καναλιών των αισθητήρων αυξάνεται ώστε να σαρώνονται γρήγορα μεγαλύτερες περιοχές
- ο ρυθμός δειγμάτων σε κάθε αισθητήρα αυξάνεται
- το δυναμικό εύρος των ιατρικών σημάτων αυξάνεται (Dynamic Range- DR)

[TABLE 1] BANDWIDTH REQUIREMENTS FOR CT, ULTRASOUND, MRI, AND DIGITAL X-RAY SYSTEMS.

SENSOR SIGNAL TYPE	NEXT-GENERATION SENSOR BANDWIDTH	TYPICAL LOSSLESS* COMPRESSION RATIOS	FIXED-RATE* COMPRESSION RATIOS GENERATING CLINICALLY ACCEPTABLE IMAGES
COMPUTED TOMOGRAPHY	~80 Gb/s	2:1	≤5:1
ULTRASOUND (RF SIGNAL)	~200 Gb/s	1:85:1	≤3:1
ULTRASOUND (BEAMFORMED SIGNAL)	~20 Gb/s	2:1	≤4:1
MAGNETIC RESONANCE IMAGING	~5 Gb/s	4:1	≤6:1
DIGITAL X-RAY	~4 Gb/s	1:8:1	≤3:1

Εικόνα 1. Bandwidth requirements for CT, Ultrasound, MRI, digital X-Ray systems

Καθώς όμως τα ιατρικά δεδομένα παρουσιάζουν και μια ποικιλία τίθεται το ερώτημα αν θα μπορούσε ένας και μόνος αλγόριθμος συμπίεσης να τα συμπιέσει αποτελεσματικά. Τα ιατρικά δεδομένα αισθητήρων παρουσιάζουν
τα παρακάτω κοινά χαρακτηριστικά:

1. Γίνεται συνήθως υπέρ-δειγματοληψία.
2. Έχουν μέτριο έως μεγάλο peak- to- average ratio (PAR)
3. Η δειγματοληψία τους γίνεται από ατελή ADCs.
   Οι μετατροπείς ADC (analog to digital converter) κάνουν την δειγματοληψία βάση ανάλυσης(μιλάμε για εικόνες) και αποτελεσματικό αριθμό bit (effective number of bits- ENOB)

Real Time- Πραγματικός χρόνος

Πολύ σημαντικός όρος για τη συμπίεση. Όταν λέμε συμπίεση δεδομένων ιατρικών αισθητήρων σε πραγματικό χρόνο εννοούμε ότι ο αλγόριθμος συμπίεσης πρέπει να λειτουργήσει αρκετά γρήγορα ώστε να συμπιέσει όλα τα δείγματα που παράγονται από τον ADC μετά τη λήψη τους από τους αισθητήρες. Μια άλλη οπτική αυτού του όρου σχετίζεται με την ταχύτητα της ανακατασκευής του εικονικού σήματος.

Επειδή οι μετατροπείς ADC που μετατρέπουν τα σήματα των εικόνων που λαμβάνονται από τους αισθητήρες σε ψηφιακά, συνήθως είναι συνδεμένα σε FPGA και η πρώτη συμπίεση θα εκτελεσθεί σε αυτά.

Ένας παράγοντας που εξαρτάται και η συμπίεση των ιατρικών δεδομένων είναι το hardware.

Συμπίεση με απώλειες- Συμπίεση χωρίς απώλειες (Lossy-lossless) στα δεδομένα ιατρικών αισθητήρων.

Η συμπίεση χωρίς απώλειες παρέχει την καλύτερη ποιότητα αλλά έχει απρόβλεπτο βαθμό συμπίεσης αλλά έχουμε τη σιγουριά ότι τα δεδομένα μας θα είναι ίδια με τα αρχικά. Η συμπίεση αυτού του είδους γίνεται ελκυστική μέθοδος όταν μπορεί να επιτύχει ½ συμπίεση των δεδομένων του αισθητήρα. Υψηλά ποσοστά συμπίεσης δεν μπορούν να επιτευχθούν εύκολα με μεθόδους συμπίεσης χωρίς απώλειες σε σήματα όπως EEG, λόγω της τυχαιότητας που παρουσιάζουν. Ένας κοινός τρόπος για να βελτιωθεί η αναλογία συμπίεσης είναι η χρήση των μαθηματικών μετασχηματισμών(βλέπε transform encoding). Μια γνωστή μέθοδος για την πολλαπλών καναλιών συμπίεση EEG, είναι αυτή με χρήση Karhunen-Loeve transform (KLT). Κύριο μειονέκτημα της KLT είναι ο μεγάλος χρόνος υπολογισμού.

Μια άλλη προσέγγιση είναι η χρήση neural network predictors για context-based μοντελοποίηση σφαλμάτων. Αυτή η τεχνική έχει δείξει κάποια βελτίωση στην απόδοση συμπίεσης αφαιρώντας το bias offset των αρχικών δεδομένων.

Η μέθοδος συμπίεσης με απώλειες είναι επίφοβη για τους ασθενείς και τους γιατρούς για ενδεχόμενη απώλεια δεδομένων με συνέπεια την λάθος διάγνωση σε ασθενή παρόλο που πειράματα και στατιστικές αναλύσεις για την εφαρμογή αυτής της μεθόδου δεν έχουν δείξει μεγάλη παραλλαγή στα δεδομένα ώστε να έχουμε τέτοιες συνέπειες.

Η συμπίεση ιατρικών δεδομένων από αισθητήρες θα πρέπει να παρέχει την επιλογή ρυθμού συμπίεσης και μετά να χρησιμοποιείτε ο αποδεκτός ρυθμός που οδηγεί σε κλινικά δεκτά αποτελέσματα. Ιδανικά επίσης, ο αλγόριθμος συμπίεσης θα πρέπει να προσφέρει και επιλογή για συμπίεση χωρίς απώλειες για εκείνες τις σπάνιες περιπτώσεις που όλα τα δεδομένα που προσλαμβάνουν οι αισθητήρες πρέπει να είναι απαράλλακτα.

1.3 Electroencephalogram (EEG) and Electrocardiogram (ECG)

Στην παρούσα διπλωματική εργασία τα σήματα με τα οποία ασχοληθήκαμε για να ελέγξουμε την απόδοση του το LEC αλγορίθμου είναι ηλεκτροεγκεφαλογραφήματα (eeg) και ηλεκτροκαρδιογραφήματα(ecg).

Το ηλεκτροεγκεφαλογράφημα ή EEG σχετίζεται με τον εγκέφαλο και το ηλεκτροκαρδιογράφημα ή ECG σχετίζεται με την καρδιά. Το EEG είναι ο εξοπλισμός που χρησιμοποιείται για τη μέτρηση ηλεκτρικών δραστηριοτήτων του εγκεφάλου και από την άλλη το ECG χρησιμοποιείται για μέτρηση των δραστηριοτήτων της καρδίας.

Το EEG χρησιμοποιείται κυρίως για τη διάγνωση επιληπτικών διαταραχών, λοιμώξεων, όγκων, εκφυλιστικών και μεταβολικών διαταραχών που θα μπορούσαν να επηρεάσουν τον εγκέφαλο. Αντιθέτως, το ECG χρησιμοποιείται για τον προσδιορισμό του ρυθμού της καρδιάς, τις θέσεις των θαλάμων της καρδιάς και αν υπάρχει οποιαδήποτε βλάβη στην καρδιά.

1.4 Σκοπός

Με τα παραπάνω γίνεται αντιληπτή η σημαντικότητα της συμπίεσης και ιδιαίτερα στα ιατρικά δεδομένα. Στην εργασία αυτή στοχεύουμε στο να μετατρέψουμε την εφαρμογή του αλγορίθμου συμπίεσης LEC από MATLAB σε C.

Αρχικά τα δεδομένα του σήματος προς συμπίεση, τα διαβάζουμε σε 16αδικο σύστημα και σε Little Endian μορφή. Στη συνέχεια μετατρέπουμε κάθε χαρακτήρα 2 byte σε μια ακολουθία 0 και 1 της οπαίας το μήκος μεγαλώνει ανάλογα με το μέγεθος αυτού του χαρακτήρα στο δεκαδικό σύστημα. Τα αποτελέσματα αυτών τον μετατροπών, δηλαδή τις ακολουθίες 0 1, τα συλλέγουμε και τα πακετάρουμε σε τμήματα των 32 bits τα οποία και αποθηκεύουμε(πίνακα ή αρχείο). Έχοντας ορίσει ένα ρυθμό με τον οποίο θα πραγματοποιείται η συμπίεση, συμπιέζουμε τα δεδομένα που έχουμε μετρώντας και το χρόνο συμπίεσης. Σκοπός είναι η εφαρμογή του LEC σε C, γλώσσα κατάλληλη για η υλοποίηση σε πραγματικού χρόνου ενσωματωμένα συστήματα, με ικανοποιητικό ποσοστό συμπίεσης των ιατρικών δεδομένων εισόδου.

Κεφάλαιο 2

2.1 Είδη κωδικοποίησης:

Συμπίεση με απώλειες και χωρίς απώλειες

Υπάρχουν δύο κατηγορίες αλγορίθμων συμπίεσης, οι απωλεστικοί ή μη αντιστρεπτοί (lossy) και οι μη απωλεστικοί ή αντιστρεπτοί (lossless) αλγόριθμοι.

Στους απωλεστικούς αλγορίθμους, όταν γίνει η συμπίεση και μετά ακολουθήσει αποσυμπίεση των δεδομένων, η τελική ακολουθία των δεδομένων διαφέρει από την αρχική. Αντίθετα στους μη απωλεστικούς αλγορίθμους, η διαδικασία συμπίεσης και αποσυμπίεσης επαναφέρει την αρχική ακολουθία. Αν πρέπει να μεταφερθούν δεδομένα με απόλυτη ακρίβεια, πιστότητα, χωρίς να αλλοιωθεί το περιεχόμενό τους, πρέπει να εφαρμοστεί μια μη απωλεστική μέθοδος συμπίεσης. Εφαρμογές ή συστήματα που μεταδίδουν αναλλοίωτες πληροφορίες από το ένα μέσο στο άλλο, για παράδειγμα κάρτες δικτύου ή modem, χρησιμοποιούν τεχνικές μη απωλεστικής συμπίεσης. Ιατρικά σήματα όπως τα υπερηχογραφήματα συμπιέζονται με τεχνικές μη απωλεστικής συμπίεσης καθώς τέτοιου είδους σήματα δεν πρέπει να αλλοιώνονται κατά τη διάρκεια της συμπίεσης, όπως επίσης και δεδομένα που συγκεντρώθηκαν από δορυφόρους. Υπάρχουν όμως εφαρμογές όπου η μικρή διαφοροποίηση από την αρχική μορφή των δεδομένων δεν επιφέρει σημαντικές αλλαγές. Έτσι, οι περισσότερες εφαρμογές που έχουν να κάνουν με σύνθετες μορφές δεδομένων όπως είναι ο ήχος, η εικόνα, το video, όπου το τελικό αποτέλεσμα αξιολογείται από τον ανθρώπινο παράγοντα (αυτί, μάτι), μπορούν να κάνουν απωλεστική συμπίεση χωρίς πολλές φορές να υπάρχουν εμφανείς αλλοιώσεις στην ποιότητα των δεδομένων. Σε γενικές γραμμές, η ερευνά που γίνεται καθοδηγεί τα απωλεστικά συστήματα συμπίεσης δεδομένων για το πώς οι άνθρωποι αντιλαμβάνονται τα εν λόγω δεδομένα.

Παράδειγμα μη απωλεστικής συμπίεσης:

Αν η αρχική πληροφορία είχε τη μορφή

ΠΠΠΟΟΟΟΟΟΛΛΥΥΥΥΥΥΜΜΕΕΕΕΕΕΕΕΣΣΣΑΑΑΑΑΑΑ

αφαιρώντας τον πλεονασμό συμπιέζεται 3Π 6Ο 2Λ 7Υ 2Μ 8Ε 3Σ 7Α

Κατά την αποκωδικοποίηση η πληροφορία αναπαράγεται με την αρχική της μορφής

ΠΠΠΟΟΟΟΟΟΛΛΥΥΥΥΥΥΜΜΕΕΕΕΕΕΕΕΣΣΣΑΑΑΑΑΑΑ

Ευρέως διαδεδομένοι μη απωλεστικοί αλγόριθμοι είναι οι: Run Length Encoding (RLE), Huffman, Delta, LZW.

Παράδειγμα απωλεστικής συμπίεσης:

Αν η αρχική πληροφορία είχε τη μορφή

ΠΠΠΟΟΟΟΟΟΛΛΥΥΥΥΥΥΜΜΕΕΕΕΕΕΕΕΣΣΣΑΑΑΑΑΑΑ

Αφαιρώντας τον πλεονασμό συμπιέζεται 3Π 6Ο 2Λ 7Υ 2Μ 8Ε 3Σ 7Α

Αν όμως ο αλγόριθμος αγνοεί τις υψηλότερες συχνότητες εμφάνισης(πχ. Πάνω από το 5) τότε κωδικοποιεί ως εξής: : 3Π 5Ο 2Λ 5Υ 2Μ 5Ε 3Σ 5Α και η πληροφορία μετά την αποσυμπίεση έχει τη μορφή:

ΠΠΠΟΟΟΟΟΛΛΥΥΥΥΜΜΕΕΕΕΕΣΣΣΑΑΑΑΑ

Κωδικοποίηση εντροπίας- κωδικοποίηση πηγής- υβριδική κωδικοποίηση

Μια απλοποιημένη ταξινόμηση των τεχνικών συμπίεσης είναι η εξής: κωδικοποίηση εντροπίας (entropy encoding) και κωδικοποίηση πηγής (source encoding).

Κωδικοποίηση εντροπίας

Η κωδικοποίηση εντροπίας αναφέρεται σε τεχνικές, οι οποίες δεν λαμβάνουν υπ' όψη τους το είδος της πληροφορίας που πρόκειται να συμπιεστεί. Με άλλα λόγια, αυτές οι τεχνικές αντιμετωπίζουν την πληροφορία ως μια απλή ακολουθία bits. Γι αυτό το λόγο, η κωδικοποίηση εντροπίας μπορεί να εφαρμοσθεί ανεξάρτητα από το

είδος της πληροφορίας. Επιπλέον, οι τεχνικές κωδικοποίησης εντροπίας προσφέρουν κωδικοποίηση χωρίς απώλειες.

Η εντροπία θεωρητικά είναι ο ελάχιστος αριθμός από bits ανά σύμβολο που απαιτείται για τη κωδικοποίηση/μετάδοση ενός μηνύματος.

Η εντροπία ενός μηνύματος υπολογίζεται με βάση την εξίσωση Shannon:

$H = -\sum_{i=1}^{n} p_i \log_2 p_i$

Όπου n ο αριθμός των συμβόλων που αποτελούν το μήνυμα και p_i η πιθανότητα εμφάνισης του συμβόλου i.

Επειδή η εντροπία υποδεικνύει τη βέλτιστη συμπίεση (χωρίς απώλειες), που μπορεί να επιτευχθεί, η αποδοτικότητα κωδικοποίησης μιας μεθόδου συχνά συγκρίνεται με την εντροπία. Η αποδοτικότητα κωδικοποίησης υπολογίζεται με βάση τον μέσο αριθμό bits ανά κωδική λέξη (codeword):

Average number of bits per codeword = $\sum_{i=1}^{n} N_i p_i$ όπου n, ο αριθμός των συμβόλων που αποτελούν το μήνυμα, p_i η πιθανότητα εμφάνισης του συμβόλου i και N_i τα bits που χρησιμοποιούνται για τη κωδικοποίηση του συμβόλου i.

Ας δούμε ένα παράδειγμα. Μπορούμε να αντικαθιστούμε κάθε ακολουθία 10 διαδοχικών μηδενικών που βρίσκουμε με ένα ειδικό χαρακτήρα ακολουθούμενο από τον αριθμό 10. Με αυτόν τον τρόπο, μειώνουμε το μήκος της ακολουθίας χωρίς να κάνουμε καμία υπόθεση για την σημασία των μηδενικών, αλλά και χωρίς να αλλοιώνεται το σήμα.

Οι τεχνικές κωδικοποίησης εντροπίας διαχωρίζονται σε δύο βασικές κατηγορίες:

· Τεχνικές Μήκους διαδρομής (Run Length Encoding)

RLC (Run Length Coding)

Zero Suppression

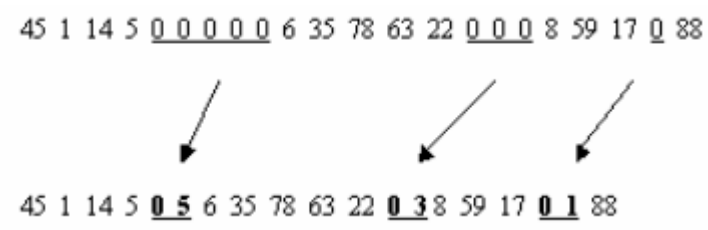
· Στατιστική Κωδικοποίηση (Statistical encoding)

- Huffman
- Αριθμητική
- Αντικατάσταση πρωτοτύπων (π.χ. LZW, LUT)

<u>Κωδικοποίηση μήκους διαδρομής</u>

Η κωδικοποίηση μήκους διαδρομής (RLE - Run Length Encoding)
βασίζεται στην παρατήρηση ότι σε πολλές περιπτώσεις μέσα σε μια ομάδα δεδομένων εμφανίζεται το ίδιο σύμβολο να επαναλαμβάνεται πολλές φορές στη σειρά.

Θα μπορούσε επομένως αυτή η ακολουθία πολλαπλών εμφανίσεων του ίδιου συμβόλου να αντικατασταθεί από δύο άλλα σύμβολα:
(α) το σύμβολο που εμφανίζεται, και
(β) πόσες φορές εμφανίζεται



Αποτελεσματική τεχνική όταν ένας χαρακτήρας επαναλαμβάνεται τουλάχιστον 4 φορές.

Η ίδια ουσιαστικά τεχνική μπορεί να εφαρμοστεί και σε εικόνες όπου η ίδια τιμή φωτεινότητας χρώματος επαναλαμβάνεται πολλές φορές. Η μέθοδος είναι αποτελεσματική για τη συμπίεση κειμένου και εικόνων δυο τόνων (άσπρο- μαύρο)

Στατική κωδικοποίηση

Τρείς βασικές μορφές:

Κωδικοποίηση Huffman:

Τα πιο συχνά χρησιμοποιούμενα σύμβολα κωδικοποιούνται με λιγότερα bits(τα σπάνια εμφανιζόμενα σύμβολα θα έχουν μεγαλύτερου μεγέθους κωδικές λέξεις, ενώ τα συχνά μικρότερου μεγέθους). Η τεχνική βασίζεται σε στατιστικές μεθόδους (πιθανότητες εμφάνισης συμβόλων). Κατασκευάζουμε δυαδικό δέντρο, αρχίζοντας από τους χαρακτήρες με τη μικρότερη πιθανότητα εμφάνισης. Η τεχνική μπορεί να χρησιμοποιηθεί και για κωδικοποίηση ομάδων συμβόλων, όπου η έννοια του «συμβόλου» αντικαθίσταται από εκείνη της «ομάδας συμβόλων». Αντικαθιστούμε τα πρότυπα (packbits encoding)

Ακολουθίες συμβόλων κωδικοποιούνται ομαδικά ως ένας νέο σύμβολο τους με λιγότερα bits. Είναι φανερό ότι η μέθοδος απαιτεί την ύπαρξη λεξικού, όπου αποθηκεύονται οι ακολουθίες που αντιστοιχούν σε κάθε κωδικό για να μπορεί να γίνει η αποσυμπίεση. Το λεξικό προκύπτει από ανάλυση του κειμένου, ενώ κάποιες ακολουθίες συμβόλων είναι εκ των προτέρων γνωστό ότι θα εμφανιστούν σίγουρα.

Αριθμητική κωδικοποίηση:

Αποτελεί την καλύτερη μέθοδο κωδικοποίησης εντροπίας όταν χρειάζεται να πετύχουμε υψηλό ποσοστό συμπίεσης. Το μειονέκτημα της είναι η πολυπλοκότητα της. Ακολουθεί μια σύντομη περιγραφή των βημάτων λειτουργίας της:

1. Υπολογισμός του αριθμού των μοναδικών συμβόλων στην είσοδο. Ο αριθμός αυτός θα αντιπροσωπεύει τη βάση b (π.χ. βάση 2 για το δυαδικό σύστημα) του αριθμητικού κωδικού.
2. Ορίζει τιμές από 0 έως b για κάθε μοναδικό σύμβολο που εμφανίζεται.
3. Ορίζει ένα κωδικό σε αυτή την τιμή.
4. Μετατρέπει το αποτέλεσμα από το βήμα 3 από βάση b σε έναν αρκετά μεγάλο σταθερό δυαδικό αριθμό για να διατηρήσει την ακρίβεια.
5. Καταγράφει το μήκος της ακολουθίας εισόδου κάπου καθώς χρειάζεται στην αποκωδικοποίηση

Αντικατάσταση προτύπων:

Αναζήτηση των πιο συχνά χρησιμοποιούμενων ακολουθιών συμβόλων κι αντικατάστασή τους από ειδικές κωδικές λέξεις (codewords)

Η τεχνική αυτή (pattern substitution) είναι παραλλαγή της τεχνικής «κωδικοποίηση μήκους διαδρομής»

Στην αγγλική, τα πιο συχνά χρησιμοποιούμενα ζεύγη χαρακτήρων είναι τα παρακάτω (προσοχή στους χαρακτήρες κενού):

“E ”, “T ”, “TH”, “ A”, “S ”, “RE”, “IN”, “HE”

Οι ειδικοί χαρακτήρες που αντικαθιστούν τα παραπάνω ζευγάρια δεν πρέπει να εμφανίζονται πουθενά αλλού στο κείμενο. Η τεχνική μπορεί να πετύχει συμπίεση της τάξης του 10%. Ο ευρέως διαδεδομένος αλγόριθμος συμπίεσης LZW (Lempel, Ziv & Welch) βασίζεται στη λογική αντικατάστασης προτύπων.

Κωδικοποίηση Πηγής

Η διαφορά αυτής της τεχνικής είναι ότι οι μετασχηματισμοί τους οποίους υφίστανται το αρχικό σήμα εξαρτώνται άμεσα από το τύπο του. Για παράδειγμα, ο

λόγος χαρακτηρίζεται από συχνά διαστήματα σιωπής, που μπορούν να περιγραφούν με πιο αποτελεσματικό τρόπο. Δηλαδή, οι μετασχηματισμοί του σήματος κάνουν χρήση των ιδιαίτερων σημασιολογικών χαρακτηριστικών που μεταφέρει το σήμα.

Γενικά, αυτές οι τεχνικές μπορούν να παράγουν μεγαλύτερα ποσοστά συμπίεσης σε σχέση με την κωδικοποίηση εντροπίας. Μειονεκτούν όμως στη σταθερότητα, γιατί το ποσοστό συμπίεσης που επιτυγχάνουν διαφοροποιείται ανάλογα με το αντικείμενο που συμπιέζεται. Πάντως, η κωδικοποίηση πηγής μπορεί να λειτουργήσει και με απώλειες και χωρίς απώλειες.

Οι τεχνικές κωδικοποίησης πηγής διακρίνονται σε τέσσερις τύπους:

· Διαφορική ή προβλεπτική κωδικοποίηση (differential or predictive encoding)

Όπως όλες οι τεχνικές πρόβλεψης (predictive techniques), βασίζεται στην πρόβλεψη (με βάση τις προηγούμενες τιμές δειγμάτων) του ποια θα είναι η επόμενη τιμή του δείγματος. Σε περίπτωση μικρών διαφορών δεν έχουμε απώλεια ενώ στην περίπτωση μεγάλων διαφορών χάνεται πληροφορία. Για παράδειγμα η διαφορική κωδικοποίηση μπορεί να χρησιμοποιηθεί για τη συμπίεση των ομοιόμορφων περιοχών των εικόνων.

- DPCM (Differential Pulse Code Modulation)
- DM (Delta Modulation

Ακραία περίπτωση της τεχνικής DPCM στην οποία χρησιμοποιείται μόνο ένα bit ανά δείγμα, υποδηλώνοντας αν το σήμα αυξήθηκε ή μειώθηκε σε σχέση με το προηγούμενο δείγμα.

· Μετασχηματισμού (transform encoding)

Στη κωδικοποίηση μετασχηματισμού, το σήμα υφίσταται ένα μαθηματικό μετασχηματισμό από το αρχικό πεδίο του χρόνου ή του χώρου σε ένα αφηρημένο πεδίο το οποίο είναι πιο κατάλληλο για συμπίεση. Αφού επιλεχθεί και εκτελεστεί ο μετασχηματισμός, βρίσκουμε τους πιο σημαντικούς από τους συντελεστές και τους περιγράφουμε με μεγάλη ακρίβεια. Τους λιγότερο σημαντικούς μπορούμε να τους περιγράψουμε με μικρότερη ακρίβεια ή και να τους αγνοήσουμε τελείως. Αυτή η διαδικασία είναι αντιστρεπτή και έχουμε απώλειες.

- FFT(Fast Fourier Transform)
- DCT (Discrete Cosine Transform)
- Karhunen-Loeve

· Στρωματοποίησης (Layered encoding)

Πρόκειται για διάκριση των αρχικών δεδομένων σε στρώματα (layers) με διαφορετικές ιδιότητες έτσι ώστε κάθε στρώμα να υποστεί διαφορετική επεξεργασία.

- Sub-band Coding

· Διανυσματική κβαντισμός (vector quantization)

Ο διανυσματικός κβαντισμός αποτελεί μια παραλλαγή της μεθόδου αντικατάστασης προτύπων στην οποία όμως επιτρέπονται απώλειες.

- Ταύτισης με προκαθορισμένα πρότυπα
- Fractals

Οι παραπάνω κατηγορίες κωδικοποίησης δεν αποκλείουν η μία την άλλη. Υπάρχουν αλγόριθμοι που συνδυάζουν τεχνικές και των τεσσάρων κατηγοριών για να επιτύχουν καλύτερα αποτελέσματα.

Υβριδική κωδικοποίηση

(JPEG, MPEG-x, H.26x)

Να σημειωθεί ότι οι δυο παραπάνω κατηγορίες κωδικοποίησης δεν αποκλείουν η μια

την άλλη. Υπάρχουν αλγόριθμοι που συνδυάζουν τεχνικές και των δυο κατηγοριών για να επιτύχουν καλύτερα αποτελέσματα

Ασύμμετρη-συμμετρική συμπίεση

Σε αυτόν τον τρόπο κατηγοριοποίησης λαμβάνεται υπόψη η σχετική πολυπλοκότητα του κωδικοποιητή και του αποκωδικοποιητή. Η ασύμμετρη συμπίεση (asymmetric compression) απαιτεί μεγαλύτερο χρόνο κατά τη διάρκεια της κωδικοποίησης παρά της αποκωδικοποίησης. Η συμπίεση γίνεται μία φορά, ενώ η αποσυμπίεση πολλές φορές (βλ. συστήματα ανάκλησης πληροφοριών).

Η συμμετρική συμπίεση (symmetric compression) απαιτεί ίσο χρόνο για κωδικοποίηση και αποκωδικοποίηση. Οι τεχνική αυτή είναι κατάλληλη για σύγχρονες εφαρμογές στις οποίες απαιτείται και τα δύο άκρα να έχουν παρόμοιες δυνατότητες όπως για παράδειγμα σε μια βιντεοδιάσκεψη.

Διορατική συμπίεση(perceptive compression)

Σε μία κωδικοποίηση με απώλειες πρέπει να λαμβάνονται υπόψη το είδος των δεδομένων που συμπιέζονται. Πρέπει να διαγράφονται μόνο τα δεδομένα που η απουσία τους δε θα γίνει αντιληπτή από τις αισθήσεις μας. Ένας κωδικοποιητής που εφαρμόζει τέτοια συμπίεση πρέπει να χρησιμοποιεί αλγορίθμους που βασίζονται στην κατανόηση της ψυχοακουστικής και ψυχοοπτικής αντίληψης.

Κλιμακωτή συμπίεση (cascaded compression)

Οι διαφορές μεταξύ της κωδικοποίησης με απώλειες και χωρίς απώλειες γίνονται εμφανής όταν ακολουθούν πολλαπλές κωδικοποιήσεις.

Προσαρμοστική- μη προσαρμοστική συμπίεση (adaptive-non adaptive compression)

Στην προσαρμοστική συμπίεση είναι μια μέθοδο κωδικοποίησης εντροπίας μη απωλέστικής συμπίεσης δεδομένων . δεν δημιουργεί πιθανοτικό μοντέλο εξετάζοντας πρώτα τα δεδομένα αλλα αρχικοποιεί τα δεδομένα με ένα μοντέλο που έχει συμφωνηθεί και ανάλογα με το τελευταίο σύμβολο το αλλάζει. Προσαρμόζεται δηλαδή στα ρέοντα δεδομένα αντιθέτως με τη μη προσαρμοστική συμπίεση

2.2 Μαθηματικοί τύποι απόδοσης και ορολογία συμπίεσης

Εφόσον έχουμε επιλέξει ένα είδος συμπίεσης δεδομένων, πρέπει να είμαστε σε θέση να μετρήσουμε την απόδοση του.

- Compression ratio (λόγος συμπίεσης): $\frac{\mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma\ \tau\circ\upsilon\ \rho\varepsilon\acute{\upsilon}\mu\alpha\tau\circ\varsigma\ \varepsilon\xi\acute{\circ}\delta\circ\upsilon}{\mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma\ \tau\circ\upsilon\ \rho\varepsilon\acute{\upsilon}\mu\alpha\tau\circ\varsigma\ \varepsilon\iota\sigma\acute{\circ}\delta\circ\upsilon}$

Έστω ότι η αποθήκευση μιας εικόνας απαιτεί 65,536 bytes και η συμπιεσμένη έκδοση του 16,384 bytes. Μπορούμε να πούμε λοιπόν ότι ο λόγος συμπίεσης είναι 4:1 ή 0,75 σύμφωνα με τον παραπάνω τύπο. Αυτό σημαίνει ότι τα δεδομένα καταλαμβάνουν χωρητικότητα 75% της χωρητικότητας που καταλάμβαναν αρχικά .

- Compression factor (παράγοντας συμπίεσης)r=$\frac{\mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma\ \tau\circ\upsilon\ \rho\varepsilon\acute{\upsilon}\mu\alpha\tau\circ\varsigma\ \varepsilon\iota\sigma\acute{\circ}\delta\circ\upsilon}{\mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma\ \tau\circ\upsilon\ \rho\varepsilon\acute{\upsilon}\mu\alpha\tau\circ\varsigma\ \varepsilon\xi\acute{\circ}\delta\circ\upsilon}$

Η μέτρηση αυτή είναι αντίθετη του λόγου συμπίεσης και όσο μεγαλύτερη είναι τόσο καλύτερη συμπίεση πραγματοποιείται. Στο παραπάνω παράδειγμα το αποτέλεσμα θα ήταν 0,25 δηλαδή τα δεδομένα θα καταλάμβαναν 25% λιγότερη χωρητικότητα.

- Bit-rate (Bits ανά pixel/byte/character)

Η απόδοση μετριέται σε bit-rate και παρέχεται ο μέσος αριθμός των bits που απαιτούνται για την συμπίεση ενός pixel ή byte ή character.

- Παραμόρφωση (distortion), Πιστότητα (fidelity), Ποιότητα (quality)

Οι παραπάνω όροι χρησιμοποιούνται για να μιλήσουμε για τη διαφορά μεταξύ των συμπιεσμένων δεδομένων και των αρχικών. Το MSE (mean square error) και το

PSNR (peak signal to noise ratio) είναι μερικές από τις ποσότητες που χρησιμοποιούνται για τη μέτρηση αυτών σε εικόνες και σε ταινίες.

- Κύκλοι ανά byte CPB (cycles per byte)

Η ταχύτητα της συμπίεσης μετριέται με το CPB και είναι ο μέσος αριθμός των μηχανικών κύκλων που χρειάζονται για να συμπιεστεί ένα byte.

- Κέρδος συμπίεσης (compression gain)

$$100\log_e \frac{\mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma\ \alpha\nu\alpha\phi\circ\rho\acute{\alpha}\varsigma}{\sigma\upsilon\mu\pi\iota\varepsilon\sigma\mu\acute{\varepsilon}\nu\circ\ \mu\acute{\varepsilon}\gamma\varepsilon\theta\circ\varsigma}$$

Το μέγεθος αναφοράς είναι είτε το αρχικό μέγεθος του ρεύματος εισόδου είτε το μέγεθος του συμπιεσμένου ρεύματος που παράχθηκε από το κάποια μέθοδο μη απωλεστικής συμπίεσης.

- Calgary Corpus και Canterbury Corpus

Το Calgary Corpus και το Canterbury Corpus είναι συλλογές δυαδικών αρχείων και αρχείων κειμένου που χρησιμοποιούνται για τη σύγκριση αλγόριθμων συμπίεσης δεδομένων. Παρακάτω παρατίθενται τα περιεχόμενα του Calgary Corpus

File	Abbrev	Category	Size
bib	bib	Bibliography (refer format)	111261
book1	book1	Fiction book	768771
book2	book2	Non-fiction book (troff format)	610856
geo	geo	Geophysical data	102400
news	news	USENET batch file	377109
obj1	obj1	Object code for VAX	21504
obj2	obj2	Object code for Apple Mac	246814
paper1	paper1	Technical paper	53161
paper2	paper2	Technical paper	82199
pic	pic	Black and white fax picture	513216
progc	progc	Source code in "C"	39611
progl	progl	Source code in LISP	71646
progp	progp	Source code in PASCAL	49379
trans	trans	Transcript of terminal session	93695

- Μοντέλο πιθανότητας

Το μοντέλο πιθανότητας χρησιμοποιείται στις μεθόδους στατιστικής συμπίεσης δεδομένων και φτιάχνεται πριν ξεκινήσει η συμπίεση. Το ρεύμα εισόδου διαβάζεται μετρώντας τον αριθμό εμφάνισης κάθε συμβόλου και υπολογίζεται η πιθανότητα εμφάνισης κάθε συμβόλου. Στη συνέχεια το ρεύμα εισόδου δεδομένων συμφώνα με τις πληροφορίες του πιθανοτικού μοντέλου που έχει χτιστεί.

2.3 Ανασκόπηση των αλγορίθμων που μελετήθηκαν:

Με βάση αυτά τα είδη κατηγοριοποίησης και εφόσον στοχεύουμε σε embedded systems και δεδομένα αισθητήρων μελετήθηκαν οι παρακάτω αλγόριθμοι και επιλέχθει για υλοποίηση ο LEC (Lossless Entropy Compression)

LZW: Ο αλγόριθμος LZW (Lempel- Ziv- Welch) είναι το αποτέλεσμα μετατροπών του αλγορίθμου LZ77 και LZ78. Λειτουργεί αντικαθιστώντας σειρές χαρακτήρων με κωδικούς. Καθώς οι κωδικοί είναι γενικά μικρότεροι από τις σειρές χαρακτήρων, τα αρχικά δεδομένα συμπιέζονται με την αντικατάσταση σειρών από χαρακτήρες με απλούς κωδικούς. Τα δεδομένα εισόδου δεν υφίστανται καμία ανάλυση απλά

δημιουργείται για κάθε ακολουθία χαρακτήρων ενας κωδικός. Οι κωδικοί είναι αυθέρετου μήκους και στην κωδικοποίηση 8 bit, οι πρώτοι 256 κωδικοί προκύπτουν από το πρότυπο σετ χαρακτήρων. Οι υπόλοιποι παράγονται όταν χρειάζεται.

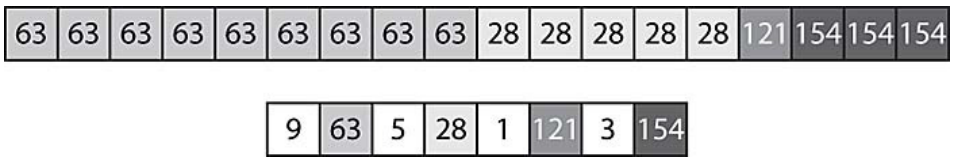
Η συμπίεση που εφαρμόζει είναι χωρίς απώλειες και που χτίζει το λεξικό του καθώς διαβάζονται τα δεδομένα. Ενδείκνυται για δίκτυα αισθητήρων γιατί το λεξικό επιτρέπει την προσπαρμογή των καινούριων δεδομένων και εκμεταλλεύεται την επαναληπτικότητα των δεδομένων. Το μειονέκτημα του σε αυτή τη χρήση είναι οι πόροι επεξεργασίας και μνήμης που χρειάζεται, κάτι που έρχονται να βελτιώσουν οι παραλλαγές του αλγορίθμου που έχουν δημιουργηθεί.

S-LZW-MC: Αποτελεί προσαρμοστική εκδοχή του S-LSW με mini-cache. Διατηρεί μικρό λεξικό, τα δεδομένα προς συμπίεση τα χώριζε σε μικρά blocks. Είναι προσαρμοστικός στις αλλαγές των δεδομένων με τη δημιουργία κάθε νέας σειράς χαρακτήρων μια καινούρια καταχώρηση στο λεξικό. Η δημιουργία συνεχώς καινούριων καταχωρήσεων όμως, κορεννύει το λεξικό. Τα υπόλοιπα δεδομένα για να κωδικοποιηθούν χρησιμοποιείται το λεξικό που έχει δημιουργηθεί ή δημιουργείται ξανά το λεξικό από το μηδέν. Η εκδοχή αυτή του αλγόριθμου LZW, εκμεταλλεύεται και την επαναληπτικότητα των δεδομένων αισθητήρα για να επιτύχει καλύτερη συμπίεση, με μια mini cache. Η mini cache είναι ένα ευρετήριο λεξικό που αποθηκεύει τα δεδομένα που δημιουργήθηκαν και χρησιμοποιήθηκαν πρόσφατα.
Παρόλα αυτά, το λεξικό του δεν διευρύνεται γι αυτό και επιδέχεται βελτίωση η αποτελεσματικότητα συμπίεσης του αλορίθμου.

miniLZO: Αλγόριθμος που προκύπτει από τον LZ77, λαμβάνοντας υπόψη τις απαιτήσεις σε μνήμη επεξεργασίας και μέγεθος κώδικα των ενσωματωμένων συστημάτων. Συμπιέζει αντικαθιστώντας ακολουθίες τριών ή περισσότερων χαρακτήρων με δείκτες από το αρχικό μέρος του αρχείου. Είναι κατά 56% πιο γρήγορος από τν S-LZW αλλα έχει κατά 23,5% χαμηλότερο ποσοστό συμπίεσης.

RLE(Run Length Encoding): Αλγόριθμος χωρίς απώλειες που αντικαθιστά χαρακτήρες με απαρίθμηση χαρακτήρων. Κάθε φορά που βρίσκει 2 ίδιους χαρακτήρες τους αντικαθιστά με δύο παρουσίες του χαρακτήρα και το μήκος τον επαναλήψεων που θα βρεθεί σε αυτές.

Οι επιδόσεις του αλγορίθμου στη συμπίεση των δεδομένων αισθητήρα είναι χαμηλές γι αυτό ούτε αυτος λήφθηκε υπόψη. Είναι κατάλληλος για δομημένα δεδομένα καθώς δε σπαταλά ενέργεια. (σελίδα 13)



Εικόνα 2. Παράδειγμα κωδικοποίησης μήκους διαδρομής.

ALDC (Προσαρμοστικός χωρίς απώλειες σύστημα συμπίεσης δεδομένων - Adaptive Lossless Data Compression):

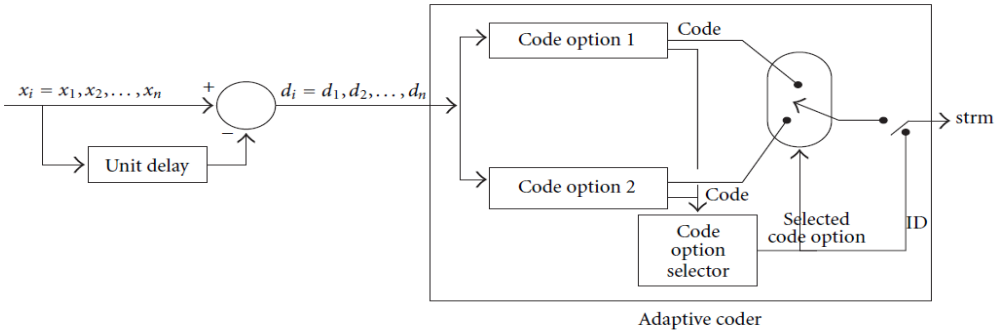
Κάθε σύστημα που προτείνετε για ασύρματα δίκτυα αισθητήρων πρέπει να είναι «ελαφρύ» δηλαδή απαιτείται εκτός από τη μείωση του μεγέθους των δεδομένων, ελάχιστους πόρους για την εκτέλεση της συμπίεσης. Ο αλγόριθμος αυτός επιτρέπει τη συμπίεση χωρίς απώλειες με χρήση πολλαπλών επιλογών κωδικοποίησης, έτσι η κωδικοποίηση προσαρμόζεται στην αλλαγή των στατιστικών δεδομένων της πηγή.

Χρησιμοποιείται είτε η κωδικοποίηση ALEC με 2 πίνακες Huffman είτε η κωδικοποίηση ALEC με 3 πίνακες Huffman. Μελετήθηκα δύο τεχνικές, η τεχνική brute force (ωμής βίας) και η τεχνική decision region(περιοχών απόφασης).

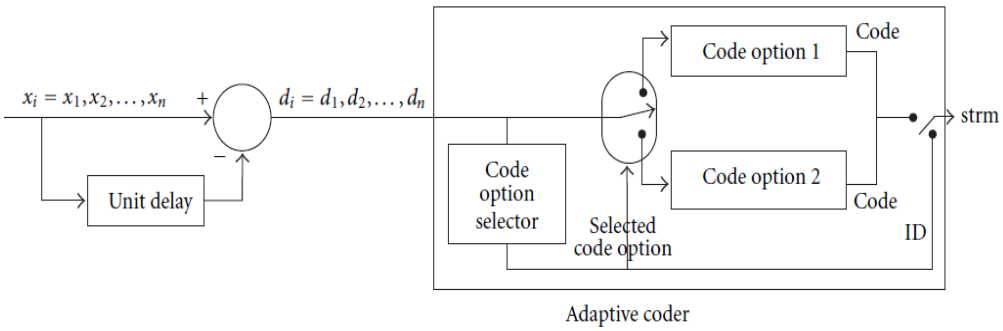
Στην τεχνική brute force ή ωμής βίας, γίνεται κωδικοποίηση και με τα δύο είδη ALEC. Τα αποτελέσματα της κωδικοποίησης συγκρίνονται και επιλέγεται η κωδικοποίηση που επιτυγχάνει το μικρότερο σε μέγεθος αποτέλεσμα.

Επειδή brute force προσέγγιση είναι υπολογιστικά εξαντλητική, χρησιμοποιήθηκε ένας πίνακας απόφασης- για χρήση ALEC με 2 πίνακες ή ALEC με 3 πίνακες- που βασίζεται στο μήκος της θεμελιώδους ακολουθίας των n δειγμάτων. Αυτή είναι η decision regions ή περιοχών απόφασης τεχνική.

Ο λόγος που δεν επιλέξαμε να μελετήσουμε ερευνητικά τον ALDC αλγόριθμο αν και είχε υψηλότερες επιδόσεις από τον LEC είναι γιατί ο LEC χρειάζεται λιγότερη μνήμη. Σε σχέση με τον S-LZW σημειώνουμε ότι ο ALDC χρησιμοποιεί πολύ



Εικόνα 3. Παράδειγμα λειτουργίας του ALDC αλγορίθμου χρησιμοποιώντας την τεχνική ωμής βίας



λιγότερη μνήμη.

Golomb: Τέλος μελετήθηκε ο αλγόριθμος Golomb ο οποίος αποτελεί μια - διαφοροποιημένη εκδοχή του LEC. Η λειτουργία του βασίζεται στο διαχωρισμό των στοιχείων με βάση την εκθετική αύξηση του μεγέθους τους. Κάθε στοιχείο (λέξη κωδικός) είναι ένα υβρίδιο μοναδιαίου(unary) και δυαδικού(binary) αριθμού : Συγκεκριμένα ο unary αριθμός (μεταβλητού μήκους) συγκεκριμενοποιεί την ομάδα, ενώ ο δυαδικός (συγκεκριμένου μήκους αριθμός) αντιπροσωπεύει το δείκτη μέσα στην ομάδα. Η διαφορά του αλγορίθμου LEC με Golomb είναι ότι χρησιμοποιούμε την κωδικοποίηση entropy αντί κωδικοποίηση unary (μονοτελής) διότι η unary κωδικοποίηση απαιτεί το εύρος των αριθμών που εκπροσωπούνται να είναι προκαθορισμένο. Αυτό απαιτείται γιατί το εύρος αυτό καθορίζει τον αριθμό των bits που χρησιμοποιούνται.

Ο αλγόριθμος LEC είναι απλός, απαιτεί λίγη μνήμη για την εκτέλεση του. Έχει μικρή υπολογιστική πολυπλοκότητα και δίνει μέχρι σήμερα την καλύτερη συμπίεση χωρίς απώλειες. Το μόνο που δε μπορούσε να κάνει είναι να προσαρμοστεί στην αλλαγή συσχετισμού των δεδομένων από τον αισθητήρα.

Κεφάλαιο 3

3.1 Σχεδίαση και ανάπτυξη του αλγορίθμου LEC σε C με έμφαση στα ενσωματωμένα συστήματα

Στην παρούσα διπλωματική εργασία υλοποιήθηκε ο αλγόριθμος συμπίεσης LEC(Lossless Entropy Compression) σε ιατρικά σήματα. Η ανάπτυξη της λειτουργικότητας του κώδικα έγινε σε C ενώ για επαλήθευση έχει εφαρμοστεί ο αλγόριθμος σε MATLAB. Μελετήσαμε την εφαρμογή του αλγορίθμου σε γλώσσα C τόσο στην τυπική της μορφή όσο και σε μια απλουστευμένη εκδοχή(χωρίς πολλούς pointers, malloc και ορισμό αρχείων) για να καλύπτονται οι ανάγκες των μικροελεγκτών των ενσωματωμένων συστημάτων.

3.2 Περιβάλλον ανάπτυξης και δομή του project μέσα σε αυτό.

Το περιβάλλον ανάπτυξης είναι το Microsoft Visual Studio Community version 14.0.25424.00. Το Microsoft Visual Studio Community είναι ένα πλήρως εξοπλισμένο, επεκτάσιμο, δωρεάν περιβάλλον ανάπτυξης (IDE) για τη δημιουργία σύγχρονων εφαρμογών και προγραμμάτων για Android, iOs, Windows, καθώς και web εφαρμογών και cloud υπηρεσιών. Το Visual Studio Community είναι μια από τις πέντε εκδοχές του Visual Studio IDE. Το Visual Studio Community είναι κατάλληλος για μικρές ομάδες ή ομάδες καινούριες σε αυτό. Οι μεγαλύτερες ομάδες θα χρειαστούν την άδεια είτε για το Professional είτε για το Enterprise Microsoft Developers Network (MSDN) που κοστίζουν περίπου $1,999 ή $5,999. Ο προγραμματιστές μπορούν να επιλέξουν τη γλωσσα ανάπτυξης της εφαρμογής τους στο Visual Studio μέσα από μια πλούσια επιλογή γλωσσών(C, C#, Visual Basic, F#, C++, JavaScript, TypeScript, Python και άλλες) Το Visual Studio καθοδηγεί τους προγραμματιστές καθώς γράφουν, αποσφαλματώνουν και δοκιμάζουν τον κώδικα τους σε οποιαδήποτε γλώσσα επιλέξουν για την ανάπτυξη του. Βοηθάει στην εύρεση και στη διόρθωση ζητημάτων του κώδικα αλλά και στην ανακατασκευή(refactor) γρήγορα και με ευκολία. Περιλαμβάνει ένα επεξεργαστή κώδικα που υποστηρίζει το IntelliSense, ένα component συμπλήρωσης κώδικα.

Αρχικά ορίστηκαν οι βιβλιοθήκες που θα χρησιμοποιηθούν και καποιες εκ των μεταβλητών (sampling rate, delaytoinit, noftimecounts). Ο κώδικας δομείται από δύο συναρτήσεις τη main και τη συνάρτηση int162lec. Μέσα στη Main γίνεται αρχικοποίηση των μεταβλητών και των πινάκων που θα χρησιμοποιηθούν και ακολουθούν κάποιοι επαναληπτικοί βρόγχοι. Ο πρώτος βρόγχος δημιουργεί τη διαφορά του πίνακα των στοιχείων προς κωδικοποίηση που θα γίνει η είσοδος στον κωδικοποιητή εντροπίας. Ο δεύτερος βρόγχος καλεί τη συνάρτηση int162lec που αναλαμβάνει την κωδικοποίηση εντροπίας για κάθε στοιχείο. Στη συνέχεια μέσα σε αυτόν, χωρίζονται τα κωδικοποιημένα στοιχεία σε blocks των 32 bit, μετατρέπονται σε δεκαδική και little endian Μορφή και γράφονται σε αρχείο. Τέλος μέσα σε αυτόν τον βρόγχο γίνεται και η συμπίεση των στοιχείων, σε ρυθμό που έχει οριστεί, η

μέτρηση του χρόνου(χρόνος cpu) συμπίεσης. Στον επόμενο βρόγχο, γίνεται πάλι η διαδικασία για τα εναπομείναντα στοιχεία.

Στη συνάρτηση int162lec γίνεται η κωδικοποίηση εντροπίας. Αντιστοιχίζετε κάθε στοιχείο σε μία ομάδα που αντιπροσωπεύεται από μια ακολουθία 0,1 και βρίσκεται πόσα bit χρειάζονται(n) για να αντιπροσωπευθεί ο αριθμός.

3.3 LEC (Lossless Entropy Compression) algorithm

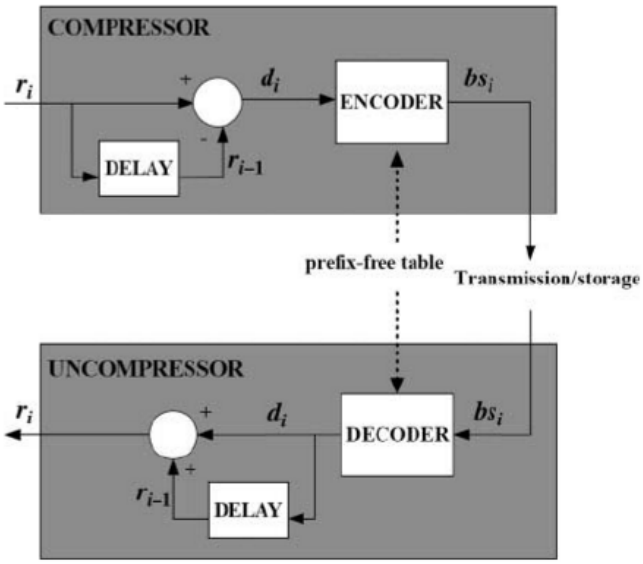
Ο αλγόριθμος LEC (Lossless Entropy compression, Μη απωλεστική συμπίεση εντροπίας) είναι μια αποτελεσματική μέθοδος, που χρειάζεται λίγη υπολογιστική ισχύ και μικρό λεξικό. Ουσιαστικά αποτελεί μια διαφοροποιημένη έκδοση της Exponential- Golomb Κωδικοποίησης 0 τάξης. Δεν υπάρχει απώλεια δεδομένων, δηλαδή στην αποκωδικοποίηση τα δεδομένα μας είναι ίδια με τα αρχικά. Είναι αποτελεσματικός όταν οι διαφορές στα συνεχόμενα δείγματα είναι μικρές, το οποίο συμβαίνει όταν το σήμα δεν είναι ομαλό.

Η λειτουργία της βασίζεται στο διαχωρισμό των στοιχείων με βάση την εκθετική αύξηση του μεγέθους τους. Όπως στην κωδικοποίηση Golomb, κάθε στοιχείο (λέξη κωδικός) είναι ένα υβρίδιο μοναδιαίου(unary) και δυαδικού(binary) αριθμού : Συγκεκριμένα ο unary αριθμός (μεταβλητού μήκους) συγκεκριμενοποιεί την ομάδα, ενώ ο δυαδικός (συγκεκριμένου μήκους αριθμός) αντιπροσωπεύει το δείκτη μέσα στην ομάδα.

Εδώ εμείς χρησιμοποιούμε κωδικοποίηση entropy αντί κωδικοποίηση unary (μονοτελής) διότι η unary κωδικοποίηση απαιτεί το εύρος των αριθμών που εκπροσωπούνται να είναι προκαθορισμένο. Αυτό απαιτείται γιατί το εύρος αυτό καθορίζει τον αριθμό των bits που χρησιμοποιούνται.

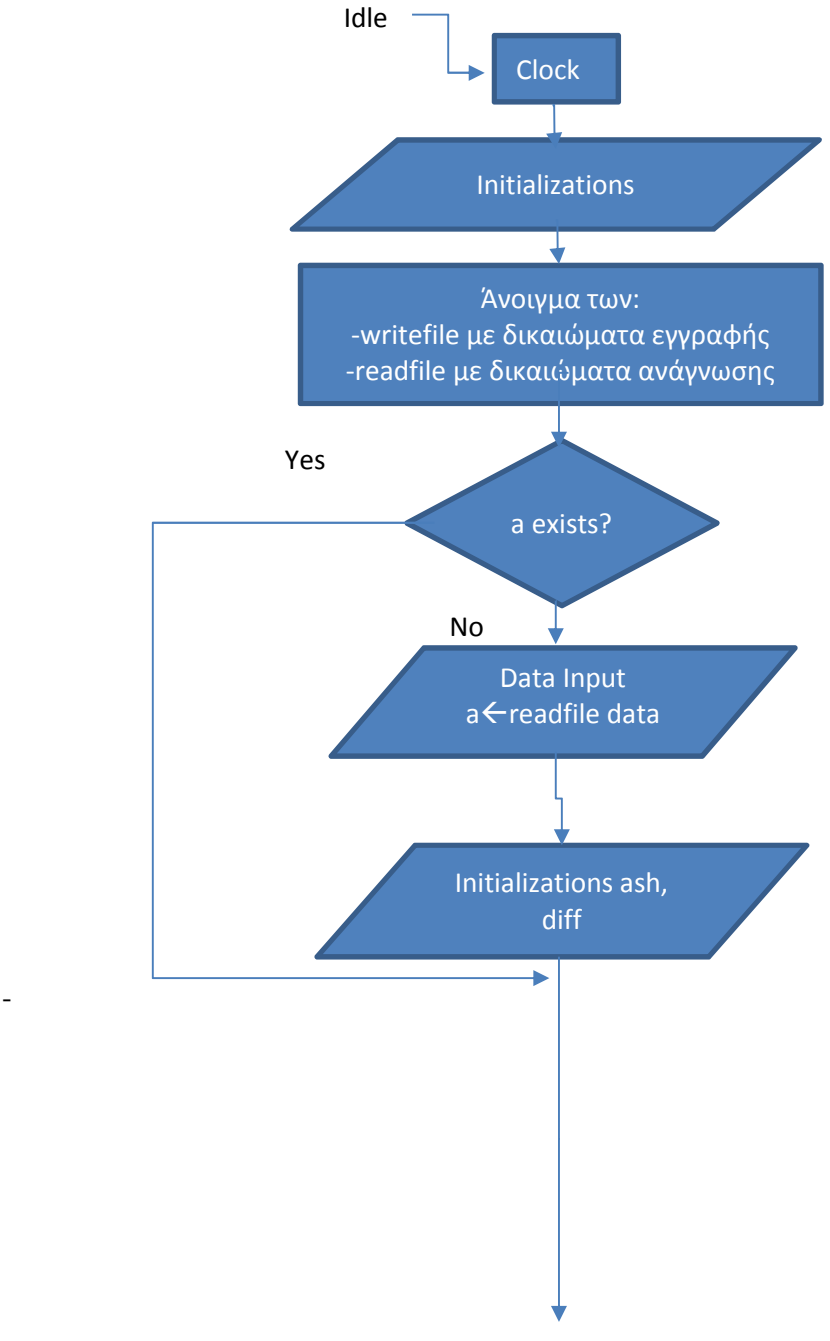
3.4 Βήματα λειτουργίας του κώδικα σε MATLAB

Το παρακάτω διάγραμμα δείχνει τα βασικά βήματα της λειτουργίας του αλγορίθμου σε MATLAB.

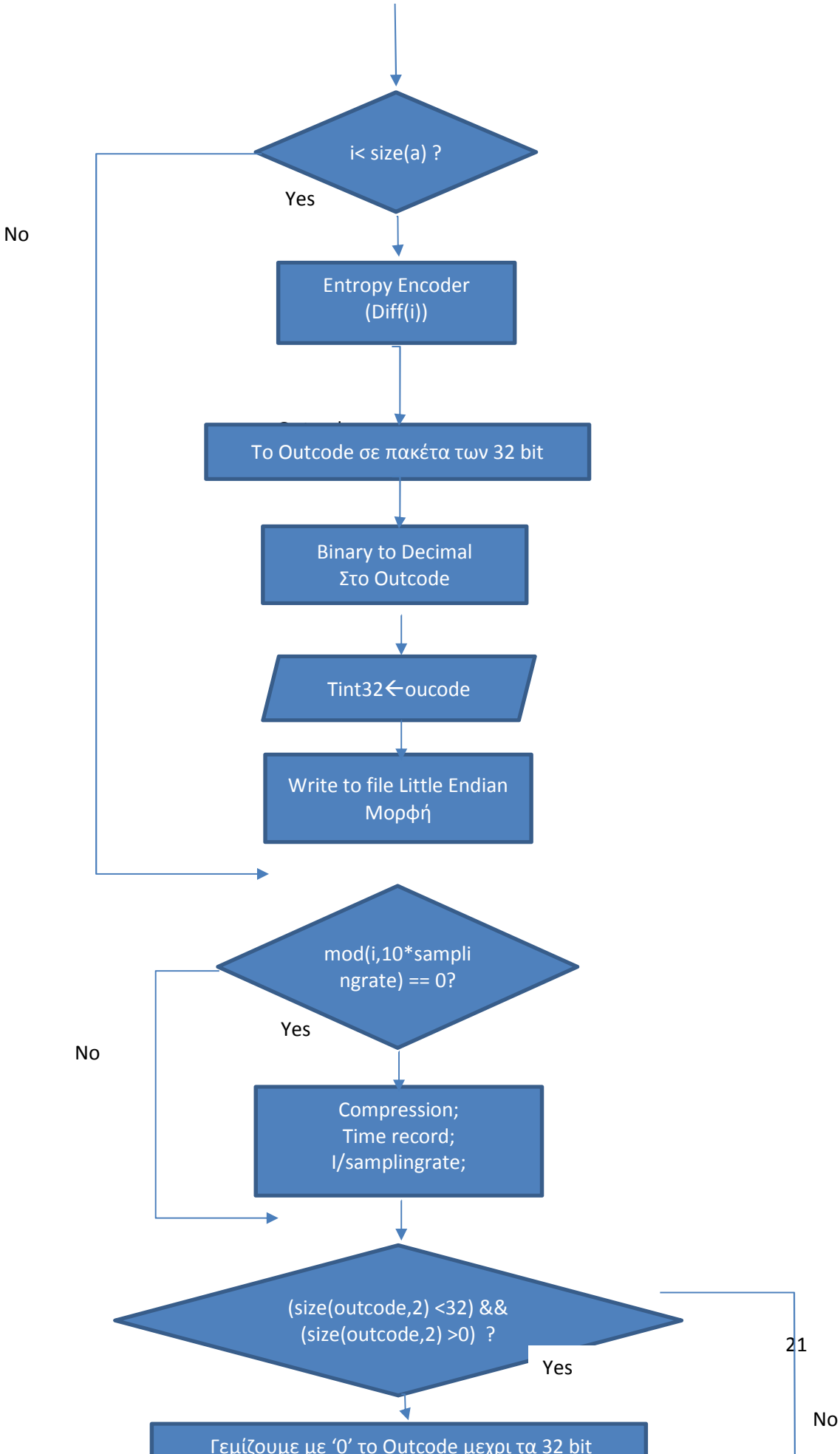


Εικόνα 5. Σχήμα λειτουργίας του κωδικοποιητή και αποκωδικοποιητή εντροπίας

- Το σήμα προς συμπίεση αρχικά φορτώνεται από αρχείο binary.
- Αποθηκεύεται σε πίνακα σε μορφή 16δική και Little Endian
- Για κάθε στοιχείο υπολογίζει τη διαφορά di = ri-ri-1 η οποία είναι και η είσοδος του entropy encoder
- Entropy encoder: Αντιστοιχεί κάθε στοιχείο σε μία ομάδα που αντιπροσωπεύεται από μια ακολουθία 0,1 και βρίσκουμε πόσα bit χρειάζονται(n) για να αντιπροσωπεύσουμε τον αριθμό
- Χωρίζουμε τα αποτελέσματα του entropy encoder σε πακέτα των 32 bit σε Little Endian μορφή
- Ορίζουμε κάθε πότε θα εφαρμόζουμε συμπίεση
- Εφαρμόζουμε συνάρτηση συμπίεσης , μετράμε χρόνο συμπίεσης
- Αποθήκευση αποτελεσμάτων σε αρχείο και σχεδιασμός διαγράμματος

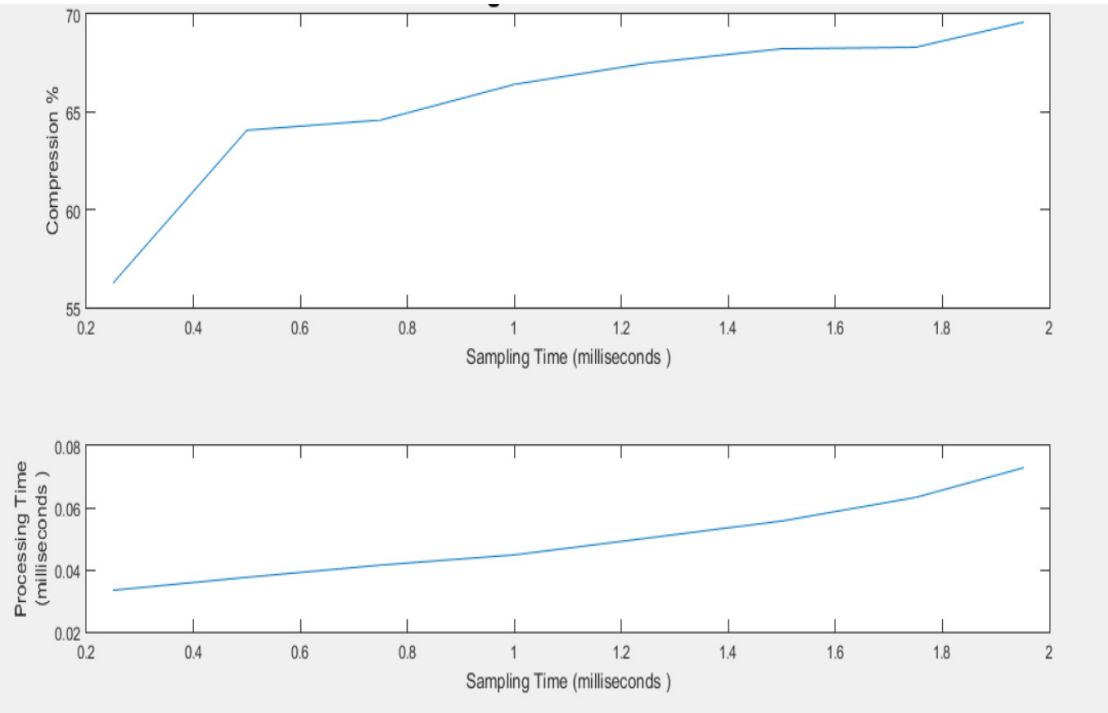


-


i< size(a) ?
Yes
No
Entropy Encoder
(Diff(i))
Το Outcode σε πακέτα των 32 bit
Binary to Decimal
Στο Outcode
Tint32←oucode
Write to file Little Endian
Μορφή
mod(i,10*sampli
ngrate) == 0?
Yes
No
Compression;
Time record;
I/samplingrate;
(size(outcode,2) <32) &&
(size(outcode,2) >0) ?
Yes
No
21

3.5 Αποτελέσματα MATLAB

Αρχικά η συμπίεση πραγματοποιήθηκε στο περιβάλλον του MATLAB. Για κάθε ιατρικό σήμα συλλέχτηκαν διαγράμματα ρυθμού και χρόνου συμπίεσης.



Εικόνα 6. Πάνω: Ποσοστό συμπίεσης σε σχέση με το χρόνο δειγματοληψίας στο MATLAB για το σήμα ***eegchb070707mit*** *Κάτω: Χρόνος συμπίεσης σε milliseconds, σε σχέση με το χρόνο δειγματοληψίας.*

Τα αποτελέσματα του ρυθμού συμπίεσης για το σήμα eegchb070707mit κυμαίνονται στο 56-69%. Το αρχικό σήμα είναι μεγέθους 307200 bytes και γίνεται 103680 bytes.

Ακολουθεί ενδεικτικά, ο ρυθμός συμπίεσης για κάποια sampling rates του σήματος:

CompressionTimes1	CompressionTimes3 (Sampling rate)
56,25	0,25
64,06	0,5
64,58	0,75
66,40	1
67,5	1,25
68,22	1,50
68,30	1,750
68,33333	1,875
68,75	2,5
69	3,125

68,75	3,75
68,39286	4,375
67,96875	5

3.6 Απαιτήσεις μετατροπής MATLAB→C

Η μετατροπή του κώδικα από matlab σε γλώσσα C δεν είναι μια τυπική μετατροπή. Το matlab διαθέτει πλούσιες βιβλιοθήκες(λογικές και μαθηματικές) ένω η c είναι χαμηλότερου επιπέδου γλώσσα. Μερικές από τις προσαρμογές που χρειάζονται να γίνουν αφορούν τα παρακάτω:

- Κατανομή της μνήμης. Η κατανομή της μνήμης στο MATLAB είναι αυτόματη. Σε κώδικα C, η κατανομή της μνήμης είναι χειροκίνητη και κατανέμετε είτε στατικά (χρησιμοποιώντας static) είτε δυναμικά (χρησιμοποιώντας malloc), ή στη στοίβα (χρησιμοποιώντας τοπικές μεταβλητές)
- Το MATLAB είναι γλώσσα βασισμένη σε arrays και παρέχει ένα πλούσιο σύνολο λειτουργιών πίνακα που επιτρέπουν τη συνοπτική κωδικοποίηση των αριθμητικών αλγορίθμων. Στη C απαιτούνται βρόγχοι επανάληψης for για να εκφραστεί ο ίδιος αλγόριθμος.
- Δυναμική πληκτρολόγηση. Το MATLAB καθορίζει αυτόματα όπως τρέχει ο κώδικας του τύπους και το μέγεθος των μεταβλητών. Η C απαιτεί ρητή δήλωση του τύπου και της λειτουργίας όλων των μεταβλητών.
- Πολυμορφισμός. Οι συναρτήσεις του MATLAB μπορούν να υποστηρίξουν πολλούς τύπους εισόδου, ενώ η C απαιτεί δηλώσεις σταθερού τύπου.

Κεφάλαιο 4

4.1 Κώδικας σε C

Η μεταφορά του κώδικα MATLAB σε C έγινε με δύο τρόπου. Αρχικά ανέπτυξα τον αλγόριθμο σε μια τυπική μορφή C (με pointers, malloc και ορισμούς αρχείων) για λόγους debugging αλλά και γιατί καλύπτεται η ανάγκη για μερικούς επεξεργαστές που χρησιμοποιούν την τυπική μορφή της γλώσσας. Στη συνέχεια, για να καλυφθεί η ανάγκη των υπόλοιπων επεξεργαστών, ανέπτυξα τον κώδικα σε μια απλούστερη μορφή χωρίς malloc και ορισμούς αρχείων, με λιγότερους pointers και με τη χρήση στατικών πινάκων.

4.2 Ανάλυση του κώδικα μορφής α'

Γενικά βήματα που ακολουθήθηκαν

- Αρχικοποιώ μεταβλητές και Pointers μεταβλητών και πινάκων.
- Βρίσκω το μέγεθος του αρχείου
- Ορίζω θέσεις στη μνήμη για τους πίνακες
- Κάνω μετατροπή ανα 16 bit σε Little Endian
- Ορίζω τους πίνακες προς συμπίεση.
- Κάνω κωδικοποίηση εντροπίας. Αποδίδω σε κάθε στοιχείο μεγέθους έως 16 bit μια σειρά από 0 και 1 προκαθορισμένη ανάλογη με το μέγεθος του στοιχείου. Πχ if (ab>15 && ab <= 31) { strcpy(bitret, "110") }

- Τα αποτελέσματα της κωδικοποίησης τα πακετάρω ανα 32 bit και μετατρέπω σε Little Endian μορφή.

```
CP[0] = (tint32 & 0x000000ffUL);
CP[1] = (tint32 & 0x0000ff00UL) >> 8;
CP[2] = (tint32 & 0x00ff0000UL) >> 16;
CP[3] = (tint32 & 0xff000000UL) >> 24;
```

- Όπως και στο MATLAB στη συνέχεια συμπιέζω, καταγράφω το χρόνο και τον αριθμό του δείγματος.

```
CompressionTimes1[timecount] = (double) 100.0*(1.0 - ((double)nofbytes + (double)delaytoinit / 2.0) / (((double)i + 1.0)*2.0));

check_t = clock();
CompressionTimes2[timecount] = ((float)(check_t - start_t) / 1000000.0F) * 1000;
CompressionTimes3[timecount] = ((double)i + 1.0) / ((double)SamplingRate);
```

Αποτελέσματα

Τα αποτελέσματα συμπίεσης είναι 56-67% περίπου. Είναι ίδια με τα αποτελέσματα που είχαμε από τον κώδικα MATLAB. Συγκεκριμένα Το αρχικό σήμα είναι μεγέθους 307200 bytes και γίνεται 103680 bytes. Ακολουθούν τα αποτελέσματα για κάποια δείγματα

CompressionTimes1	CompressionTimes3 (Sampling rate)
56	0.23
57.14	0.27
63.07	0.507
64.21	0.7421
66.92	1.01
67,50	1.25
68.20	1.52
68.44	1.757
68.33	1.875
68,75	2.5
69,00	3.125
68,75	3.75
68,39	4.375

4.3 Ανάλυση του κώδικα μορφής β’

Τα βήματα που ακολουθούνται για την υλοποίηση του κώδικα της μορφής β’ είναι τα ίδια με της μορφής α’ εκτός κάποιων λεπτομερειών. Γενικά σε αυτή τη μορφή η ανάπτυξη του κώδικα είναι απλουστευμένη χωρίς πολλούς pointers, malloc και ορισμό αρχείων. Ενδεικτικά έχω εκχωρήσει σε πίνακα 1000 byte, τα πρώτα 1000 byte δεδομένων του σήματος προς επεξεργασία. Οτιδήποτε αποθηκεύεται σε αρχείο στη μορφή α’ καταχωρείτε πλέον σε πίνακα.

Αποτελέσματα

Τα αποτελέσματα συμπίεσης είναι 50-70% περίπου σε σήμα 1000 bytes. Ακολουθούν τα αποτελέσματα για τα πρώτα 13 δείγματα

CompressionTimes1	CompressionTimes3 (Sampling rate)
50.000	0.125000
59.375000	0.25000
62.5000	0.37500
65.625000	0.500
65.000	0.62500
65.625000	0.75000
66.964286	0.87500
67.187500	1.000
68.055556	1.125000
68.125000	1.25000
68.181818	1.37500
68.75000	1.500
68.75000	1.62500

Κεφάλαιο 5
Κρίσιμα Τεχνικά Θέματα
5.1 Low level programming - Little Endian and Bitwise Operations

Για να γίνει ο κώδικας κατάλληλος για μικροελεγκτές, γράφτηκε με αρκετές πινελιές low-level programming. Για να έχουμε τη ακριβή μετατροπή από γλώσσα MATLAB σε C ενασχολήθηκα με το αρχείο σε bit level.

Για την ανάγνωση και την εμφάνιση στοιχείων χρησιμοποιούνται specifiers τύπου x (unsigned hexadecimal integer) .

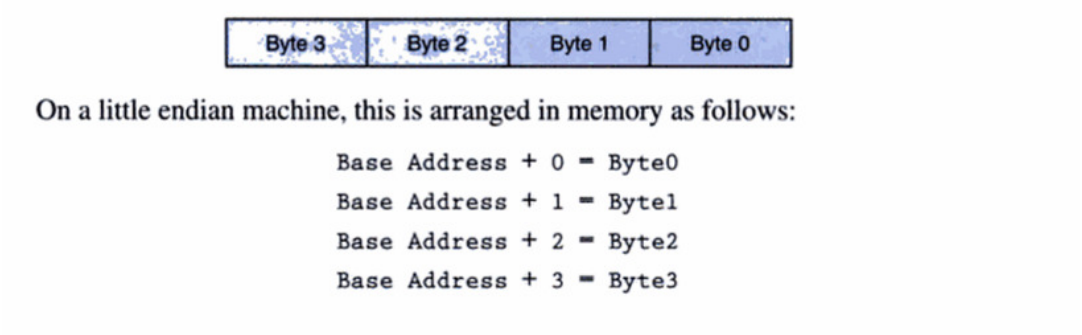
Η μετατροπή των bits απο Big Endian Little Endian είναι και αυτή μια από τις μετατροπές που γίνονται στο MATLAB πολύ απλα. Για να ταξινομηθούν τα bits ενός αρχείου σε μορφή little endian αρκεί 'l' or 'ieee-le'. Στη γλώσσα c από την άλλη η διαδικασία για να αλλάξει το endianess δεν είναι τόσο απλή, και τα byte μετατρέπονται με τη χρήση bitwise shift operators.

```
CP[0] = (tint32 & 0x000000ffUL);
CP[1] = (tint32 & 0x0000ff00UL) >> 8;
CP[2] = (tint32 & 0x00ff0000UL) >> 16;
CP[3] = (tint32 & 0xff000000UL) >> 24;
```

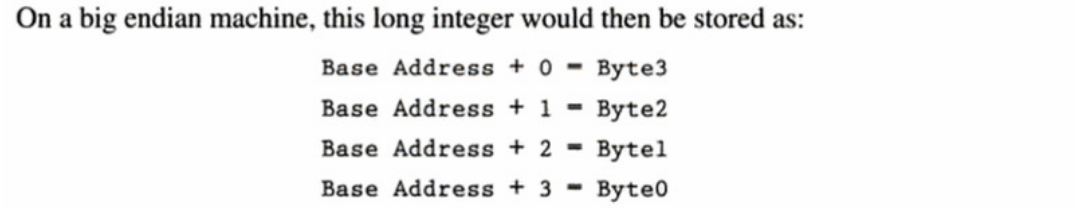
Τι είναι όμως το Endianess;

Ο όρος "endian αναφέρεται στη «σειρά» των bytes στην αρχιτεκτονική ενός υπολογιστή. Ή όταν ένα στοιχείο δεδομένων περιέχει περισσότερα από ένα byte, υπάρχουν δύο λογικοί τρόποι να αποθηκευτούν αυτά στη μνήμη :

- Big Endian: Τα byte αποθηκεύονται στη «φυσική» τους σειρά (το πιο αριστερό Byte έρχεται πρώτο)
- Little- Endian: Τα byte αποθηκεύονται στην αντίστροφη σειρά(το πιο αριστερό Byte έρχεται τελευταίο)



Εικόνα 7. Little Endian μορφή



Εικόνα 8.Big Endian μορφή

Little Endian versus Big Endian

Η κάθε μέθοδος έχει τα πλεονεκτήματα και τα μειονεκτήματα της, χωρίς απαραίτητα κάποια από τις δύο να υπερτερεί της άλλης. Το Big Endian μέθοδος είναι πιο φυσική στους περισσότερους ανθρώπους και γι' αυτό είναι ευκολότερη η ανάγνωση hex. Η θέση του υψηλότερου byte είναι η πρώτη, επιτρέποντας μας έτσι να τεστάρουμε εάν ο αριθμός είναι θετικός ή αρνητικός κοιτάζοντας το byte στο offset zero(Σε σύγκριση με το little endian που πρέπει να γνωρίζεις το μήκος του αριθμού και μετα να παρακάμψεις bytes για να βρεις το byte με την πληροφορία του προσήμου. Στην αποκωδικοποίηση δεδομένων σε σχήματα όπως Huffman και LZW, η πραγματική λέξη-κωδικός μπορεί να χρησιμοποιηθεί ως δείκτης στον πίνακα παραπομπής αν αποθηκευτεί σε big endian.

Παρόλα αυτά η big endian μορφή έχει και τα μειονεκτήματα της. Η μετατροπή από 32-bit ακέραια διεύθυνση σε 16-bit ακέραια διεύθυνση απαιτεί big endian μηχάνημα για να γίνει η πρόσθεση. Οι πράξεις με υψηλή αριθμητική ακρίβεια είναι πιο γρήγορες και εύκολες σε μηχάνημα little endian. Οι περισσότερες αρχιτεκτονικές που χρησιμοποιούν το Big endian σχήμα δεν επιτρέπουν οι λέξεις να γράφονται εκτός των

ορίων της διεύθυνσης της λέξης με αποτέλεσμα να σπαταλάτε χώρος. Η μορφή Little endian είναι γενικά πιο εύχρηστη στην ανάγνωση και γραφή.

ΟΙ σχεδιαστές λογισμικού αλλά κάθε πρόγραμμα που γράφει ή διαβάζει δεδομένα από αρχείο πρέπει να έχει επίγνωση της σειράς των bytes στο συγκεκριμένο πρόγραμμα ή μηχάνημα αντίστοιχα. Για παράδειγμα, το GIF είναι little endian, το JPEG big endian, το PC paintbrush little endian. Μερικές εφαρμογές υποστηρίζουν και τα δύο format, όπως για παράδειγμα τα Microsoft WAV και AVI αρχεία, τυπικά ορίζοντας έναν αναγνωριστικό μέσα στο αρχείο.

5.2 Carriage return and Line feed

Άλλη μια διαφορα μεταξύ matlab και C σε bit level, είναι ο χαρακτήρας hex 0D (carriage return) που εμφανίζεται πρίν τον hex 0A(line feed).

Το Enter key ερμηνεύεται διαφορετικά από μια πλατφόρμα σε άλλη:

- Τα Windows χρησιμοποιούν carriage return και line feed.
- Τα Unix μόνο line feed και
- τα Macintosh μόνο carriage return

Στην περίπτωση μας, η διαφορά αυτή οφείλεται σε διαφορά εντολών που αναλαμβάνουν το γράψιμο του αρχείου στη C και στο MATLAB. Στο MATLAB το Enter key γράφεται μόνο με Line feed (0A) ενώ στη C επειδή το αρχείο που γράφεται είναι text, το Enter key ερμηνεύεται ως carriage return και line feed(0D 0A). Για να έχουμε συνεπώς τα ίδια αποτελέσματα εξαλείψαμε το carriage return.

Κεφάλαιο 6

6.1 Αξιολόγηση της υλοποίησης του αλγορίθμου

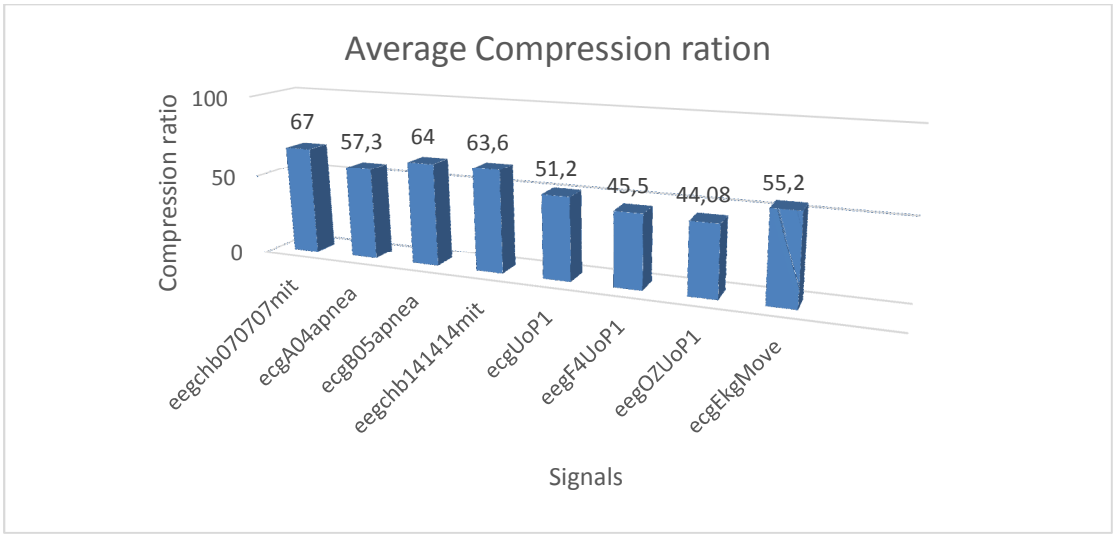
Στην παρούσα διπλωματική εργασία υλοποιήθηκε ο αλγόριθμος συμπίεσης LEC(Lossless Entropy Compression) σε ιατρικά σήματα. Η επιλογή του αλγορίθμου αυτού εξυπηρετεί τη συμπίεση των ιατρικών δεδομένων καθώς εκεί οι απώλειες πληροφοριών δεν είναι επιθυμητές. Επιπλέον, ο αλγόριθμος αυτος είναι αποτελεσματικός ενώ παράλληλα υλοποιείται σε λίγες γραμμές κώδικα. Η ανάπτυξη του έγινε σε C, γλώσσα κατάλληλη για τον προγραμματισμό των μικροελεγκτών των ενσωματωμένων συστημάτων. Για επαλήθευση έχει εφαρμοστεί ο ίδιος αλγόριθμος σε MATLAB. Το προγραμματιστικό περιβάλλον ήταν το Visual studio και το Keil uVision 5.

Μελετήσαμε την εφαρμογή του αλγορίθμου σε γλώσσα C τόσο στην τυπική της μορφή όσο και σε μια απλουστευμένη εκδοχή για να καλύπτονται οι ανάγκες των μικροελεγκτών των ενσωματωμένων συστημάτων.

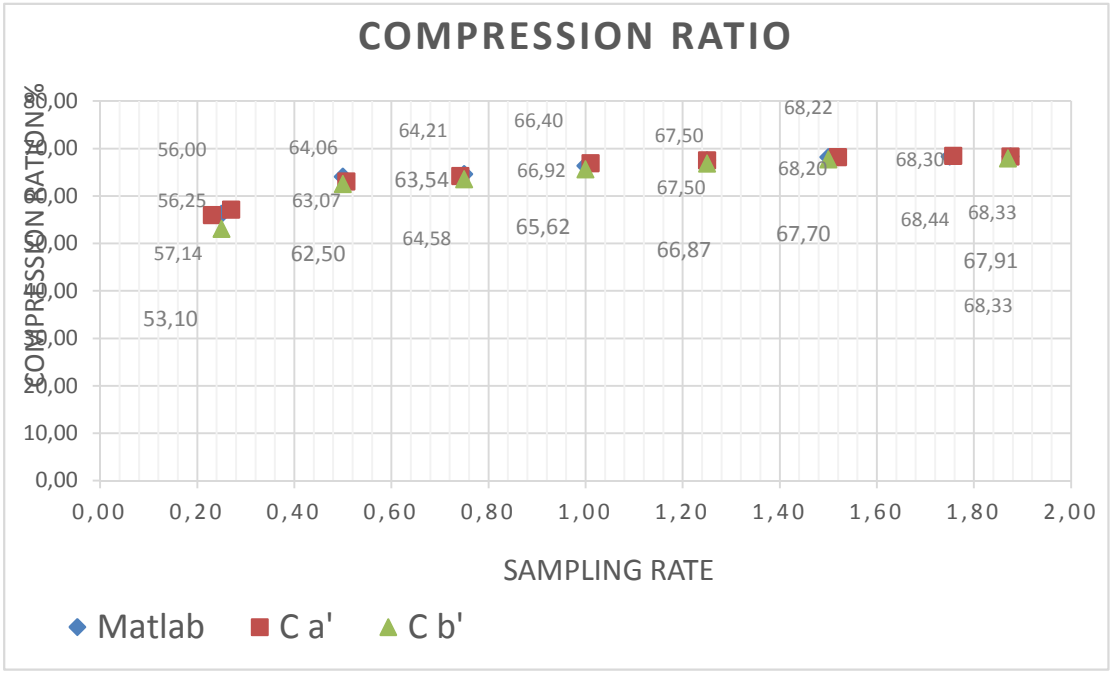
Στον πίνακα που ακολουθεί φαίνεται το αποτέλεσμα των συμπιεσμένων δεδομένων ,ενδεικτικά όπως προηγουμένως, για το σήμα eegchb070707mit σε δεκαεξαδική μορφή. Καταφέραμε να έχουμε ίδια δεδομένα κατά την κωδικοποίηση όλων των σημάτων που επεξεργαστήκαμε σε όλες τις εκδοχές της C που την εφαρμόσαμε.

```
00000040 57 8E B0 59 91 5C 23 A0  49 36 32 84 87 95 44 C7  W.°Y‘\# I62„‡•DH
00000050 2C DD 59 B6 5A B2 39 71  64 E3 F8 AC 8E E2 0B 89  ,éY¶Z²9qdγψ¬.β.‰
00000060 77 2D 49 D2 BE AA 06 32  9C 58 7A 45 85 0D 04 71  w-I.Ÿ..2.XzE…..q
00000070 8B 01 82 AD D7 B2 23 71  11 1B 90 58 1A 99 AF 2C  ‹.‚-X²#q...X.™–,
00000080 22 0B C7 F6 58 23 8A 62  52 45 49 3E 62 AB 06 59  ".HφX#.bREI>b«.Y
00000090 AC B2 6C 44 AE 91 C8 88  23 E7 1C 04 42 C2 3D 5A  ¬²lD®‘Θ.#η..BB=Z
000000A0 23 F7 48 D7 46 08 07 11  D5 39 92 84 1A 6C 1C DB  #χHXF...Y9’„.l.Ÿ
000000B0 AE 1A 35 88 6B 7C 45 92  0B C4 E7 28 CD 91 E6 23  ®.5.k|E’.Δη(N‘ζ#
000000C0 32 B1 DA 48 74 E4 98 38  09 23 72 AD F2 7C B8 87  2±ÏHtδ.8.#r-ς|Ë‡
000000D0 8C B1 46 BA 39 A8 11 45  B1 90 88 1C 1C 55 E1 88  .±FI9¨.E±....Uα.
000000E0 C5 60 23 35 A8 41 45 16  74 AE C0 F6 6B 50 78 85  E`#5¨AE.t®ïφkPx…
000000F0 24 0B C7 9A 5A 2B 4F B3  48 6C 22 C8 5A 93 A4 90  $.H.Z+O³Hl"ΘZ“¤.
00000100 E4 83 F6 B0 23 AD 35 78  91 B5 1A 8A 66 1F F1 E8  δƒφ°#-5x‘μ..f.ρθ
00000110 1D A5 89 CD C4 8A 9E 33  A4 F1 39 72 38 86 E3 75  .¥‰NΔ..3¤ρ9r8†γu
00000120 C8 07 AD D1 44 49 BA E7  C8 46 36 8E E9 DC 2B 9A  Θ.-PDIΪηΘF6.ιά+.
00000130 AD 18 55 55 6B AE 5A 89  DB 2B D7 1A 4A D3 51 0B  -.UUk®Z‰Ÿ+X.JΣQ.
00000140 99 E9 DE 2B 8A 23 D1 23  73 25 32 36 95 25 F9 9E  ™ιή+.#P#s%26•%ω.
00000150 07 8D AA 5C 8D 63 28 BA  2C 1B B1 D1 23 04 B2 E8  ...\.c(Ί,.±P#.²θ
00000160 28 86 60 5D 12 54 92 A2  F5 9C 73 34 24 A1 28 8E  (†`].T’Άυ.s4$¨(.
00000170 F7 DE 08 94 34 5F C1 4A  C5 90 3D 92 93 2C 07 5E  χή.”4_AJE.=’“,.^
00000180 3E 57 50 B9 AF 24 DF 58  12 05 1B 1B 43 1A D9 38  >WPΉ–$ίX....C.Ω8
00000190 D2 91 6C 14 DA 7B AF 4A  82 1A 31 F6 B8 D6 58 A3  .‘l.Ϊ{–J,.1φΈΦX£
000001A0 AF 5C 63 24 23 19 69 96  61 E0 B5 55 C8 C4 B1 34  –\c$#.i–aϋμUΘΔ±4
000001B0 A8 DE 23 DF 1A A3 2C D7  51 2B 47 C2 CF 55 35 07  ¨ή#ί.£,XQ+GBOU5.
000001C0 91 7C D5 60 E3 96 84 C2  55 96 18 86 8A 7C AF 74  ‘|Υ`γ–„BU–.†.|–t
000001D0 6A 05 35 D6 91 F5 B0 30  C5 56 9A 67 88 6C 60 0B  j.5Φ‘υ°0EV.g.l`.
000001E0 78 35 B2 F6 9C CB 16 8C  AC 1A 56 45 91 D7 06 99  x5²φ.Λ..¬.VE‘X.™
000001F0 26 4C 0C 1D 47 A7 48 92  23 79 8A D9 B2 24 51 64  &L..G§H’#y.Ω²$Qd
00000200 6C 44 10 39 1A 49 E7 68  7B D4 5C B5 EF 91 48 92  lD.9.Iηh{T\μο‘H’
00000210 58 28 5A 1F 7B 96 6B AD  2D 6A C8 2A 5B 62 68 82  X(Z.{–k--jΘ*[bh,
```

Εικόνα 9. Δεκαεξαδικής μορφής δεδομένα που προέκυψαν από τον κωδικοποιητή εντροπίας για το σήμα ***eegchb070707mit***



Εικόνα 10. Μέσο ποσοστό συμπίεσης των eeg - ecg σημάτων



Εικόνα 11. Ποσοστά ρυθμού συμπίεσης σε MATLAB, C α', C β' ενδεικτικά για μερικά sampling rate του σήματος eegchb070707

Το μέσο ποσοστό συμπίεσης των σημάτων που επεξεργαστήκαμε φαίνονται στο παραπάνω διάγραμμα. Το μεγαλύτερο μέσο ποσοστό συμπίεσης επιτυγχάνεται στο πρώτο σήμα eegchb070707mit, 67,60%

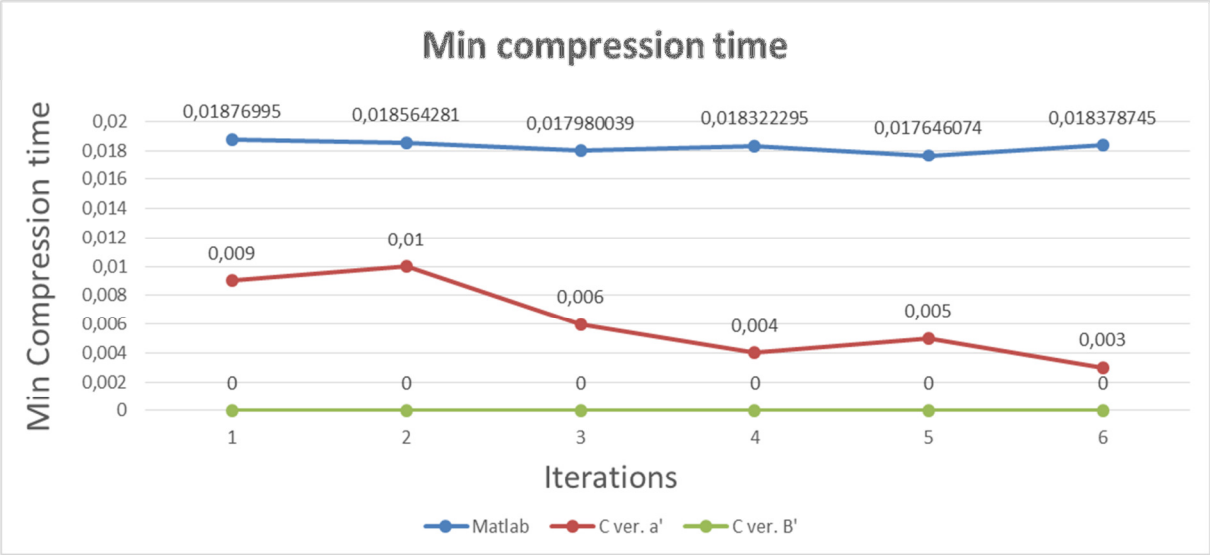
Ενώ τα κωδικοποιημένα στοιχεία είναι τα ίδια στον κώδικα που αναπτύχθηκε και στις δύο γλώσσες προγραμματισμού παρατηρούνται σε λίγα σημεία κατά ένα πολύ μικρο ποσοστό- γύρω στο 0,4- μικρότερα ποσοστά συμπίεσης στον κώδικα C εκδοχή β'. Καθώς τα δεδομένα που βγαίνουν από τον κωδικοποιητή εντροπίας είναι τα ίδια και σε MATLAB και σε C, αυτή η διακύμανση μπορεί μόνο να δικαιολογηθεί από τον τρόπο που υπολογίζεται το μέγεθος των δεδομένων προς συμπίεση. Το MATLAB και η C εκδοχή α' χρησιμοποιούν την εντολή ftell-fseek και dir()- file.bytes αντίστοιχα, ενώ στην εκδοχή β' υπολογίζονται «ωμά» με έναν μετρητή για τις γεμάτες θέσεις του πίνακα. Η εντολή ftell δεν είναι αξιόπιστη εντολή και οδηγεί ορισμένες φορές σε διαφορετικούς υπολογισμούς. [10]

Ο χρόνος συμπίεσης, όταν γίνεται σε τερματικά, έχει κόστος τόσο στο σκληρό δίσκο όσο και στην cache του επεξεργαστή. Καθώς ο κώδικας του αντικειμένου της εργασίας προορίζετε για ενσωματωμένα συστήματα, με περιορισμένη ποσότητα ειδικού σκοπού hardware, ο χρόνος κωδικοποίησης των δεδομένων είναι ακόμα πιο σημαντικός. (bus transfer, ram, flash μνήμη)

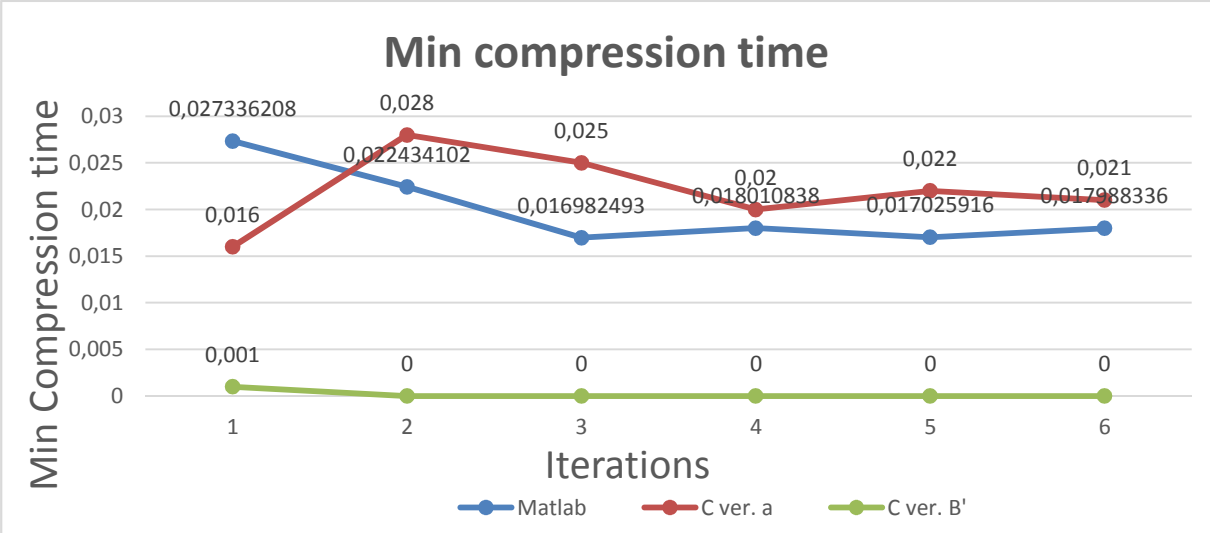
Για το λόγο αυτό, στη συνέχεια εξετάστηκε ο χρόνος που χρειάστηκε για την κωδικοποίηση των δεδομένων. Για να τρέξω τον κώδικα και να συγκεντρώσω χρονικές μετρήσεις ακολούθησα μια διαδικασία για να βεβαιωθώ ότι τα αποτελέσματα μου θα είναι όσο το δυνατόν λιγότερο επηρεασμένα από διεργασίες του υπολογιστή. Αρχικά έκανα defragment, απενεργοποίησα τις περισσότερες διεργασίες εκκίνησης και έκλεισα τον υπολογιστή. Με το άνοιγμα του υπολογιστή, περίμενα 3 λεπτά, άνοιξα το MATLAB ή το visual studio αντίστοιχα και περίμενα αλλά 2 λεπτά ώστε να τελειώσουν οι διεργασίες της εκκίνησης τους. Στη συνέχεια έτρεξα τον κώδικα αρκετές φορές συνεχόμενα. Πριν εκτελεστεί ο κώδικας, οι εντολές

print απενεργοποιήθηκαν, μπαίνοντας σε μορφή σχολίου, για να μην επιβαρύνουν το χρόνο εκτέλεσης.

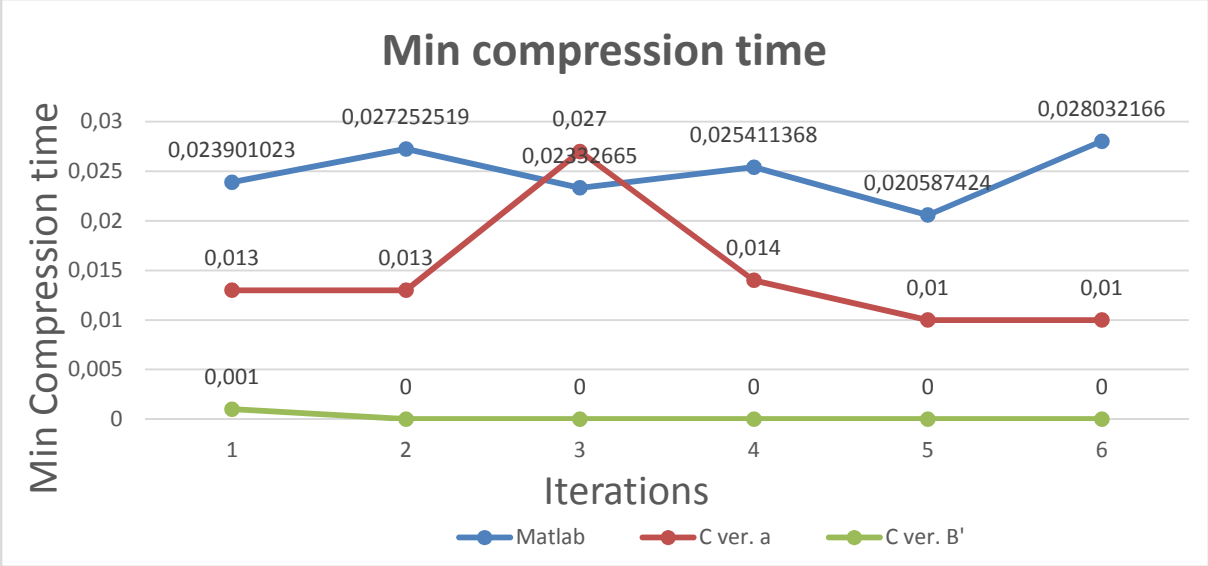
Ακολουθεί μια σύγκριση του μέγιστου, ελάχιστου και μέσου χρόνου κωδικοποίησης των δεδομένων, ενδεικτικά, για 6 iterations μεταξύ του κώδικα που αναπτύχθηκε στο MATLAB και στη C (έκδοση α' και β') και των 8 ιατρικών σημάτων που επεξεργαστήκαμε.



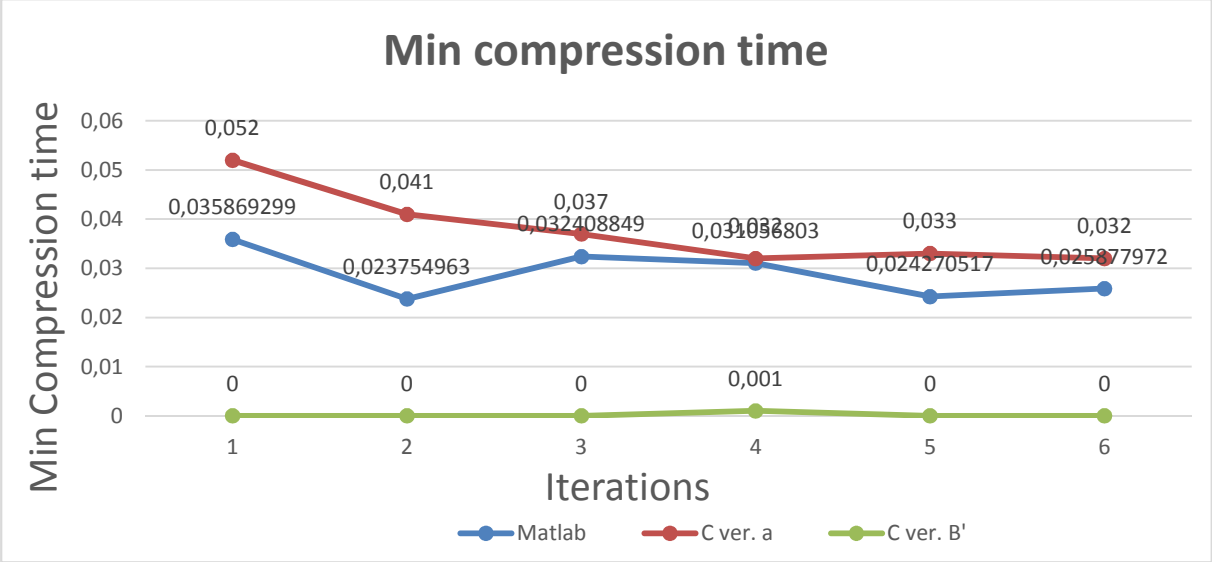
Εικόνα 12. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegchb070707mit***



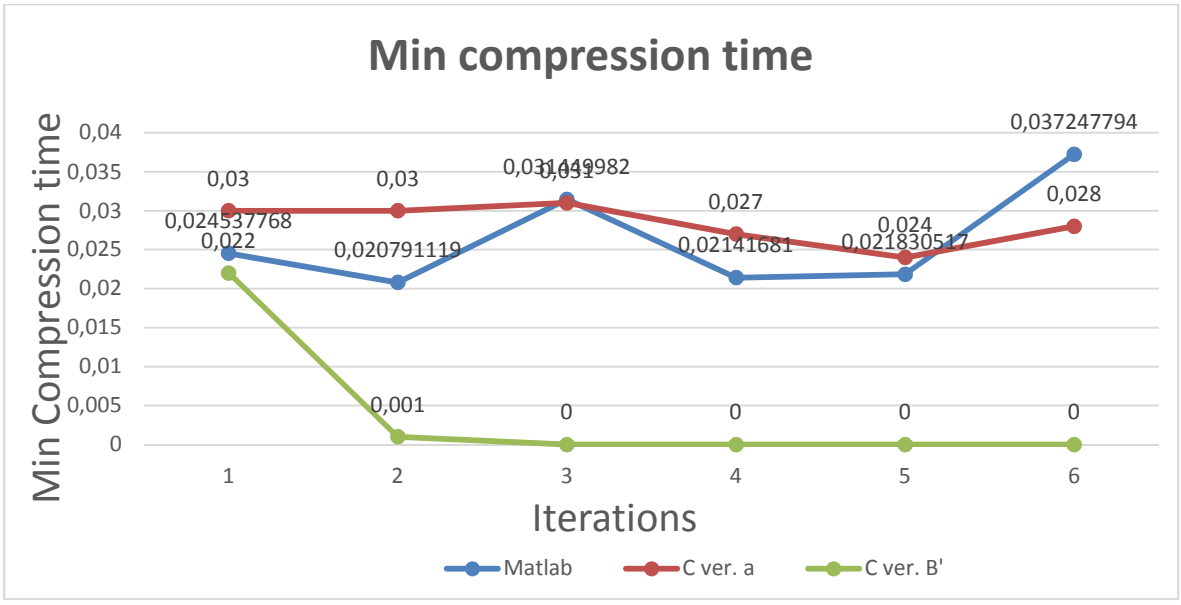
Εικόνα 13. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgA04apnea***



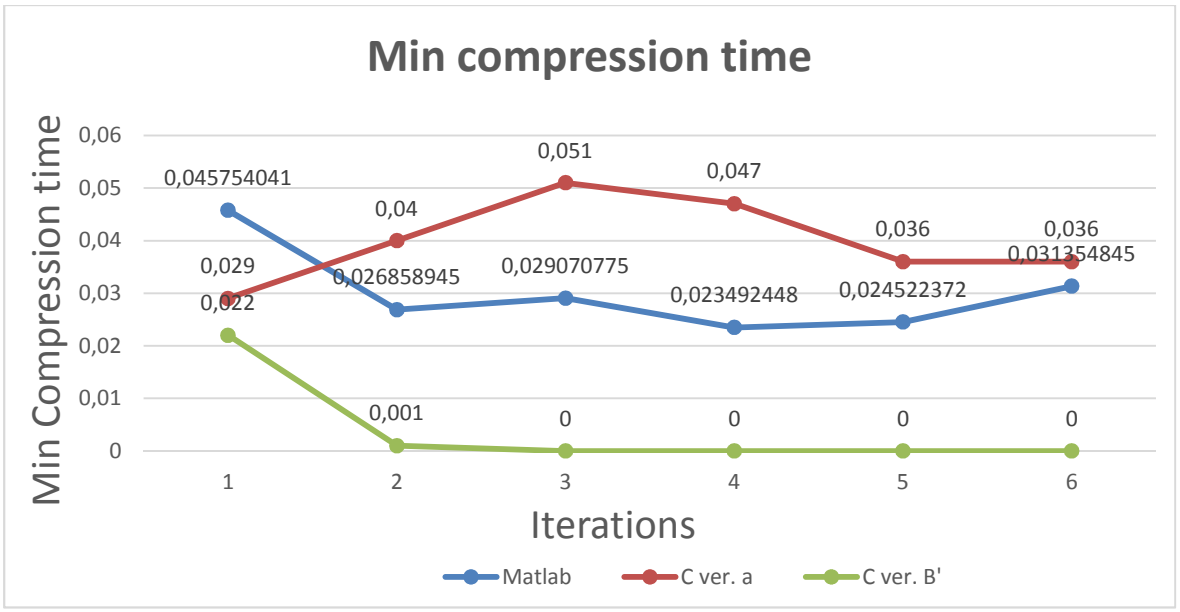
Εικόνα 14. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgB05apnea***



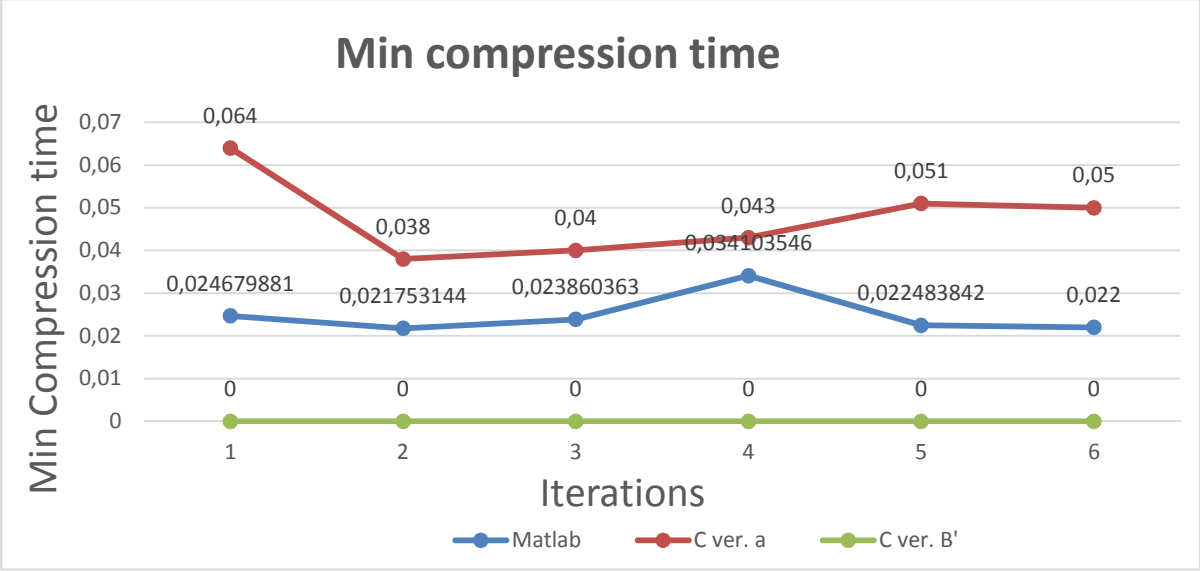
Εικόνα 15. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegchb141414mit***



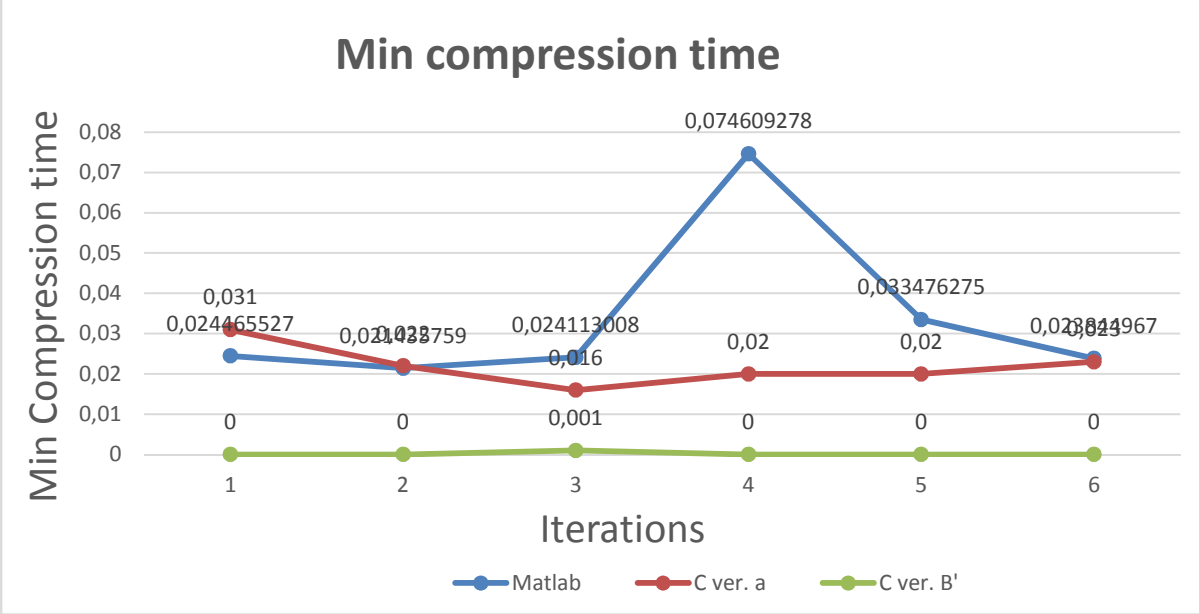
Εικόνα 16. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgUoP***



Εικόνα 17. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegF4UoP***



Εικόνα 18. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegOZUop***

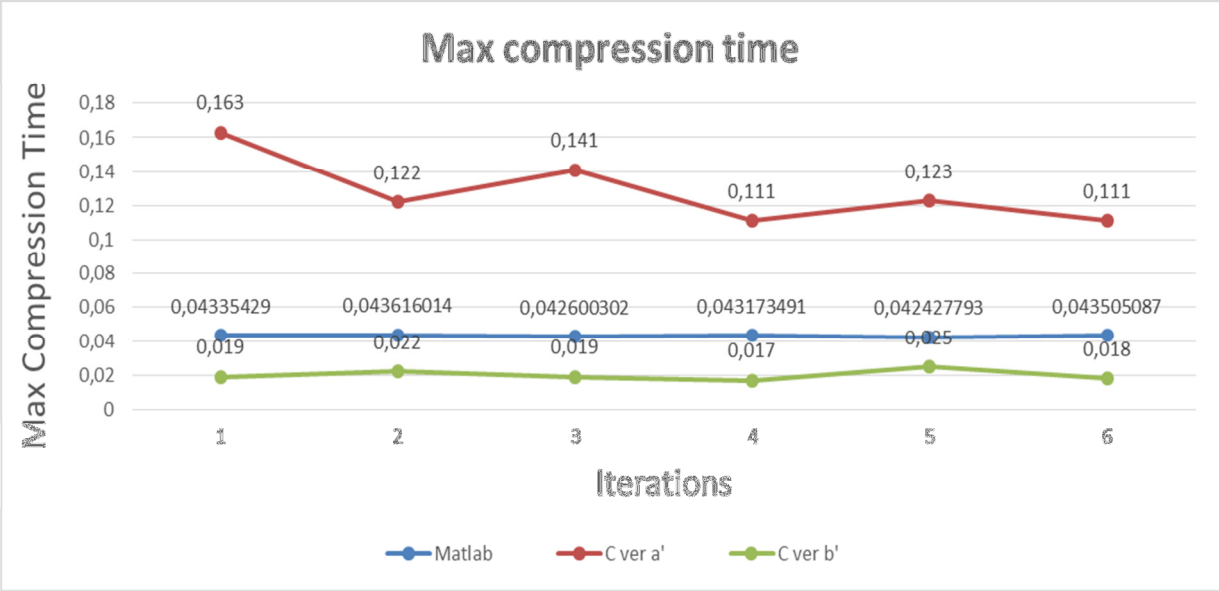


Εικόνα 19. Σύγκριση του μικρότερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgEkgMove***

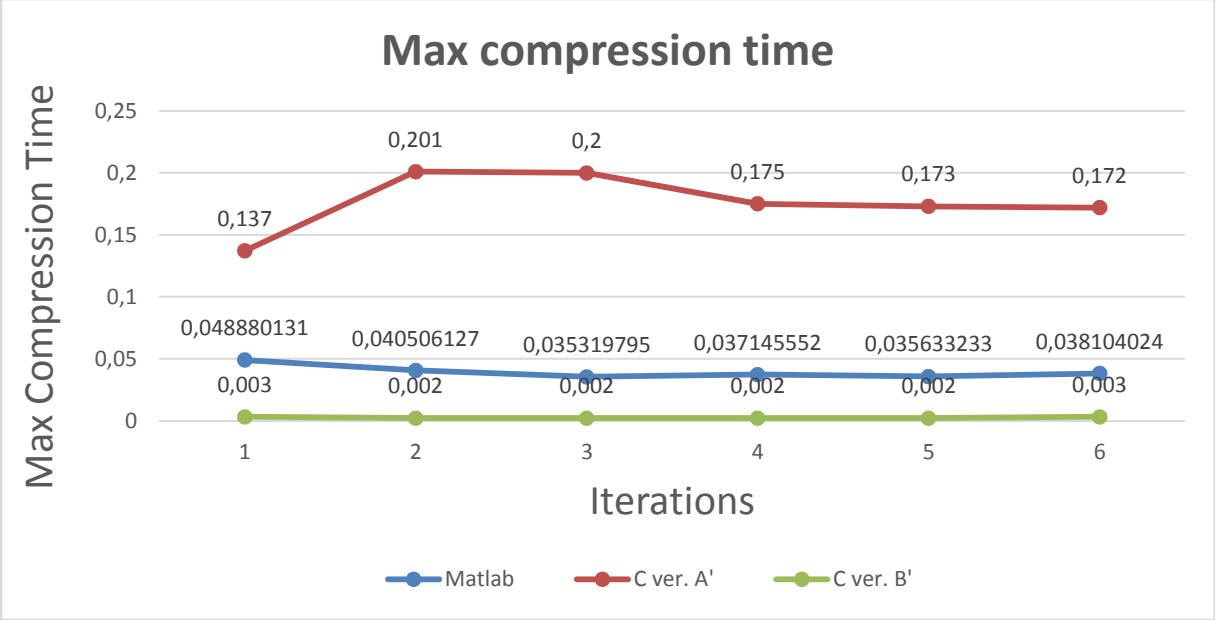
Τα παραπάνω διάγραμματα δείχνουν τον ελάχιστο χρόνο συμπίεσης των δεδομένων μεταξύ του κώδικα σε Matlab, C εκδοχή α' (έκδοση με pointers, αποθήκευση δεδομένων σε αρχείο, κτλ.) και C έκδοση β' (έκδοση με αποθήκευση δεδομένων σε πίνακα, χωρίς pointer κτλ.). Παρατηρούμε ότι ο ελάχιστος χρόνος συμπίεσης πραγματοποιείται στην κωδικοποίηση C εκδοχή β' με τιμές πολύ κοντά στο μηδέν. Ακουλουθεί η εκδοχή του MATLAB και τέλος η εκδοχή α' με pointers και αποθήκευση σε αρχείο με τον μεγαλύτερο min time που φθάνει μεχρι τα 0,07 Milliseconds. Από τα παραπάνω είναι φανερό ότι η εκδοχή a' C δεν έχει τα αποτελέσματα που περιμέναμε(C είναι γλώσσα compiled και όχι Interpreted όπως το MATLAB γι αυτό περιμέναμε τουλάχιστον να είναι γρηγορότερου του) και έρχεται τελευταία σε χρόνο έναντι των υπολοίπων. Αυτή η παρατήρηση ανάγεται σε αρκετά

αίτια όμως στο σημείο αυτό καταγράφουμε απλά την τάση αυτή και προχωράμε στα επόμενα διαγράμματα χρόνου, ωστέ να προχωρήσω στην ανάλυση επιλογικά.

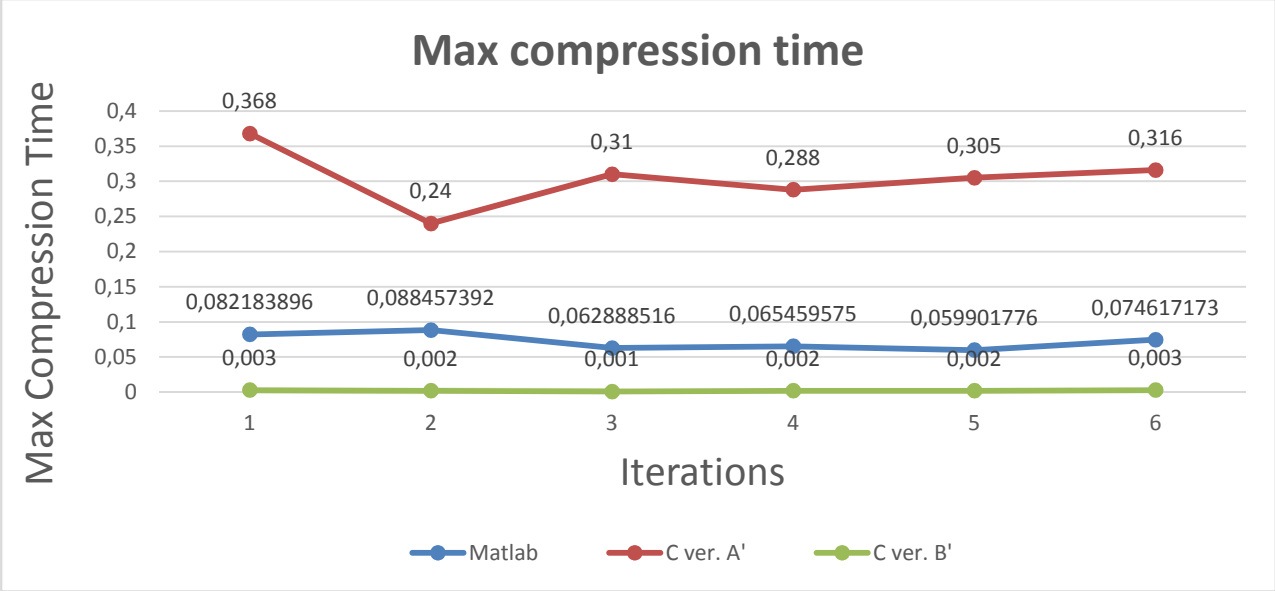
Επίσης χρειάζεται να σημειωθέι η τάση στη μορφή a' στο μέσο μικρότερο χρόνο συμπίεσης, με τα eeg σήματα να έχουν το μεγαλύτερο μέσο μικρότερο χρόνο συμπίεσης ένω στην εκδοχή b' τα ecg.



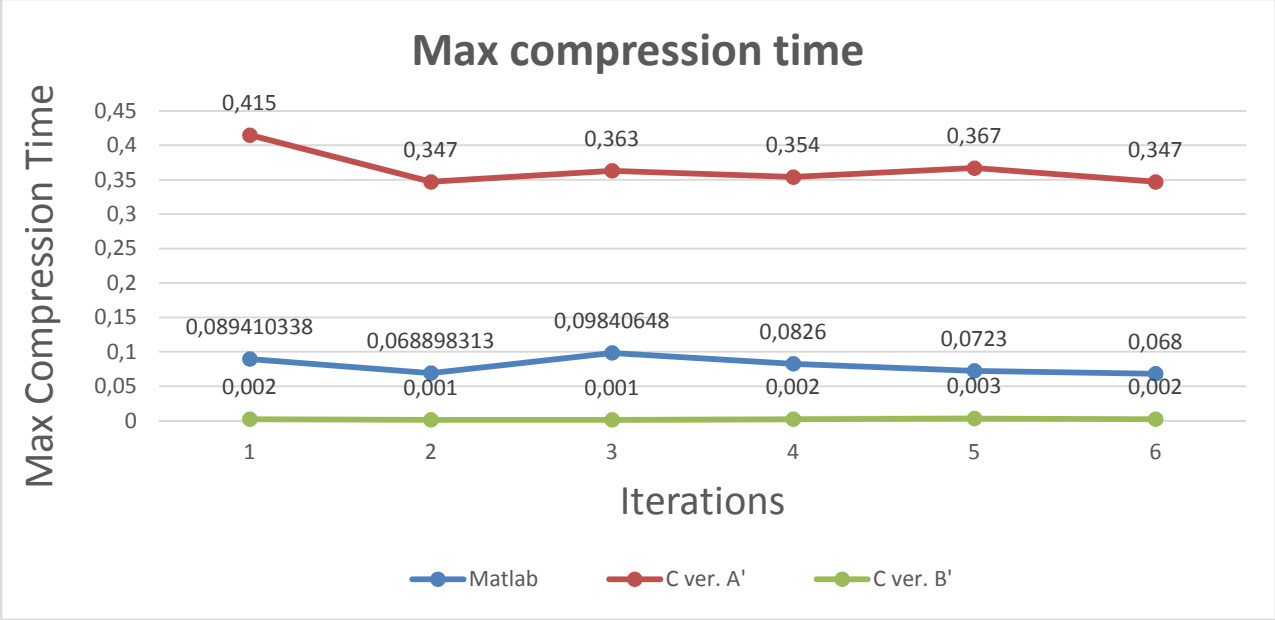
Εικόνα 20. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα ***eegchb070707mit***



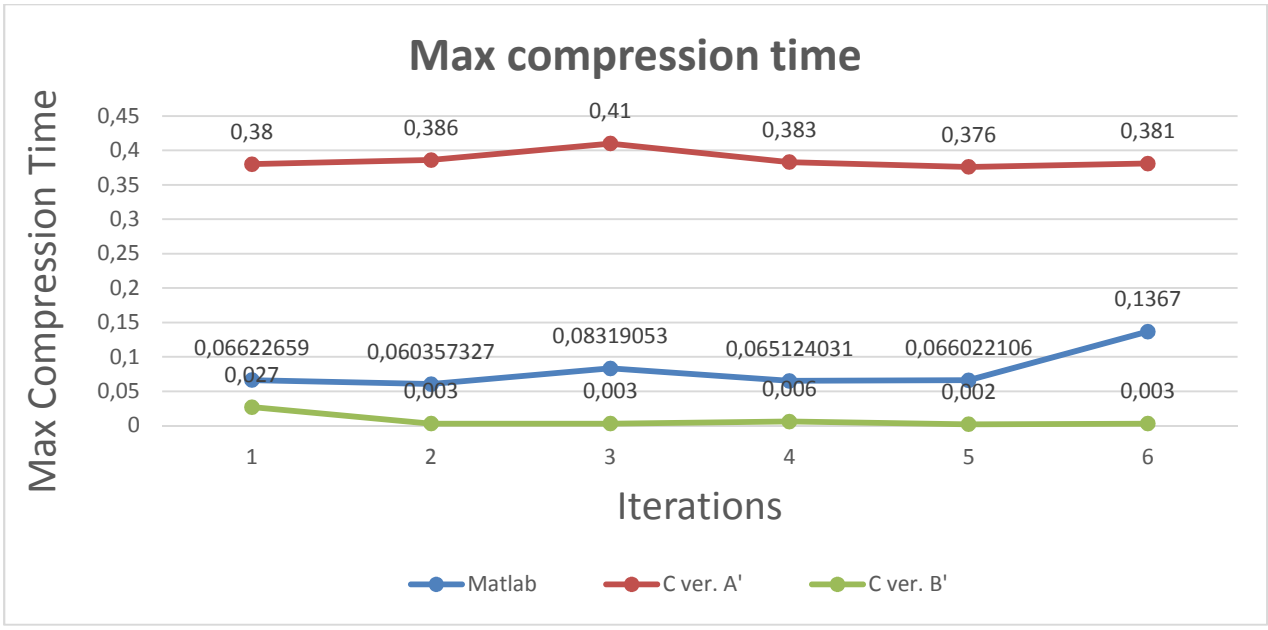
Εικόνα 21. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgA04apnea***



Εικόνα 22. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgB05apnea***

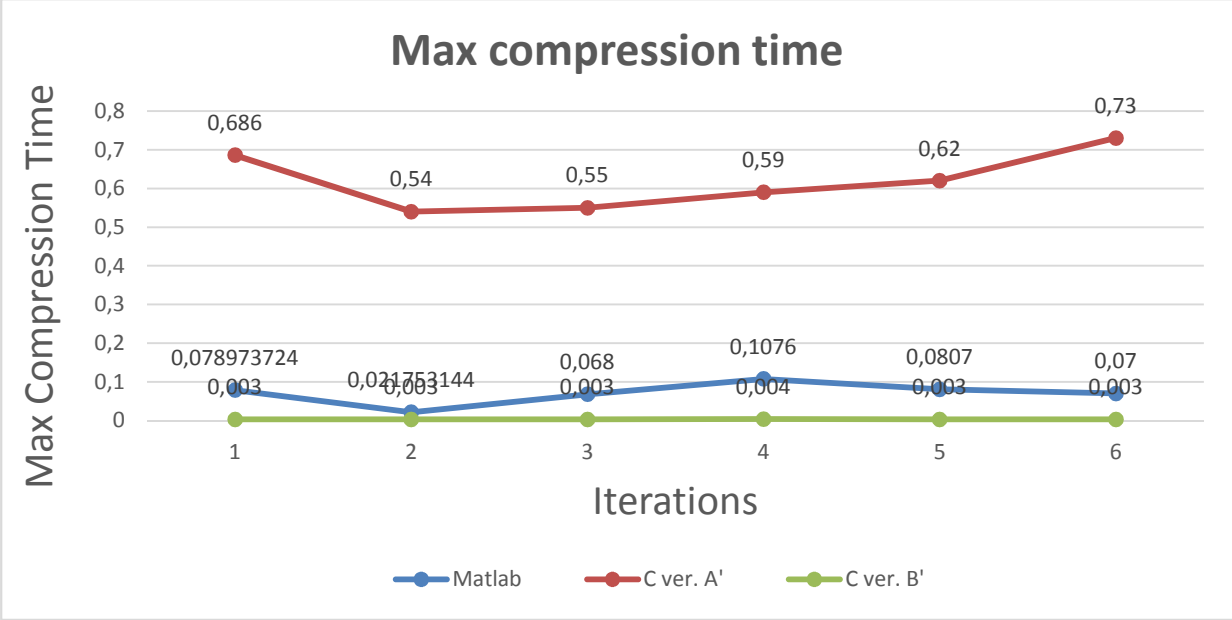


Εικόνα 23. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegchb141414mit***

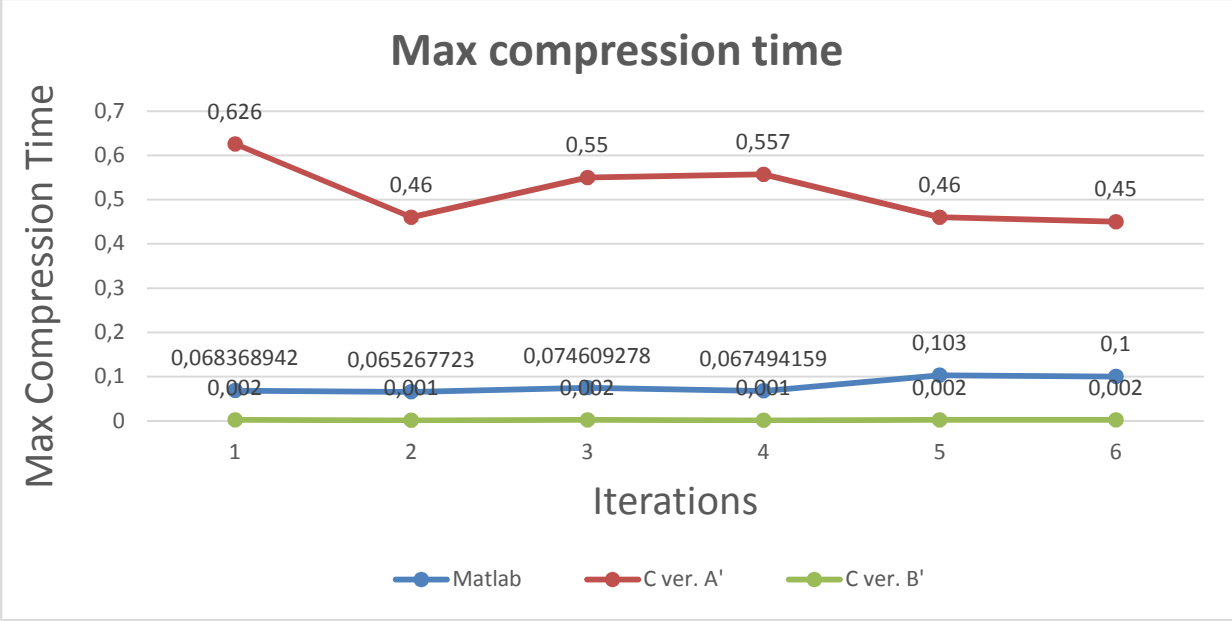


Εικόνα 24. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgUoP1***

Εικόνα 25. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegF4UoP1***



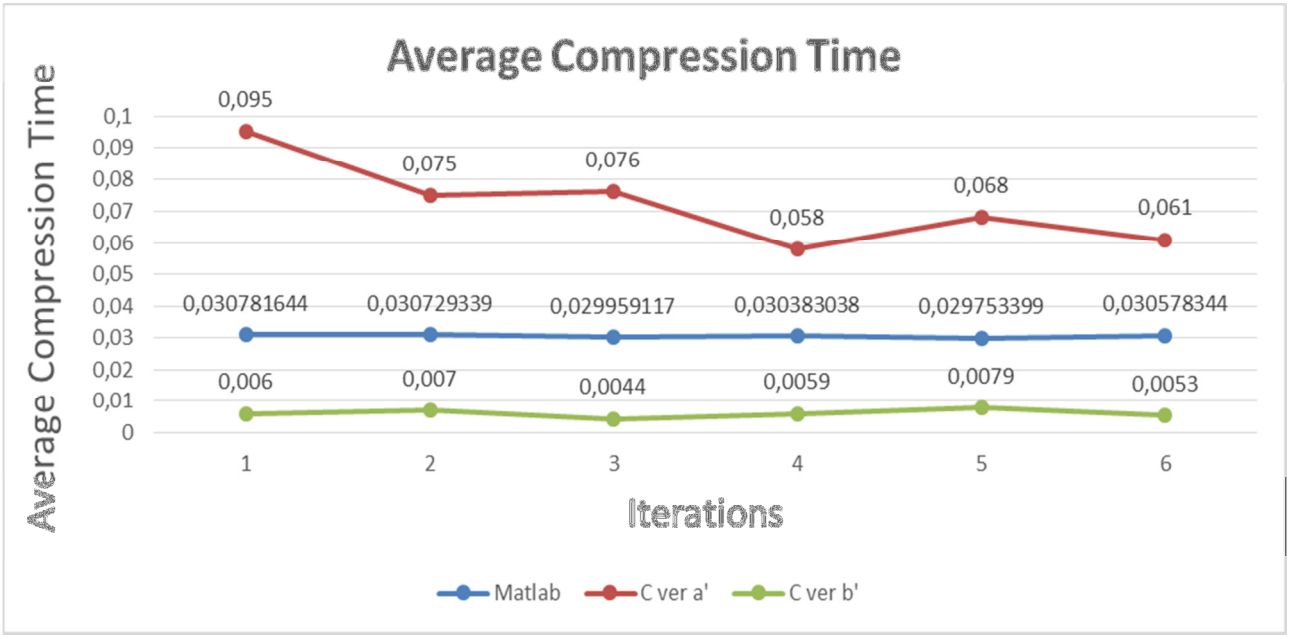
Εικόνα 26. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***eegOZUoP1***



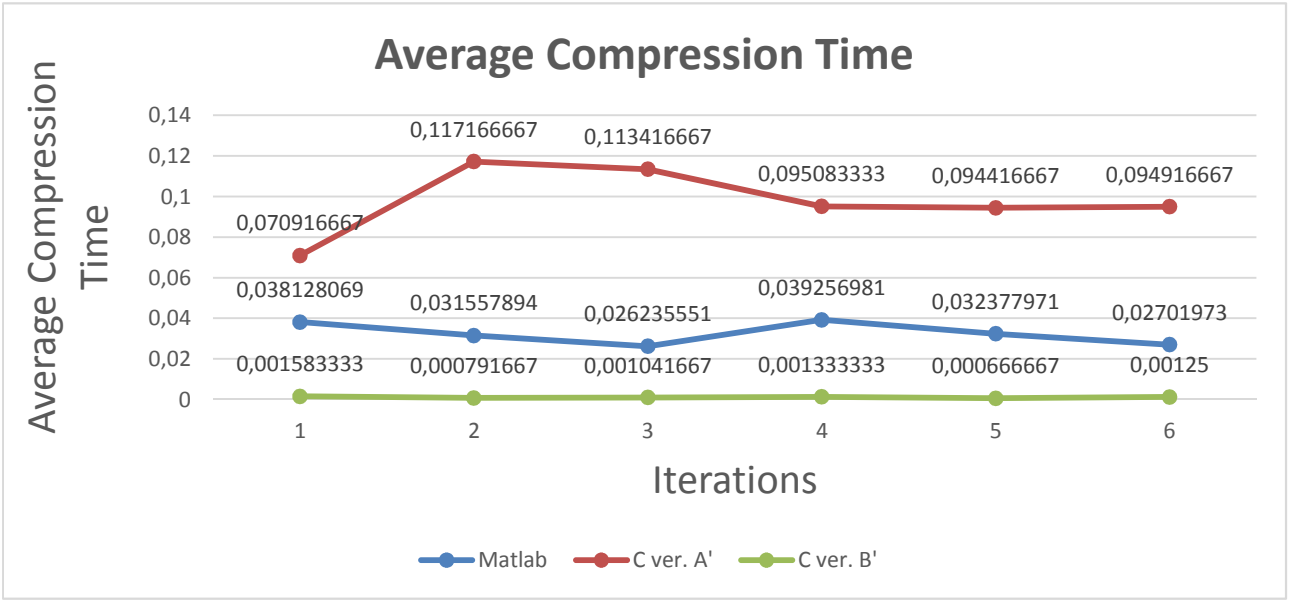
Εικόνα 27. Σύγκριση του μεγαλύτερου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα του σήματος ***ecgEkgMove***

Η μεγαλύτερη διάρκεια της συμπίεσης των δεδομένων έγινε από τον κώδικα σε γλώσσα C α' εκδοχή. Ακολουθεί το MATLAB και η C β' εκδοχή αρκετά χαμηλότερα 0,2 έως 0,5 milliseconds. Ο κώδικας C β' εκδοχή έχει τη μικρότερη μέγιστη διάρκεια συμπίεσης συμπίεσης γύρω στα 0,02 milliseconds μέσο όρο. Όπως προκύπτει από τα παραπάνω, η τάση να έρχεται σε χρονική διάρκεια τελευταίος (χαμηλότερο χρόνο αρα καλύτερο χρόνο) ο κώδικα b' C εκδοχή, ακολούθως MATLAB και τέλος C εκδοχή a', επιβεβαιώνεται και εδώ.

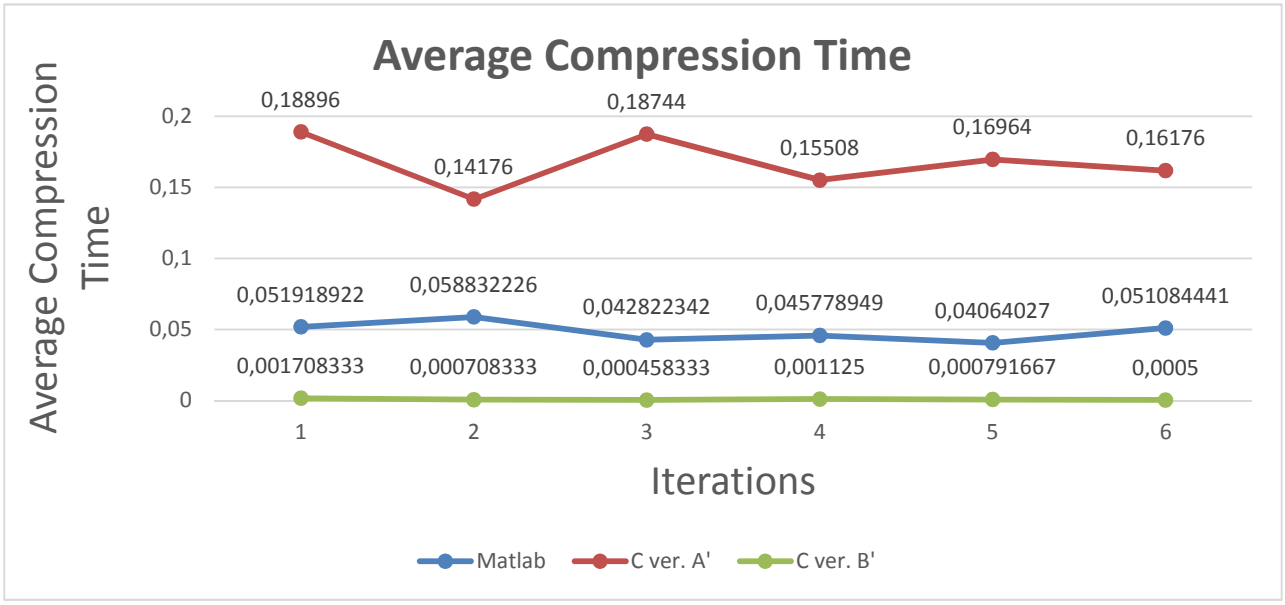
Ο μέσος μέγιστος χρόνος συμπίεσης των σημάτων στο MATLAB τοποθετείται γύρω από τις ίδιες τιμές περίπου, με τα σήματα με κατάληξη apnea να σημειώνουν το χαμηλότερο μέσο μικρότερου χρόνου συμπίεσης και τα ecgUoP1 και eegOZUoP1 το μεγαλύτερο. Στον κώδικα C a' εκδοχή προηγούνται ξεκάθαρα τα eegOZUoP1 eegF4UoP1 ενώ στη b' εκδοχή το eegchb070707mit και το ecgUoP1. Γίνεται λοιπόν εύκολα αντιληπτό ότι τα σήματα με κατάληξη UoP1 παρουσιάζουν μεγαλύτερο μέσο μέγιστο χρόνο συμπίεσης.



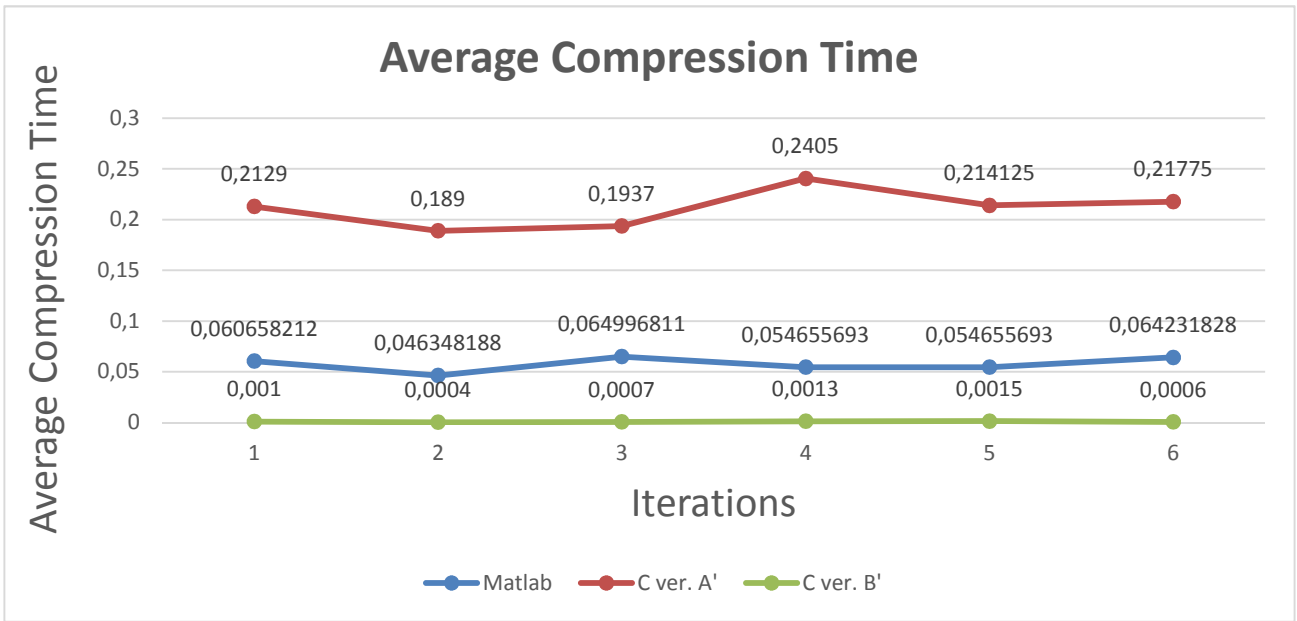
*Εικόνα 28. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **eegchb070707mit***



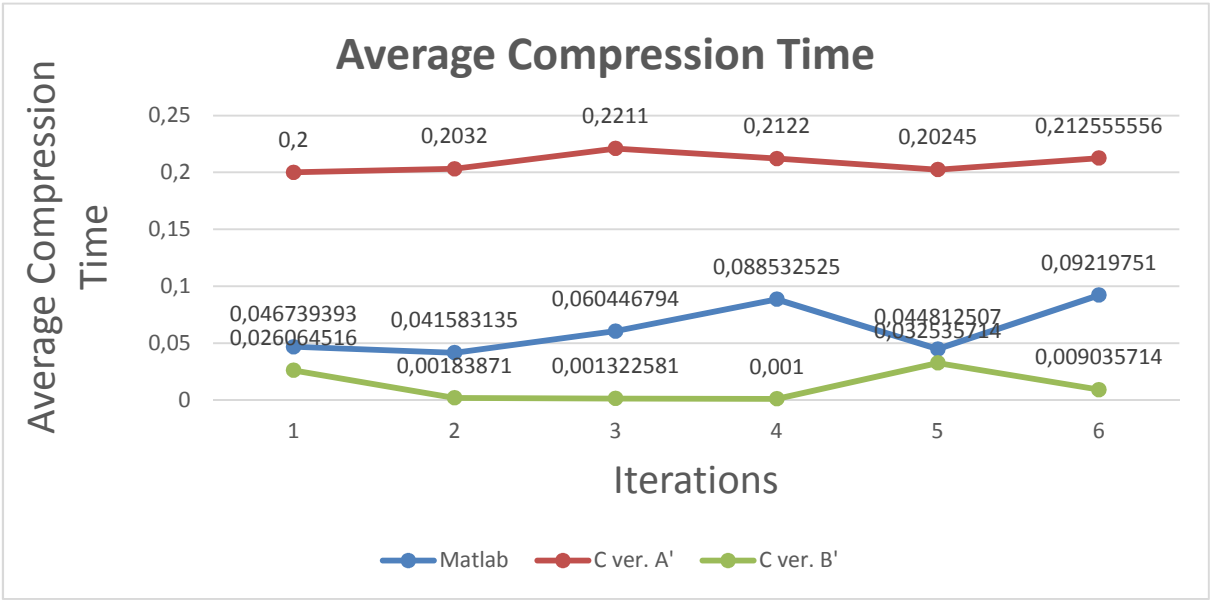
*Εικόνα 29. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **ecgA04apnea***



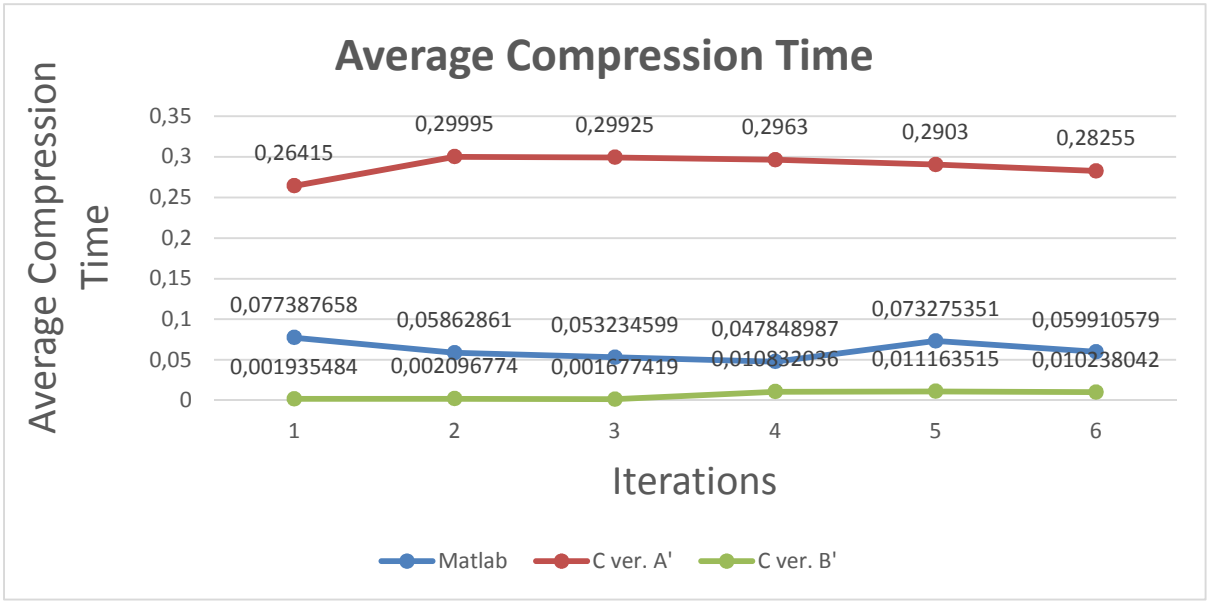
Εικόνα 30. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα ***ecgB05apnea***



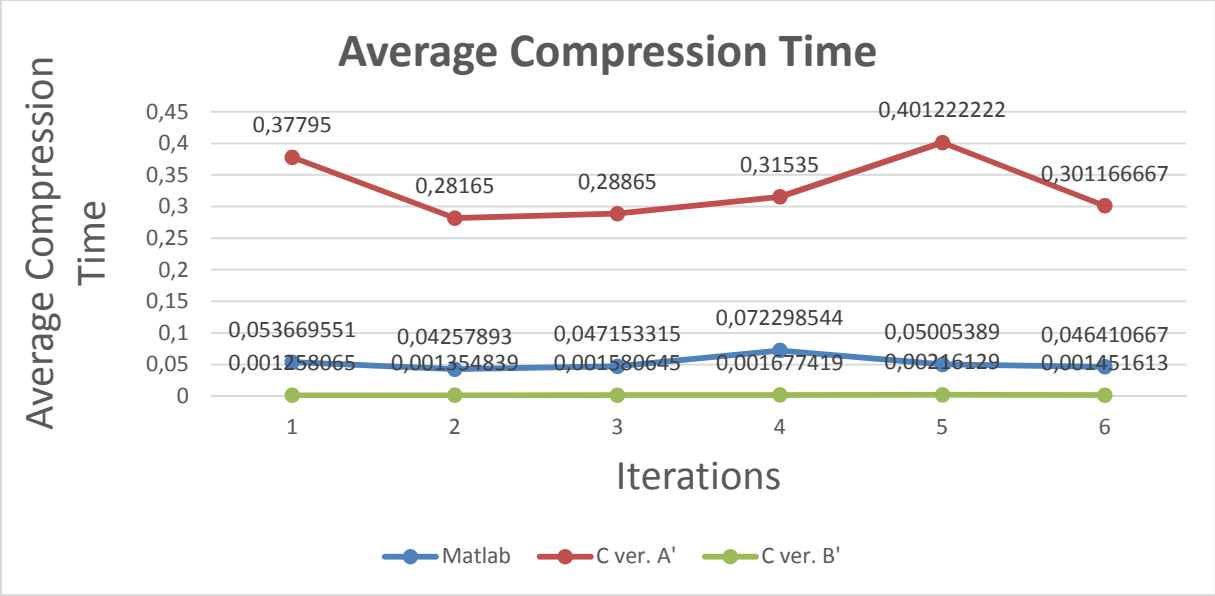
Εικόνα 31. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα ***eegchb141414mit***



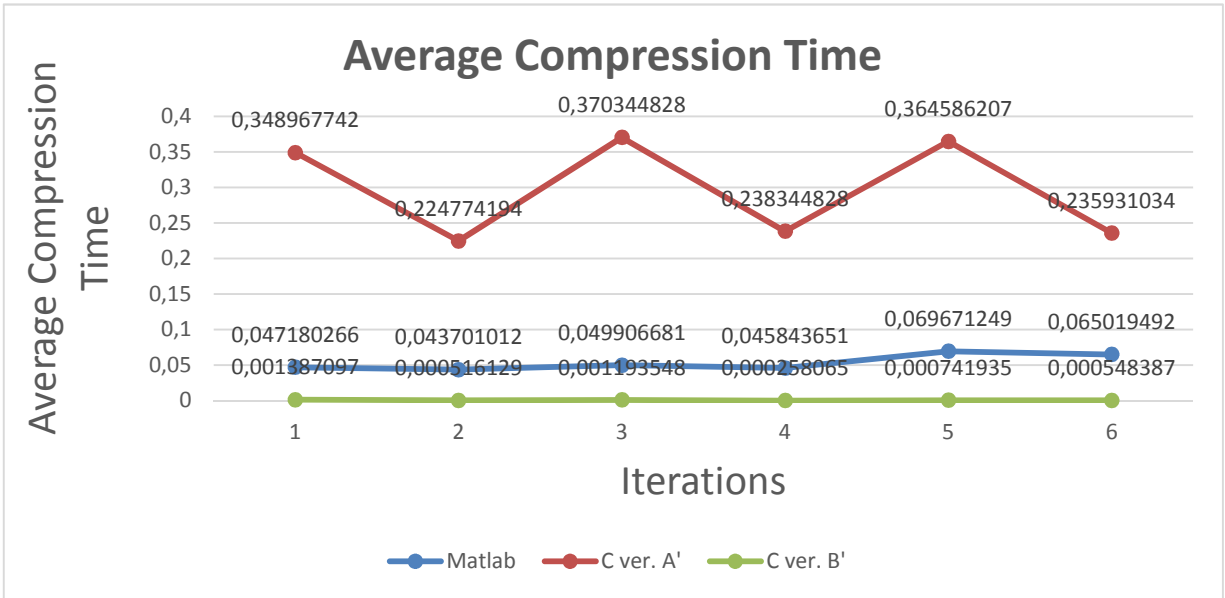
*Εικόνα 32. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **ecgUoP***



*Εικόνα 33. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **eegF4UoP***



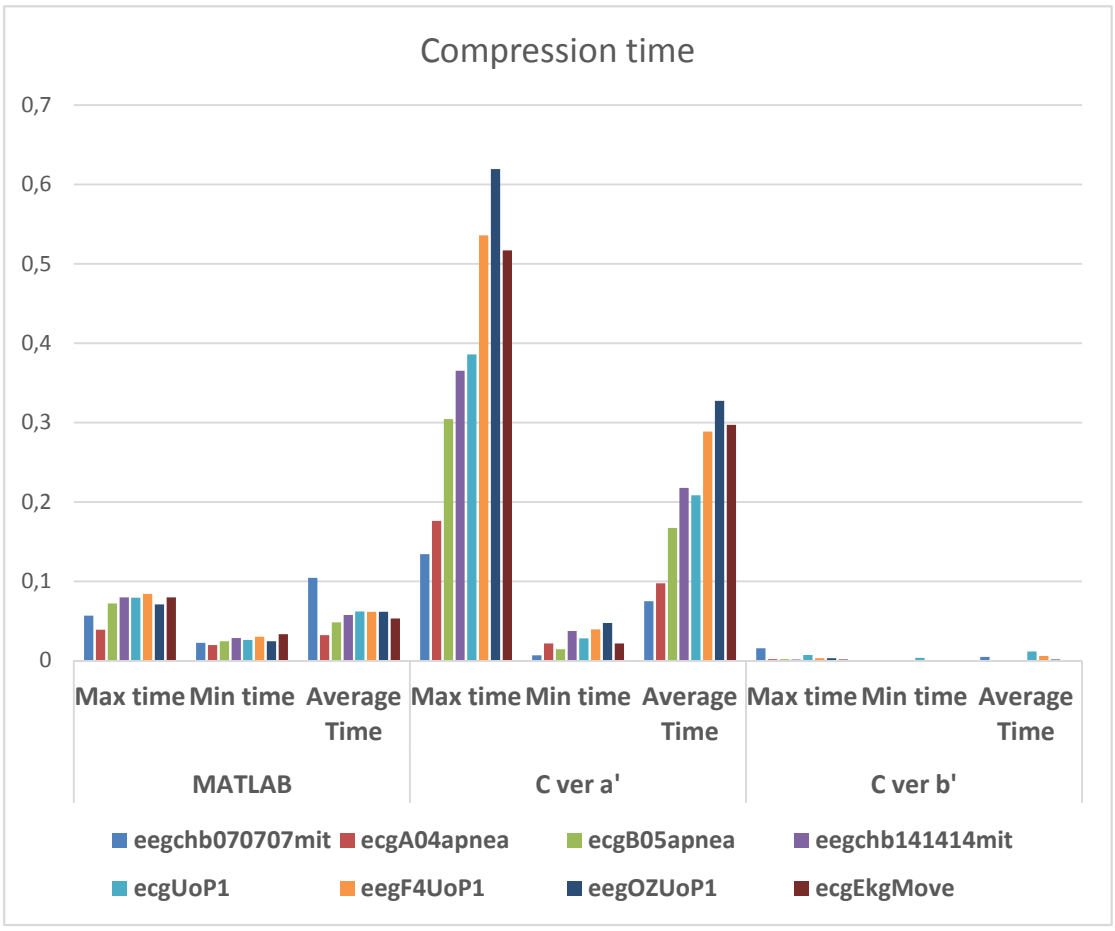
*Εικόνα 34. Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **eegOZUop***



*Εικόνα 35. . Σύγκριση του μέσου χρόνου συμπίεσης για όλες τις εκδοχές του κώδικα **eegOZUop***

Τον καλύτερο μέσο χρόνο συμπίεσης επιτυγχάνει η εκδοχή β' C γύρω από το 0,001, έπεται το MATLAB με 0,0004-0,01 και τέλος ακολουθεί η εκδοχή α' με το μέσο χρόνο συμπίεσης 0.0728-0,35. Για μια ακόμη φορά το μοτίβο επαληθεύεται. Δηλαδή ο μέσος χρόνος συμπίεσης είναι περισσότερος στην εκδοχή C a', ακολουθώντας

MATLAB και τέλος η b' εκδοχή της C. Ο μέσος χρόνος συμπίεσης σε κώδικα MATLAB είναι παρόμοιος για όλα τα σήματα εκτός του σήμα eegchb070707mit που προηγείται κατά 0,04 milliseconds. Στην εκδοχή a' του κώδικα C τα σήματα eegF4UoP1, eegOZUoP1 και ecgEkgMove είχαν πάλι την υψηλότερη καταγραφή χρόνου- εδώ στο μέσο χρόνο συμπίεσης. Στην εκδοχή b' του κώδικα C μεγαλύτερο μέσο χρόνο συμπίεσης έχει το ecgUoP1, eegF4UoP1 και το eegchb141414mit. Συμπερασματικά και σε αυτές τις μετρήσεις τα σήματα με UoP1 καθυστερούν περισσότερο στη συμπίεση.



Ο χρόνος συμπίεσης στο MATLAB (μέσος, μικρότερος, μεγαλύτερος) των σημάτων κυμαίνεται στα ίδια milliseconds. Ξεχωρίζει, στο διάγραμμα μέσου χρόνου συμπίεσης, το σήμα eegch070707 με περίπου 0,04 milliseconds μεγαλύτερο μέσο χρόνο συμπίεσης από τα υπόλοιπα σήματα. Ωστόσο σε αντίθεση με το διάγραμμα του MATLAB, στο διάγραμμα των μετρήσεων του χρόνου συμπίεσης για τον κώδικα c a' εκδοχή, παρουσιάζεται μια τάση των σημάτων συγκεκριμένα των eegOZUoP1, eegF4UoP1 στα οποία παρατηρείται μεγαλύτερος χρόνος συμπίεσης σε όλες τις μετρήσεις. Το σήμα ecgEkgMove έρχεται εξίσου ψιλά μαζί με τα προαναφερθέντα σήματα σε όλες τις μετρήσεις εκτός από το μέσο μικρότερο χρόνο συμπίεσης. Εκεί παίρνει τη θέση του το eegchb141414mit. Ακολούθως στις μετρήσεις που λήφθηκαν για τον κώδικα c b' εκδοχή, τα σήματα ecgUoP1, eegF4UoP1 και eegchb070707mit έχουν το μεγαλύτερο χρόνο συμπίεσης. Εύκολα λοιπόν από όλα τα παραπάνω συνάγουμε το συμπέρασμα ότι ο χρόνος συμπίεσης των σημάτων UoP1 είναι μεγαλύτερος. Τα σήματα αυτά ήταν τα πιο απαιτητικά της συλλογής μας με συχνότητα δειγματοληψίας 2500 Hz και μέγεθος δείγματος 16 bit.

Ο κώδικα MATLAB σημειώνει όπως παρατηρούμε αργή κωδικοποίηση. Το MATLAB γενικά έχει μεγαλύτερο χρόνο εκτέλεσης από τη C καθώς εκτός από την εκτέλεση του πραγματικού κώδικα, το περιβάλλον πρέπει επίσης να μεταφράσει το πρόγραμμα σε κώδικα μηχανής . Αυτό συμβαίνει γιατί είναι ερμηνευμένη (interpreted) γλώσσα, που σημαίνει ότι μεταγλωττίζεται (compile) γραμμή προς γραμμή, με κάθε γραμμή πρώτα να μεταγλωττίζεται και στη συνέχεια να τρέχει και να προχωράει στην επόμενη. Η C από την άλλη είναι γλώσσα μεταγλώττισης (compiled), δηλαδή πριν τρέξει ο κώδικας ο μεταγλωττιστής μεταφράζει όλο το πρόγραμμα σε γλώσσα μηχανής. Μόλις γίνει αυτό, το πρόγραμμα μπορεί να τρέξει.

Εικόνα 36. Συγκεντρωτικό συγκριτικό διάγραμμα μεταξύ του μέσου μικρότερου, μέσου μεγαλύτερου και μέσου χρόνο συμπίεσης του κάθε σήματος σε όλες τι μορφές που αναπτύχθηκε ο κώδικας (MATLAB, C ver a', C ver b')

Το overhead του λογισμικού με την ερμηνεία των οδηγιών του MATLAB είναι ο κύριος λόγος που περιμένουμε το MATLAB να μην κάνει τόσο καλούς χρόνους.

Λόγω όλων αυτών, είναι αναμενόμενο η γλώσσα C να σημειώνει τους καλύτερους χρόνους (εκδοχή β').

Παρόλα τα πλεονεκτήματα που αναλύθηκαν παραπάνω της C έναντι του MATLAB, η έκδοση α' C κάνει το χειρότερο χρόνο.

Οι επιδόσεις της C εκδοχή α' αιτιολογούνται, πρώτον, από τους πολλούς pointers. Στην περίπτωση της εκδοχής β' συγκεκριμένα, ο μεταγλωττιστής ξέρει τη διεύθυνση του πίνακα (που είναι και η διεύθυνση του πρώτου στοιχείου) και προσπελάζει. Στην περίπτωση της εκδοχής α', ξέρει τη διεύθυνση του pointer και διαβάζει την τιμή του, η οποία δείχνει στη θέση της μνήμης. Και έτσι γίνεται μια επιπλέον ανακατεύθυνση, με αποτέλεσμα ο κώδικας να γίνεται πιο αργός.

Δεύτερον, στην εκδοχή α' το άνοιγμα/ κλείσιμο αρχείων που γίνεται αρκετές φορές για να διαβάζουμε στοιχεία και γράφουμε τα κωδικοποιημένα αποτελέσματα. Η προσπέλαση αρχείων σε σχέση με την προσπέλαση στοιχείων σε πίνακα επιβαρύνει το χρόνο εκτέλεσης, κάνοντας τον κώδικα C εκδοχή β' πιο αποδοτικό.

6.2 Αξιολόγηση της υλοποίησης του αλγορίθμου στο περιβάλλον Keil uVision

Μέχρι τώρα μελετήθηκε η εφαρμογή του αλγορίθμου σε γλώσσα C (τυπική και απλουστευμένη εκδοχή) και αξιολογήθηκε η απόδοση του σε compression ratio και compression time (min –max- average) σε διάφορα σήματα eeg και ecg. Για να

επιτελέσουμε το σκοπό της διπλωματικής, ο οποίος ήταν η ανάπτυξη του αλγόριθμου LEC σε γλώσσα κατάλληλη για εφαρμογή σε ενσωματωμένα συστήματα, δοκιμάσαμε να τρέξουμε τον αλγόριθμο σε περιβάλλον ανάπτυξης για ενσωματωμένα συστήματα. Το περιβάλλον που επιλέχθηκε ήταν το Keil μVision V5.16a. Η προσομοίωση έγινε σε ARM® Cortex®-M0+ επεξεργαστή. Για να καταφέρουμε να εκτελέσουμε τον κώδικα σε αυτό το περιβάλλον κάναμε τις παρακάτω αλλαγές:

- Καθώς επιλέξαμε να κάνουμε προσομοίωση η μέτρηση του χρόνου βάση πραγματικού ρολογιού επεξεργαστή ήταν αδύνατη, γι' αυτό για αρχή περιοριστήκαμε στην καταγραφή της μέτρησης compression ratio.
- Κάναμε τις αρχικοποιήσεις πινάκων εκτός της συνάρτησης Main

Στην αποσφαλμάτωση του κώδικα παρατηρήθηκε πολύ μικρότερο compression ratio από αυτό που σημειώθηκε στις προηγούμενες εκτελέσεις του κώδικα. Το πρόβλημα εντοπίστηκε στους πίνακες outocode οι οποίοι δεν «γέμιζαν» με κωδικοποιημένα δεδομένα. Η συνάρτηση strcpy είναι η συνάρτηση που έστελνε τα κωδικοποιημένα δεδομένα στο outcode και πρόκειται για μια μη ασφαλή συνάρτηση που δεν βάζει null terminator στο string και αντικαθιστά string σε συγκεκριμένου μεγέθους πίνακα. Τα αποτελέσματα του compression ratio προέκυψαν εσφαλμένα από αυτή τη συνάρτηση γι' αυτό και αντικαταστάθηκε με την strncat() βάζοντας και null terminator. Βρέθηκε, μετά από τις κατάλληλες αλλαγές, το compression ratio κατά μέσο όρο στο 61.81% και το μέγιστο έως και 69.2% ενδεικτικά για το σήμα eegchb070707mit. Στις προηγούμενες εκτελέσεις του κώδικα το σήμα είχε μέσο compression ration 67%.

Κεφάλαιο 7

6.3 Συμπεράσματα

Η παρούσα εργασία παρουσιάζει την αξία της συμπίεσης των ιατρικών δεδομένων(εγκεφαλογραφήματά και καρδιογραφήματά) και εξετάζει την απόδοση του αλγορίθμου LEC (Lossless Entropy Compression). Μετά από μελέτη αρκετών αλγορίθμων αποφασίσαμε να ασχοληθούμε με αυτόν λόγω της μικρής υπολογιστικής πολυπλοκότητας του και του καλού ρυθμού συμπίεσης χωρίς απώλειες(καλύτερο ρυθμό συμπίεσης χωρίς απώλειες εως σήμερα). Είναι απλός, διατηρεί μικρό λεξικό, απαιτεί λίγη μνήμη για την εκτέλεση του και το μόνο μειονέκτημα του είναι η αδυναμία προσαρμογής στην αλλαγή συσχετισμού των δεδομένων από τον αισθητήρα.

Η υλοποίηση του αλγορίθμου έγινε με σκοπό την εφαρμογή σε ενσωματωμένα συστήματα. Για τη μελέτη του αλγόριθμου σχεδιάστηκε και αναπτύχθηκε κώδικας σε C, γλώσσα κατάλληλη για εφαρμογή σε ενσωματωμένα συστήματα, ο οποίος δεχόταν τα ιατρικά σήματα τα κωδικοποιούσε και έπαιρνε μετρήσεις χρόνου και ρυθμού συμπίεσης. Για να καλυφθούν οι ανάγκες αρκετών μικροελεγκτών, αναπτύχθηκε ο αλγόριθμος σε δύο εκδοχές γλώσσας στη C. Μια εκδοχή με κώδικα στην τυπική μορφή της C(με malloc, pointers, ορισμούς αρχείων) και μια εκδοχή απλούστερη(με στατικούς πίνακες, χωρίς pointers και λιγότερες malloc). Για τον έλεγχο και την επαλήθευση των αποτελεσμάτων είχαμε ως αναφορά την υλοποίηση του κώδικα σε γλώσσα MATLAB. Το περιβάλλον ανάπτυξης ήταν το Microsoft Visual Studio Community.

Η μετατροπή του κώδικα από τη γλώσσα MATLAB σε C είχε αρκετές απαιτήσεις ειδικά σε binary επίπεδο. Το MATLAB διαθέτει πλούσιες βιβλιοθήκες, αυτόματη κατανομή μνήμης, υποστηρίζει πολλούς τύπους δεδομένων και είναι γλώσσα βασισμένη σε πίνακες. Στη C δεν έχουμε αυτά τα πλεονεκτήματα και γενικά αποτελεί

χαμηλότερου επιπέδου γλώσσα. Για να γίνει ακριβής μετατροπή μεταξύ των γλωσσών μελετήθηκε ο κώδικας σε bit level (μετατροπές little endian –big endian, bitwise operations)

Η αξιολόγηση του αλγόριθμου έγινε σε 8 σήματα, εγκεφαλογραφήματα και καρδιογραφήματα, συγκρίνοντας τα δεδομένα εξόδου από τον κωδικοποιητή εντροπίας, το μέσο ρυθμό και χρόνο συμπίεσης που επιτυγχάνονται μεταξύ του κώδικα MATLAB- C a' εκδοχή- C b' εκδοχή. Η μετατροπή ήταν επιτυχής καθώς τα δεκαεξαδικά δεδομένα εξόδου του κωδικοποιητή εντροπίας ήταν όμοια σε όλες τις εκδοχές του κώδικα. Τα αποτελέσματα του μέσου ρυθμού συμπίεσης των σημάτων ήταν για μέγιστο μέσο ρυθμό συμπίεσης 67% σε σήμα eeg και ελάχιστο μέσο ρυθμό συμπίεσης 44,8% σε σήμα πάλι eeg.

Ο χρόνος συμπίεσης αποτελεί πολύ σημαντικό κριτήριο για την αποδοτικότητα του αλγορίθμου καθώς ο κώδικας προορίζεται για ενσωματωμένα συστήματα με περιορισμένη ποσότητα ειδικού σκοπού hardware (bus transfer, ram, flash μνήμη). Για τη μέτρηση του χρόνου συμπίεσης ακολουθήθηκε μια διαδικασία περιορισμών των διεργασιών του υπολογιστή και ανασυγκρότησης του δίσκου για να έχουμε όσο το δυνατόν πιο καθαρές μετρήσεις χρόνου. Μετρήθηκε ο μέγιστος, μικρότερος και ο μέσος χρόνος συμπίεσης σε όλες τις εκδοχές του κώδικα σε όλα τα ιατρικά σήματα. Ο ελάχιστος χρόνος συμπίεσης πραγματοποιείται στην κωδικοποίηση C εκδοχή β' με τιμές πολύ κοντά στο μηδέν. Ακολουθεί η εκδοχή του MATLAB και τέλος η εκδοχή α' με pointers και αποθήκευση σε αρχείο με τον μεγαλύτερο min time που φθάνει μέχρι τα 0,07 milliseconds. Η μεγαλύτερη διάρκεια της συμπίεσης των δεδομένων έγινε από τον κώδικα σε γλώσσα C α' εκδοχή. Ακολουθεί το MATLAB και η C β' εκδοχή αρκετά χαμηλότερα 0,2 έως 0,5 milliseconds. Ο κώδικας C β' εκδοχή έχει τη μικρότερη μέγιστη διάρκεια συμπίεση γύρω στα 0,02 milliseconds κατά μέσο όρο. Τον καλύτερο μέσο χρόνο συμπίεσης επιτυγχάνει η εκδοχή β' C γύρω από το 0,001, έπεται το MATLAB με 0,0004-0,01 και τέλος ακολουθεί η εκδοχή α' με το μέσο χρόνο συμπίεσης 0.0728-0,35. Γενικά παρατηρείται ότι τον καλύτερο χρόνο συμπίεσης έχει η C b' εκδοχή. Ακολουθεί το MATLAB και τελευταίος έρχεται ο κώδικας C b' εκδοχή. Είναι αναμενόμενο το MATLAB να μην έχει καλές επιδόσεις στο χρονική διάρκεια της συμπίεσης διότι ως interpreted γλώσσα έχει μεγαλύτερο χρόνο εκτέλεσης από τη C. Οι επιδόσεις της C a' εκδοχή είναι εντυπωσιακές καθώς περιμέναμε να έχουν προβάδισμα έναντι του MATLAB. Ο χαμηλός χρόνος συμπίεσης της αιτιολογείται από τους πολλούς pointers, και το άνοιγμα/ κλείσιμο αρχείων που γίνονται αρκετές φορές για να διαβαστούν και να γραφτούν τα δεδομένα. Στη C b' εκδοχή γίνεται προσπέλαση στοιχείων σε πίνακα(ο μεταγλωττιστής ξέρει τη διεύθυνση του πίνακα) έτσι δεν έχουμε ανακατευθύνσεις και η διαδικασία της εκτέλεσης γίνεται πιο ελαφριά.

Βιβλιογραφία

[1] Introduction to Data Compression, Second Edition (The Morgan Kaufmann Series in Multimedia Information and Systems)
Author: Khalid sayood

[2] http://users.iit.demokritos.gr/~ntsap/courses/bes04/lectures/mm03.pdf
[3]http://www.medialab.ntua.gr/education/MultimediaTechnology/MultimediaTechnologyNotes/chap2a_3.htm
[4]http://digitalschool.minedu.gov.gr/modules/ebook/show.php/DSB103/173/1206,4408/
[5] Compression of medical sensor data- Al Wagener
[6] Data Compression The Complete Reference- David Salomon
[7] An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks – Francesco Marcelloni, Massimo Vecchio
[8] The Essentials of Computer Organization and Architecture- Linda Null, Julia Lobur
[9] www.mathworks.com
[10] Ftell: https://www.securecoding.cert.org/confluence/display/c/FIO19-C.+Do+not+use+fseek()+and+ftell()+to+compute+the+size+of+a+regular+file
[11] https://www.visualstudio.com/vs/community/
[12] http://www.differencebetween.net/science/difference-between-ecg-and-eeg/#ixzz4Q4rQoE7e
[13] An Adaptive Lossless Data Compression Scheme for Wireless Sensor Networks- Jonathan Gana Kolo,1 S. Anandan Shanmugam,1 DavidWee Gin Lim,1
Li-Minn Ang,2 and Kah Phooi Seng3
[14] An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks - Francesco Marcelloni1,∗ and Massimo Vecchio2
[15] Lossless Compression of Time-Series Data Based on Increasing Average of Neighboring Signals – Tetsuya Takezawa, Koichi Asakura, and Toyohide Watanabe
[16] Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks - Christopher M. Sadler and Margaret Martonosi