|  |
| --- |
|  |



**ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ**

**Τμήμα Μηχανικών Πληροφορικής**

Τομέας Υπολογιστικών Τεχνικών & Συστημάτων

**Υλοποίηση αλγορίθμων Ψηφιακής Επεξεργασίας Εικόνας με δυνατότητα χρήσης τους από το διαδίκτυο**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**Σιδηρόπουλος Ανδρέας-Ανέστης**

**ΑΕΜ 3325**

**Επιβλέπων**

**Στρουθόπουλος Χαράλαμπος**

**ΣΕΡΡΕΣ**

**ΜΑΡΤΙΟΣ 2019**

**ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ**

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Πρόλογος

Σκοπός της παρούσας πτυχιακής εργασία είναι: α) Η ανάπτυξη αλγορίθμων Ψηφιακής Επεξεργασίας Εικόνας για εκπαιδευτικούς και πρακτικούς λόγους ώστε να χρησιμοποιούνται διαδικτυακά σε σχήμα client-server από φοιτητές και άλλους ενδιαφερόμενους. Τούτο είναι εξαιρετικά χρήσιμο για εκπαιδευτική χρήση αλλά κι εφαρμογή. β) Η χρήση και εμβάθυνση στον διαδικτυακό προγραμματισμό για εφαρμογές αυτού του τύπου.

Υλοποιήθηκαν διαδικασίες κατωφλίωσης ψηφιακών εικόνων κλίμακας του γκρι με βάση την εντροπία, την διασπορά και με την χρήση νευρωνικού δικτύου, με παράλληλη παρουσίαση του θεωρητικού μέρους και δυνατότητα άσκησης του χρήστη. Η εργασία υλοποιήθηκε εξολοκλήρου στην γλώσσα σεναριακού προγραμματισμού JavaScript.

Οφείλω να ευχαριστήσω τον καθηγητή του τμήματος Μηχανικών Πληροφορικής, του ΤΕΙ Κεντρικής Μακεδονίας, Χαράλαμπο Στρουθόπουλο για την εμπιστοσύνη που μου έδειξε, αναθέτοντας μου το θέμα της πτυχιακής εργασίας. Επιπροσθέτως θα ήθελα να ευχαριστήσω τον συμφοιτητή Ντρούφλια Αντρέα για την πολύτιμη βοήθεια του στην ανάπτυξη ιστοσελίδων.

**ΛέξειςΚλειδιά:**

JavaScript, Κατωφλίωση, Εντροπία, Διασπορά, Νευρωνικό δίκτύο, ΨΕΕ, Otsu, Kapur

Πίνακας περιεχομένων

Κεφάλαιο 1. 3

1.1 Βασικές έννοιες και ορισμοί ΨΕΕ 3

1.1.1 Ψηφιακή Επεξεργασία Εικόνας 3

1.1.2 Ψηφιακή εικόνα 3

1.1.3 Είδη ψηφιακών εικόνων 3

1.1.4 Κατωφλίωση 3

1.1.5 HTML 3

1.1.6 CSS 3

1.1.7 JavaScript 3

1.1.8 Bootstrap 3

1.1.9 jQuery 3

1.1.10 JSON 3

1.1.11 Plotly 3

1.1.12 MVC Αρχιτεκτονική 3

1.1.13 PNG 3

1.1.14 LocalStorage 3

Κεφάλαιο 2. 3

2.1 Ιστόγραμμα 3

2.2 Εξισορροπημένο ιστόγραμμα 3

2.3 Κατωφλίωση με βάση την διασπορά 3

2.4 Κατωφλίωση με βάση την εντροπία 3

2.5 Πολυκατωφλίωση με χρήση νευρωνικού δικτύου 3

Κεφάλαιο 3. 3

3.1 Τεχνικά θέματα υλοποίησης 3

3.2 Βασική δομή της εφαρμογής 3

3.3 Εισαγωγή αλγορίθμου ΨΕΕ στην εφαρμογή 3

3.4 Μελλοντικές επεκτάσεις 3

Κεφάλαιο 4. 3

4.1 Εγχειρίδιο χρήσης 3

4.1.1 Αρχική σελίδα 3

4.1.2 Σελίδα πρακτικής εξάσκησης 3

Κεφάλαιο 5. 3

# 

**Εισαγωγή**

Στην παρούσα πτυχιακή εργασία μελετήθηκαν οι ακόλουθοι τέσσερις αλγόριθμοι επεξεργασίας ψηφιακών εικόνων αποχρώσεων του γκρι:

* Εύρεσης και εξισορρόπησης του ιστογράμματος.
* Δυαδικοποίησης με τη μέθοδο της διασποράς.
* Δυαδικοποίησης με τη μέθοδο της εντροπίας.
* Μείωσης των χρωμάτων.

Το περιβάλλον που αναπτύχθηκε ενσωματώνει τους παραπάνω αλγορίθμους παρουσιάζοντας την θεωρητική επεξήγηση τους, παρέχοντας τους τη δυνατότητα για πρακτική εξάσκηση και την δυναμική εκτέλεση αυτών. Επιπλέον οι κατασκευαστικές του προδιαγραφές καθιστούν ικανή οποιαδήποτε μελλοντική επέκταση της εφαρμογής, ενσωματώνοντας νέους αλγορίθμους ή επεκτείνοντας τους ήδη υπάρχοντες. Στα κεφάλαια που θα ακολουθήσουν θα παρουσιάσουμε τις απαραίτητες έννοιες, σχετικές με το αντικείμενο της εργασίας και θα αναφερθούμε στο τρόπο υλοποίησης της. Συγκεκριμένα στο πρώτο κεφάλαιο αναφέρεται εισαγωγικά τι είναι ψηφιακή εικόνα, ψηφιακή επεξεργασία εικόνας, ποιοι είναι οι τύποι και τα χαρακτηριστικά των ψηφιακών εικόνων, αλλά και βασικές έννοιες που θα μας απασχολήσουν παρακάτω. Στο δεύτερο κεφάλαιο παρουσιάζονται οι αλγόριθμοι ΨΕΕ που πραγματεύεται η εργασία. Στο τρίτο κεφάλαιο παρουσιάζονται τεχνικά θέματα που αφορούν την υλοποίηση της εργασίας, σημεία του κώδικα που χρήζουν επεξήγησης και η δυνατότητα και ο τρόπος εμπλουτισμού με επιπλέον αλγορίθμους. Το τέταρτο κεφάλαιο αποτελεί εγχειρίδιο χρήσης.

## Βασικές έννοιες και ορισμοί ΨΕΕ

### Ψηφιακή Επεξεργασία Εικόνας

Επεξεργασία εικόνας ονομάζεται κάθε μορφή αλγοριθμικής επεξεργασίας, ανάλυσης και χειρισμού ψηφιακών δεδομένων εικόνας ή βίντεο. Στην επεξεργασία εικόνας, τόσο η είσοδος όσο και η έξοδος των υπολογισμών είναι δεδομένα εικόνας/βίντεο (έγχρωμα, ασπρόμαυρα ή σε αποχρώσεις του γκρίζου). Από την επεξεργασία εικόνας εκπορεύονται επίσης και αλγόριθμοι ανάλυσης/κατανόησης εικόνας. Μεγάλο μέρος του επιστημονικού υποβάθρου της επεξεργασίας εικόνας παρέχεται από την επεξεργασία σήματος, καθώς η ψηφιακή εικόνα μπορεί να θεωρηθεί δισδιάστατο χωρικό σήμα και το βίντεο τρισδιάστατο χωροχρονικό σήμα.

### Ψηφιακή εικόνα

Η ψηφιακή εικόνα είναι ένα πεπερασμένο σύνολο περιοχών όπου κάθε περιοχή είναι χρωματισμένη με χρώμα που προέρχεται από ένα πεπερασμένο σύνολο χρωμάτων. Στις περισσότερες των περιπτώσεων, μια ψηφιακή εικόνα είναι ένα ορθογώνιο, διαιρεμένο με γραμμές και στήλες σε ορθογώνιες περιοχές που κάθε μία έχει συγκεκριμένο χρώμα. Μια τέτοια περιοχή ονομάζεται στοιχείο της εικόνας ή εικονοστοιχείο. Στην αγγλική λέγεται *pixel* ή *pel*, όρος ο οποίος προέρχεται από τη σύντμηση των λέξεων picture element. Από τα παραπάνω καταλαβαίνουμε πως μια ψηφιακή εικόνα είναι ένα δισδιάστατο ψηφιακό σήμα.

Αν κάθε χρώμα κωδικοποιηθεί με έναν αριθμό τότε η ορθογώνια ψηφιακή εικόνα περιγράφεται από έναν πίνακα αριθμών *J×K*, όπου *J* το πλήθος των γραμμών και *K* το πλήθος των στηλών της ψηφιακής εικόνας. Η τιμή *I(j,k)* με *k=0,1,2….K-1* και *j=0,1,2….J-1* είναι ο κωδικός του χρώματος της ψηφιακής εικόνας.

Η μετατροπή μιας εικόνας σε ψηφιακή μορφή ουσιαστικά είναι η μετατροπή ενός δισδιάστατου αναλογικού σήματος σε ψηφιακό και απαιτεί τις διαδικασίες της δειγματοληψίας και του κβαντισμού.

### Είδη ψηφιακών εικόνων

Υπάρχουν τρία είδη ψηφιακών εικόνων που χαρακτηρίζονται από το πλήθος των χρωμάτων που περιέχουν:

1. *Δυαδικές εικόνες* (*binary images*): Κάθε εικονοστοιχείο των εικόνων μπορεί να χρωματιστεί με ένα από δύο χρώματα.(συνήθως άσπρο ή μαύρο). Για κάθε εικονοστοιχείο απαιτείται ένα bit πληροφορίας, π.χ. με τιμή μηδέν (0) για το μαύρο και ένα (1) για λευκό. Οι εικόνες των εγγράφων που αποτελούνται μόνο από το χρώμα του χαρτιού και της μελάνης αναπαρίστανται σε δυαδική ψηφιακή μορφή.
2. *Εικόνες αποχρώσεων του γκρι* (*gray level images*):Κάθε εικονοστοιχείο των εικόνων μπορεί να χρωματιστεί με μία από τις αποχρώσεις του γκρι οι οποίες ξεκινούν από το μαύρο και καταλήγουν στο λευκό. Από αυτές τις αποχρώσεις συνήθως λαμβάνονται 256 αντιπροσωπευτικές που κωδικοποιούνται με τιμές 0,1,2….255. Η απόχρωση κάθε εικονοστοιχείου προφανώς απαιτεί πληροφορία ενός byte.
3. *Έγχρωμες εικόνες* (*color images*) στις οποίες κάθε εικονοστοιχείο χρωματίζεται με χρώματα που προέρχονται από την ανάμειξη των αποχρώσεων του κόκκινου, πράσινου και μπλε (*RGB*). Για κάθε ένα από τα τρία αυτά χρώματα λαμβάνονται 256 αποχρώσεις δηλαδή πληροφορία του ενός byte. Συνεπώς κάθε εικονοστοιχείο της έγχρωμης εικόνας, απαιτεί 3 bytes.

### Κατωφλίωση

Συχνά τα εικονοστοιχεία ενός αντικειμένου μιας εικόνας παίρνουν τιμές σε ένα μικρό διάστημα αποχρώσεων. Αυτό οδηγεί συνήθως στη δημιουργία ενός τοπικού μέγιστου στην περιοχή του ιστογράμματος της εικόνας. Η εύρεση τέτοιων τοπικών μεγίστων διευκολύνει τον εντοπισμό των αντικειμένων της εικόνας και την απόδοσή της με λιγότερες κύριες αποχρώσεις. Παρακάτω θα περιγράψουμε διάφορες τεχνικές για τον καθορισμό τιμών του πεδίου των αποχρώσεων μεταξύ των οποίων εμφανίζονται τοπικά μέγιστα του ιστογράμματος. Οι τιμές αυτές λέγονται κατώφλια.

Για την μετατροπή μιας εικόνας αποχρώσεων του γκρι σε δυαδική είναι επιθυμητός ο προσδιορισμός των 2 κύριων αποχρώσεών της. Με τον τρόπο αυτό οι αποχρώσεις της εικόνας χωρίζονται σε 2 κλάσεις C1, C2 που κάθε μια αποτελείται από τις αποχρώσεις που βρίσκονται πιο κοντά σε μια από τις 2 κύριες αποχρώσεις. Ο διαχωρισμός αυτός ισοδυναμεί με την εύρεση μιας τιμής κατωφλιού Τ για την οποία θα ισχύει



Ακολούθως από την αρχική εικόνα *I* προκύπτει η δυαδική εικόνα *I΄* κάθε εικονοστοιχείο της οποίας δίνεται από τη σχέση.



### HTML

Το 1990 ο Sir Timothy Berners-Lee δημιούργησε μια προδιαγραφή της HTML και κατασκεύασε τον πρώτο φυλλομετρητή ιστού και το λογισμικό εξυπηρετητή. Σκοπός της δημιουργίας της ήταν να δοθεί η δυνατότητα στους ερευνητές, από όλο τον κόσμο, να οργανώσουν και να συγκεντρώσουν τις πληροφορίες τους. Η HTML (Hyper Text Markup Language) ορίζεται ως γλώσσα σήμανσης υπερκειμένου. Ο όρος υπερκείμενο (Hyper Text) αποτελεί μια ειδική περίπτωση βάσης δεδομένων. Η HTML είναι η κύρια γλώσσα σήμανσης για την δημιουργία ιστοσελίδων αλλά και διαδικτυακών εφαρμογών. Τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Επιπλέον είναι σχετικά απλή γλώσσα περιγραφής, η οποία δεν θα πρέπει να συγχέεται με τις γλώσσες προγραμματισμού. Σκοπός της γλώσσας HTML είναι να περιγράφει τη δομή και τις ιδιότητες των αντικειμένων των εγγράφων υπερκειμένου, ενώ μιας γλώσσας προγραμματισμού είναι να παράγει ένα εκτελέσιμο πρόγραμμα. H έκδοση 3.2 αναπτύχθηκε και προτυποποιήθηκε τον Ιανουάριο του 1997 αποκλειστικά από το W3C και έκτοτε αποτελεί την κύρια γλώσσα σήμανσης του διαδικτύου. Πλέον βρισκόμαστε στην έκδοση 5 [3].

### CSS

Η γλώσσα επικαλυπτόμενων στυλ μορφοποίησης CSS (Cascading Style Sheets) χρησιμοποιείται για τη δημιουργία οδηγιών μορφοποίησης τα οποία μπορούν να εφαρμοστούν σε σχεδόν οποιοδήποτε στοιχείο εγγράφου γραμμένο σε γλώσσα σήμανσης ( HTML, XML, κ.α. ). Δημιουργήθηκε για να καταφέρει να επιτρέψει τον διαχωρισμό της παρουσίασης και του περιεχομένου συμπεριλαμβανομένων πτυχών όπως την διάταξη, χρώμα και γραμματοσειρά. Η HTML καθορίζει δομή, όχι μορφή παρουσίασης [3].

### JavaScript

Η JavaScript είναι μια γλώσσα σεναριακού προγραμματισμού. Κατά την εκτέλεση της χρησιμοποιεί διερμηνέα (interpreter). Δημιουργήθηκε το Μάιο του 1995 από τον Brendan Eich [1][3]. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού (browser),ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται στην οθόνη. Κατάφερε να γίνει μια από τις δημοφιλέστερες γλώσσες προγραμματισμού στο διαδίκτυο εκτοπίζοντας τους αντιπάλους της σε τέτοιο βαθμό, που σπανίως χρησιμοποιούνται σήμερα. Πλέον χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων όπως έγγραφα PDF και μικρές εφαρμογές της επιφάνειας εργασίας [3]. Η δημοφιλής χρήση της και η ραγδαία εξέλιξη στο χώρο του διαδικτύου, οδήγησε στην ανάγκη να δημιουργηθούν βιβλιοθήκες καθιστώντας την γραφή και τον τρόπο χειρισμού του κώδικα της ευκολότερο.

### Bootstrap

Το Bootstrap αναπτύχθηκε το 2010 από τον Mark Otto και τον Jacob Thornton. Είναι ένα ελεύθερο, ανοιχτού κώδικα πλαίσιο εργασίας (framework) για τη σχεδίαση ιστοσελίδων (front-end) και διαδικτυακών εφαρμογών. Περιλαμβάνει βασικά πρότυπα σχεδίασης των HTMLκαι CSSγια την τυπογραφία, τις φόρμες, τα κουμπιά και άλλα στοιχεία διεπαφής, καθώς και προαιρετικές επεκτάσεις JavaScript.Η έκδοση 3 υποστηρίζει τις τελευταίες εκδόσεις των φυλλομετρητών Google Chrome, Mozilla Firefox, Internet Explorer, Opera και Safari [5].

### jQuery

Η jQuery είναι μια βιβλιοθήκη για την γλώσσα προγραμματισμού JavaScript σχεδιασμένη με στόχο να απλοποιήσει την υλοποίηση σεναρίων (scripting).Προσφέρει ένα εύκολο τρόπο να φέρεις εις πέρας ένα πλήθος από διεργασίες, με γρήγορο τρόπο και συνέπεια. Είναι ελεύθερο λογισμικό, ανοιχτού κώδικα και υποστηρίζεται από όλους τους φυλλομετρητές δίχως την εγκατάσταση πρόσθετων εφαρμογών. Αναπτύχθηκε από τον John Resig και κυκλοφόρησε τον Ιανουάριο του 2006. Χρησιμοποιείται σε πάνω από το 65% των 10.000 ιστοσελίδων με τη μεγαλύτερη επισκεψημότητα παγκοσμίως.[1][7]

### JSON

Το JSON (Java Script Object Notation) όπως υπονοεί και το όνομα της προσφέρει έναν εναλλακτικό τρόπο να δημιουργούνται στιγμιότυπα αντικειμένων και που μπορεί επίσης να ενεργήσει ως γενικής χρήσης σύνταξη για ανταλλαγή δεδομένων. Το JSON είναι μια απλή και συμπαγής σύνταξη για αντικείμενα της JavaScript. Αφού τα αντικείμενα εκφρασθούν σε JSON μπορούν εύκολα να μετατραπούν σε συμβολοσειρά που μπορούν να αποθηκευτούν και να μεταδοθούν [1].

### Plotly

Plotly επίσης γνωστή και ως Plot.ly είναι ένα διαδικτυακό εργαλείο ανάλυσης και οπτικοποίησης δεδομένων. Παρέχει διαδικτυακά εργαλεία για την σχεδίαση γραφημάτων, την ανάλυση στοιχείων και την στατιστική ανάλυση. Δημιουργήθηκε από τους Alex Johnson, Jack Parmer, Chris Parmer και Matthew Sundquist το 2012 και εδρεύει στο Montreal, Quebec [6].

### MVC Αρχιτεκτονική

Το Model–View–Controller (σε συντομογραφία αναφέρεται ως MVC) είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για την δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήση από την μορφή που έχει αποθηκευτεί στο σύστημα. Το κύριο μέρος του μοντέλου είναι το αντικείμενο Model το οποίο διαχειρίζεται την ανάκτηση/αποθήκευση των δεδομένων στο σύστημα. Το αντικείμενο View χρησιμοποιείται μόνο για να παρουσιάζεται η πληροφορία στον χρήστη (π.χ. με γραφικό τρόπο). Το τρίτο μέρος είναι ο Controller ο οποίος δέχεται την είσοδο και στέλνει εντολές στο αντικείμενο Model και στο View.

### PNG

### PNG

Το PNG (Portable Network Graphics) είναι μια μορφή αρχείου ράστερ (raster graphics image) η οποία υποστηρίζει συμπίεση δεδομένων χωρίς απώλειες. Αποτελεί την πιο ευρέως χρησιμοποιούμενη μορφή συμπίεσης εικόνων χωρίς απώλειες στο διαδίκτυο. Υποστηρίζει εικόνες βασισμένες σε παλέτες χρωμάτων, RGB 24 bit ή RGBA 32 bit χρωμάτων, εικόνες σε κλίμακα του γκρι ( με ή χωρίς το κανάλι A για διαφάνεια) και εικόνες RGB/RGBA σε πλήρη έγχρωμη μορφή χωρίς παλέτα. Εγκρίθηκε για την χρήση του από την ομάδα Internet Engineering Steering Group στις 14 Οκτωβρίου του 1996 και δημοσιεύθηκε ως πρότυπο ISO /IEC το 2004.

### LocalStorage

Με το local storage οι διαδικτυακοί τόποι μπορούν να αποθηκεύουν δεδομένα σε τοπικό επίπεδο, μέσα στον φυλλομετρητή. Οι ποσότητες πληροφορίας που μπορούν να αποθηκευτούν φτάνουν τουλάχιστον τα 5MB, δίχως να επηρεάζεται η απόδοση της ιστοσελίδας. Είναι ασφαλέστερα από τα cookies διότι δεν αποστέλλονται στον εξυπηρετητή (server). Όλες οι σελίδες από μία προέλευση (κοινή πηγή) μπορούν να έχουν πρόσβαση στα ίδια δεδομένα. Η λήξη τους πραγματοποιείται με την έξοδο από τον φυλλομετρητή.

# 

**Εισαγωγή**

Η ψηφιακή επεξεργασία εικόνας (ΨΕΕ) αποτελεί έναν ευρύ επιστημονικό κλάδο που αναπτύχθηκε με την ραγδαία εξέλιξη των υπολογιστών. Ο όρος εικόνα χρησιμοποιείται ευρύτερα από την απλή απεικόνιση ενός σκηνικού και περιλαμβάνει την αποτύπωση κάθε είδους πληροφοριών. Τα υπερηχογραφήματα, οι μαγνητικές τομογραφίες, οι δορυφορικές φωτογραφίες κ.α. μπορούν να επεξεργαστούν ως ψηφιακές εικόνες.

Οι στόχοι της ΨΕΕ είναι οι εξής:

1. Η ψηφιοποίηση και κωδικοποίηση εικόνων με σκοπό την αποθήκευση, μετάδοση και εκτύπωσή τους.
2. Η βελτίωση και η αποκατάσταση των εικόνων με σκοπό την καλύτερη απεικόνισή τους.
3. Η ανάλυση και κατανόηση των εικόνων.

## Ιστόγραμμα

Αν η ψηφιακή εικόνα αποτελείται από αποχρώσεις του γκρι μπορούμε να δημιουργήσουμε την κατανομή του πλήθους των εικονοστοιχείων που έχουν την ίδια τιμή απόχρωσης για κάθε απόχρωση. Η κατανομή αυτή λέγεται ιστόγραμμα των αποχρώσεων της εικόνας και δίνεται μαθηματικά από την σχέση



όπου *g=0,1,2..G-1*, *G* το πλήθος των αποχρώσεων, *j=0,...,J-1*, *k=0,...,K-1*, *J* το πλήθος των γραμμών, *K* το πλήθος των στηλών της εικόνας, *I(j,k)* η τιμή της φωτεινότητας στο σημείο *(j,k)* και *h(g)* το πλήθος των εικονοστοιχείων με απόχρωση *g*. Το ιστόγραμμα δηλαδή μας πληροφορεί πόσα εικονοστοιχεία υπάρχουν με συγκεκριμένη τιμή απόχρωσης. Η συνάρτηση



λέγεται κανονικοποιημένο ιστόγραμμα και δίνει την πιθανότητα ένα τυχαίο εικονοστοιχείο της εικόνας να έχει απόχρωση *g*.

## Εξισορροπημένο ιστόγραμμα

Η διάκριση γειτονικών περιοχών σε εικόνες αποχρώσεων του γκρι, είναι δύσκολη όταν η διαφορά τιμών των αποχρώσεων είναι μικρή. Η αύξηση των χρωματικών αντιθέσεων μιας εικόνας διευκολύνει την διάκριση των περιοχών αυτών. Μια τεχνική για την επίτευξη αυτού του σκοπού είναι η *εξισορρόπηση του ιστογράμματος* της εικόνας (*Histogram equalization)*. Σύμφωνα με αυτήν οι τιμές των αποχρώσεων των εικονοστοιχείων μεταβάλλονται έτσι ώστε να αυξηθεί η αντίθεση μεταξύ των περιοχών με διαδοχικές τιμές αποχρώσεων ανάλογα με το ποσοστό του πλήθους των εικονοστοιχείων τους. Συγκεκριμένα, αν *G* είναι το πλήθος όλων των αποχρώσεων της παλέτας του γκρι, *h(g)* το κανονικοποιημένο ιστόγραμμα της αρχικής εικόνας *ΙK*X*J*, *g=0,1,…,G-1* και *P(g)* η συνάρτηση αθροιστικής πιθανότητας του *g*, προκύπτει μία νέα εικόνα *Ι΄* ίδιων διαστάσεων με τη *Ι*, κάθε εικονοστοιχείο (*k,j*) της οποίας έχει απόχρωση *I΄(k,j) =* [(*G-1*)*P(I(k,j)*]. Με άλλα λόγια κάθε τιμής *g* μιας απόχρωση της αρχικής εικόνας αντικαθίσταται από μια τιμή *g΄ =* [(*G-1*)*P(g)*]. Υπενθυμίζεται ότι για την *P(g)* ισχύουν οι σχέσεις

 άρα και *P*(*g*) = *P*(*g-*1)+ *h*(*g*).

Παράδειγμα: Έστω ο ακόλουθος πίνακας τιμών των αποχρώσεων του γκρι μιας ψηφιοποιημένης εικόνας:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 10 | 14 | 14 | 14 |
| 14 | 14 | 25 | 25 | 25 |
| 25 | 26 | 27 | 27 | 27 |
| 25 | 25 | 25 | 25 | 26 |

Οι αποχρώσεις *g΄* προκύπτουν σύμφωνα με τους ακόλουθους υπολογισμούς:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***g*** | ***H*(*g*)** | ***h*(*g*)** | ***P*(*g*)** | ***g΄*** |
| 10 | 2 | 2/20 | 2/20 | 255\*2/20 = **25** |
| 14 | 5 | 5/20 | 7/20 | 255\*7/20 = **89** |
| 25 | 8 | 8/20 | 15/20 | 255\*15/20 = **191** |
| 26 | 2 | 2/20 | 17/20 | 255\*17/20 = **217** |
| 27 | 3 | 3/20 | 20/20 | 255\*20/20 = **255** |

Για να χρησιμοποιηθούν όλες οι αποχρώσεις από 0 έως 255 η τελική απόχρωση προκύπτει σύμφωνα με τη σχέση:

 όπου Pm=min{P(g)} και r(∙) η συνάρτηση στρογγυλοποίησης

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **g** | **H(g)** | **h(g)** | **P(g)** |  | **g'** |
| 10 | 2 | 0,1 | 0,1 | 0,00 | 0 |
| 14 | 5 | 0,25 | 0,35 | 0,28 | 71 |
| 25 | 8 | 0,4 | 0,75 | 0,72 | 184 |
| 26 | 2 | 0,1 | 0,85 | 0,83 | 213 |
| 27 | 3 | 0,15 | 1 | 1,00 | 255 |
|  | 20 | 1 |  |  |  |

Ιστόγραμμα h(g)

Εξισορροπημένο ιστόγραμμα h(g*΄*)

Στο παρακάτω σχήμα φαίνεται η εφαρμογή της εξισορρόπησης του ιστογράμματος στην εικόνα με τις πιπεριές.

|  |  |
| --- | --- |
| pepsgray | pepseqlz |
| **pepshst** | pepseqh |

Παρακάτω παρουσιάζεται η διαδικάσια δημιουργίας του ιστογράμματος υλοποιημένη σε JavaScript. Δέχεται ως όρισμα πίνακα ακεραίων 256 στοιχείων και επιστρέφει τον πίνακα αθροιστικής πιθανότητας.

**function** eqHistogram( hst)  
{  
**var** sum = 0.0, prob = **new** Array(256) ;  
**var** g ;  
  
**for**(g = 0; g <256 ; g++)   
 {  
 sum += hst[ g];  
 }  
 prob[ 0] = hst[ 0] / sum ;  
**for**(g = 1; g <256 ; g++)

{  
 prob[ g] = prob[ g-1] + hst[ g] / sum ;

}  
  
**return** prob;  
}

## Κατωφλίωση με βάση την διασπορά

Μια μέθοδος για την εύρεση της τιμής κατωφλίου Τ είναι η μέθοδος του Οtsu. Σύμφωνα με αυτήν η τιμή *Τ* πρέπει να είναι τέτοια ώστε η συνολική εσωτερική διασπορά  εντός των κλάσεων να είναι ελάχιστη και η συνολική διασπορά  μεταξύ των κλάσεων μέγιστη. Σύμφωνα με την μέθοδο δημιουργούμε το κανονικοποιημένο ιστόγραμμα *h(g)* της εικόνας. Η συνάρτηση *h(g)* δίνει την πιθανότητα να έχει ένα τυχαίο εικονοστοιχείο της εικόνας απόχρωση *g.*

Έστω Τ μια τιμή κατωφλιού που διαχωρίζει τις αποχρώσεις σε δύο κλάσεις *C1* (*g<T*) και *C2* (*T≤g*). Η πιθανότητα ένα τυχαίο εικονοστοιχείο να έχει απόχρωση της κλάσεως *C1* είναι *p1* και δίνεται από τη σχέση



όμοια για την κλάση *C2* είναι *p2* και δίνεται από τη σχέση



Είναι προφανές ότι ισχύει

*p1+p2=1.*

Η μέση τιμή *μ1* της πρώτης κλάσης είναι



Η μέση τιμή *μ2* της δεύτερης κλάσης είναι και ισούται με



Η συνολική μέση τιμή μ δίνεται από τη σχέση



Από τις τρείς προηγούμενες σχέσεις εύκολα αποδεικνύεται η σχέση

*μ=p1μ1+p2μ2*

Η διασπορά της πρώτης κλάσης δίνεται από τη σχέση



και η διασπορά της δεύτερης κλάσης είναι



H συνολική διασπορά  είναι



Η συνολική εσωτερική διασπορά των κλάσεων ορίζεται από τη σχέση



Η συνολική διασπορά μεταξύ των κλάσεων ορίζεται από τη σχέση



Αναζητούμε τώρα την κατάλληλη τιμή του Τ ώστε η  να είναι ελάχιστη και μέγιστη. Προς τούτο θεωρούμε το πηλίκο



και ζητάμε την τιμή του Τ για την οποία το λ μεγιστοποιείται.

Για να μειωθούν οι υπολογισμοί θεωρούμε τις σχέσεις



Επειδή το *σ2* είναι ανεξάρτητο του *Τ* και η συνάρτηση λ/(1+λ) είναι αύξουσα συνεπάγεται πως για να είναι το *λ* μέγιστο αρκεί να είναι το  μέγιστο. Με τον τρόπο αυτό η τελικά η κατάλληλη τιμή του κατωφλίου *Τ* είναι εκείνη που μεγιστοποιεί την ποσότητα



Παρακάτω παρουσιάζεται η διαδικάσια κατωφλίωσης με βάση την διασπορά, υλοποιημένη σε JavaScript. Δέχεται ως όρισμα πίνακα ακεραίων 256 στοιχείων και επιστρέφει την τιμή κατωφλιού.

**function** otsu(h)  
{  
  
**var** m, m1, p1, sum, sb=**new** Array(256), max , g, T, Tf, Tl, Tb ;  
**for**( T = 0; T <256; T++ ) {sb[ T] = 0.0 ;}  
  
sum= 0.0 ;

**for**( g = 0; g <256; g++ ) sum += h[ g] ;*// or sum = im.J \* im.K*m = 0.0 ;*// mean value and nomarlized histogram computation.***for**( g = 0; g <256; g++)  
 {  
 h[ g]/=sum ; *// Histogramm normalization.* m += h[ g]\*g ;   
 }   
  
**for**( g = 0 ; h[ g] == 0.0 ; g++){}  
 Tf = g ; *// 1st non zero h(g) value.***for**( g =255 ; h[ g] == 0.0 ; g--){}  
 Tl = g ; *// last non zero h(g) value.***if**( Tf == Tl ) **return** (-1); *// invalid histogram containig zero values  
  
 // Δεν είναι απαραίτητη για τον αλγόριθμο.***for**( T = Tf +1; T <= Tl ; T++ )   
 {  
 **if**( h[ T] != 0.0 )  
 {  
 p1 = 0.0 ; m1 = 0.0 ;  
  
 **for**(g = Tf ; g <T; g++)   
 {  
 p1 = p1 + h[ g] ; m1 += h[ g] \* g ;  
  
 }  
 m1 = m1 / p1 ;  
 sb[ T ] = p1\*(m-m1)\*(m-m1)/(1-p1) ;  
   
 }  
 }  
  
 Tb = Tf+1 ; max = sb[ Tb] ;  
**for**( T = Tf + 1 ; T < Tl ; T++ )  
 {  
  
 **if**( sb[ T] > max ){ max = sb[ T] ; Tb = T ; }  
 }  
  
 **return** Tb ;  
}

## Κατωφλίωση με βάση την εντροπία

Μια διαφορετική προσέγγιση για την εύρεση κατάλληλης τιμής κατωφλίου Τ βασίζεται στην έννοια της εντροπίας. Σύμφωνα με την εκδοχή που παρουσίασε ο Kapur, η τιμή Τ χωρίζει το κανονικοποιημένο ιστόγραμμα h(g) σε δυο κλάσεις c1 και c2. Η πρώτη κλάση μπορεί να θεωρηθεί ως πηγή πληροφορίας με σύμβολα g = 0, …Τ-1 και αντίστοιχες πιθανότητες



όπου



Η μέση πληροφορία (εντροπία) που προκύπτει της c1 είναι



Όμοια η μέση πληροφορία που προκύπτει από την c2 είναι



Ζητείται η πρώτη κλάση που θα αντιστοιχηθεί στην τιμή ‘0’ και η δεύτερη που θα αντιστοιχηθεί στο ‘1’ να προσφέρουν συνολικά την μεγαλύτερη δυνατή μέση πληροφορία. Δηλαδή η ζητούμενη τιμή του κατωφλίου Τα είναι εκείνη για την οποία Ε1(Τ)+Ε2(Τ) γίνεται μέγιστο.

Παρακάτω παρουσιάζεται η διαδικασία κατωφλίωσης με βάση την εντροπία, υλοποιημένη σε JavaScript. Δέχεται ως όρισμα πίνακα ακεραίων 256 στοιχείων και επιστρέφει την τιμή κατωφλιού.

**function** kapur(h)  
{  
**var** p1, sum, He1, He2, He=**new** Array(256), max ;  
**var** g, T, Tf, Tl, Tb ;  
**for**( T = 0; T <256; T++ )  
 {  
 He[ T] = 0.0 ;  
 }  
 sum= 0.0 ; **for**( g = 0; g <256; g++ ) sum += h[ g] ;  
  
**for**( g = 0; g <256; g++)  
 {  
 h[ g]/=sum ;  
 }  
**for**( g = 0 ; h[ g] == 0.0 ; g++){}  
 Tf = g ; *// 1st non zero h(g) value.***for**( g =255 ; h[ g] == 0.0 ; g--){}  
 Tl = g ; *// last non zero h(g) value.***if**( Tf == Tl ) **return** 0; *//No threshold exists for a monochrome image.*

**for**( T = Tf + 1 ; T <= Tl ; T++ )  
 {  
 **if**( h[ T] != 0.0 )  
 {  
 p1 = 0.0 ;  
 **for**(g = Tf ; g < T; g++)  
 {  
 p1 = p1 + h[ g] ;  
 }  
 He1 = 0.0 ;  
 **for**(g = Tf ; g < T; g++)  
 {  
 **if**( h[ g] >0.0)  
 {  
 He1 += h[ g] \* Math.log( h[ g]) ;  
 }  
 }  
 He1 = Math.log(p1) - 1/p1 \* He1 ;  
 He2 = 0.0 ;  
 **for**(g = T ; g <=Tl; g++)  
 {  
 **if**( h[ g] >0.0)  
 {  
 He2 += h[ g] \* Math.log( h[ g]) ;  
 }  
 }  
 He2 = Math.log(1-p1) - 1/(1-p1) \* He2 ;  
 He[ T ] = He1 + He2 ;  
 }  
 }  
 Tb = Tf+1 ;  
 max = He[ Tb] ;  
**for**( T = Tf + 1 ; T < Tl ; T++ )

{  
 **if**( He[ T] > max ){ max = He[ T] ; Tb = T ; }  
 }  
  
 **return** Tb ;  
}

## Πολυκατωφλίωση με χρήση νευρωνικού δικτύου

Εάν κάθε αντικείμενο της εικόνας αποτελείται από μία κύρια και κάποιες παραπλήσιες αποχρώσεις του γκρι , τότε δημιουργείται η συσσώρευση των τιμών του ιστογράμματος σε συγκεκριμένες περιοχές. Η επιλογή πολλαπλών κατωφλίων μπορεί να θεωρηθεί ως ένα πρόβλημα εύρεσης τιμών  με σκοπό το μετασχηματισμό της αρχικής εικόνας με *L* αποχρώσεις του γκρι σε μια νέα με *J* αποχρώσεις. Ειδικότερα για *T(1)<T(2)<…<T(J-1),* τότε η νέα εικόνα ορίζεται ως



όπου *G* μία συνάρτηση τιμών των επιπέδων του γκρι. Μία τεχνική πολλαπλών κατωφλίων που βασίζεται στο ιστόγραμμα πρέπει να πετυχαίνει έναν ικανοποιητικό διαχωρισμό των διαφόρων περιοχών του ιστογράμματος. Οι μέθοδοι της διασποράς και της εντροπίας είναι δυνατόν να εφαρμοσθούν και στην επίλυση του προβλήματος της πολυκατωφλίωσης, έχουν όμως μεγάλο υπολογιστικό κόστος.

Μία αποτελεσματική προσέγγιση είναι να θεωρήσουμε την εύρεση πολλαπλών κατωφλίων ως πρόβλημα εύρεσης κλάσεων (clustering problem). Ένα κατάλληλο νευρωνικό δίκτυο μπορεί να λύσει αποτελεσματικά προβλήματα εύρεσης κλάσεων. Είναι γνωστό ότι ο κύριος σκοπός ενός νευρωνικού δικτύου *Kohonen* για την δημιουργία ενός αυτό-οργανωμένου πίνακα απεικόνισης χαρακτηριστικών (SOFM: Self Organized Feature Map) είναι η αντιπροσώπευση ενός πολυπληθούς συνόλου διανυσμάτων εισόδου με ένα ολιγομελέστερο σύνολο “πρωτοτύπων ανυσμάτων” ώστε να επιτευχθεί μια καλή προσέγγιση του αρχικού χώρου εισόδου που να ικανοποιεί τα κύρια στατιστικά χαρακτηριστικά του. Θεωρούμε ένα δίκτυο με νευρώνες (κόμβους) όπως στο παρακάτω σχήμα.

*w0*

*w1*

*wj-2*

**I**

x

*wj-1*

y

Γειτονιά του *j* νευ-

ρώνα για *d*(*t*)=2

*wj*

I(x,y)

*wj+1*

*wj+2*

*wJ-1*

* Συγκροτείται το σύνολο εκπαίδευσης του ΑΠΑΧ από τις τιμές των αποχρώσεων των εικονοστοιχείων της εικόνας.
* Θεωρούμε τη μεταβλητή επανάληψης *t* που παίρνει ακέραιες τιμές από 0 μέχρι μία προκαθορισμένη τελική τιμή *Τ* (π.χ. *Τ=*100.000).
* Θεωρούμε τη μεταβλητή του ρυθμού εκμάθησης *α(t).*
* Θεωρούμε το μήκος *d(t)* που ορίζει μια υποπεριοχή (γειτονιά) στην διάταξη των νευρώνων.
* Έστω w*j*το βάρος της σύναψης μεταξύ του *j* νευρώνα του επιπέδου ανταγωνισμού και της εισόδου του.

Εκτελούνται τα παρακάτω βήματα:

1. Αρχικοποιούνται (*t=0*) τα βάρη των συνάψεων *wj(0)* με τυχαίες τιμές από 255.
2. Αρχικοποιείται η *α(0)* με μια μεγάλη τιμή, συνήθως μεταξύ 0.2 και 0.5.
3. Αρχικοποιείται η *d(0)* με την τιμή J/2, που είναι ίση με το μισό του εύρους του κανάβου.
4. Επιλέγεται τιμή απόχρωσης *Ι(x,y)* ενός τυχαίου εικονοστοιχείου από το σύνολο εκπαίδευσης.
5. Υπολογίζεται η έξοδος *oj(t)*κάθε νευρώνα από τη σχέση

.

1. Ο νευρώνας *c* ανακηρύσσεται νικητής εάν ικανοποιείται η συνθήκη

*oc(t) = min{oξ(t)}.*

Εάν οι έξοδοι δύο νευρώνων είναι ίσες, τότε κατά σύμβαση επιλέγεται αυτός με το μικρότερο δείκτη.

1. Tα βάρη w*j* των συνάψεων ανανεώνονται σύμφωνα με τις παρακάτω σχέσεις



*wj(t+1)=wj(t)+Δwj(t)*

όπου *Nc* το σύνολο των δεικτών των νευρώνων που βρίσκονται μέσα στην γειτονιά του νικητή νευρώνα και πλευρά *d(t)*.

1. Αυξάνεται η μεταβλητή επανάληψης κατά ένα και αποδίδονται νέες τιμές στις μεταβλητές *α(t), d(t)* σύμφωνα με τις σχέσεις:





Τα βήματα 4 έως 8 επαναλαμβάνονται έως ότου η μεταβλητή *t* πάρει τη μέγιστη τελική τιμή *Τ*. Είναι φανερό πως οι μεταβλητές *α(t)* και *d(t)* συγκλίνουν στο μηδέν καθώς η *t* τείνει στην τιμή *Τ.*

Μετά την εκπαίδευση κάθε άνυσμα εισόδου του ΑΠΑΧ αποδίδεται στον νικητή νευρώνα. Κάθε νευρώνας του επιπέδου εξόδου αντιπροσωπεύει μία ομάδα προτύπων (cluster). Πρότυπα με μεγάλη ομοιότητα αντιπροσωπεύονται από τον ίδιο νευρώνα. Ο χάρτης χαρακτηριστικών του Kohonen οργανώνει τους νευρώνες του επιπέδου ανταγωνισμού με τέτοιο τρόπο ώστε οι ομοιότητες μεταξύ των προτύπων να απεικονίζονται με σχέσεις γειτνίασης επάνω στον κάναβο του επιπέδου ανταγωνισμού

Παρακάτω φαίνεται η αρχική εικόνα και το ιστόγραμμα της στην οποία θα εφαρμοσθεί πολυκατωφλίωση



Μετά την εκπαίδευση ενός ΝΔ *Kohonen* για την πολυκατωφλίωση των νευρώνων προκύπτουν οι τιμές των βαρών που αναγράφονται στον Πιν. 2.5.1 και οι τιμές των κατωφλίων του Πιν.2.5.2. Οι τιμές αυτές αποτελούν ουσιαστικά ένα μικρότερο πλήθος αποχρώσεων για την απόδοση της αρχικής εικόνας.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *W0* | *w1* | *W2* | *w3* | *W4* | *W5* | *W6* | *w7* |
| 69 | 111 | 153 | 200 | 218 | 231 | 240 | 246 |

Πίνακας 2.5.1: Οι τιμές των συντελεστών του ΝΔ μετά την εκπαίδευση.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *T0* | *T1* | *T2* | *T3* | *T4* | *T5* | *T6* |
| 90 | 132 | 176 | 209 | 224 | 235 | 243 |

Πίνακας 2.5.2 Οι τιμές των κατωφλίων.

Στον Πιν. 2.5.3 φαίνονται οι τιμές των κατωφλίων για διαφορετικό πλήθος (*J*) νευρώνων του νευρωνικού δικτύου

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *J* | *T0* | *T1* | *T2* | *T3* | *T4* |
| 2 | 164 |  |  |  |  |
| 3 | 151 | 227 |  |  |  |
| 4 | 106 | 179 | 229 |  |  |
| 5 | 106 | 178 | 224 | 237 |  |
| 6 | 97 | 143 | 191 | 225 | 239 |

Πίνακας 2.5.3. Οι τιμές των κατωφλίων για *J=2,…,6*



J=2 J=3



J=4 J=5

Σχήμα 2.5.1. Τα ιστογράμματα και οι αντίστοιχες τιμές κατωφλίων για διαφορετικές τιμές του *J*.

Πλήθος αποχρώσεων 2 Πλήθος αποχρώσεων 3

Πλήθος αποχρώσεων 4 Πλήθος αποχρώσεων 5



Σχήμα 2.5.2:Οι εικόνες που προέκυψαν από την πολυκατωφλίωση της εικόνας.

Η βέλτιστη τιμή του *J* για ένα δεδομένο σύνολο τιμών {2,…,*Jmax*} ελαχιστοποιεί το σφάλμα προσαρμογής μεταξύ των συναρτήσεων *H(**)* και *HJ(**)*. Η συνάρτηση *H(**)* υπολογίζεται από τη σχέση

 (3.2.3.8)

και είναι συνάρτηση πυκνότητας πιθανότητας επειδή .

Η πιθανότητα *fj* εμφάνισης της κλάσης *Cj*, η μέση τιμή *μj* της κλάσης και η διασπορά της  δίνονται αντίστοιχα από τις σχέσεις







Η συνάρτηση προσαρμογής  είναι το άθροισμα



όπου



Μπορεί να δειχθεί εύκολα ότι η  είναι επίσης συνάρτηση πυκνότητας πιθανότητας. Το σφάλμα προσαρμογής  για ένα πλήθος κλάσεων *J* ορίζεται ως



και υπολογίζεται για *J*=2,…,*Jmax*. Η τιμή *J*o είναι βέλτιστη όταν το αντίστοιχο σφάλμα προσαρμογής  ικανοποιεί τη συνθήκη

=*minimum*{}

Ο Πίν. 2.5.3 δείχνει τις τιμές του  για *J=*2,…,6. Η μικρότερη από αυτές τις τιμές προκύπτει για *J=3* και θεωρείται ότι είναι η βέλτιστη τιμή *Jo*. Όπως μπορεί να παρατηρηθεί από το ιστόγραμμα  (Σχ. 2.5.1), αυτή η τιμή είναι κατάλληλη επειδή αντιστοιχεί στο πλήθος των σημαντικών υψωμάτων του ιστογράμματος.

Παρακάτω παρουσιάζεται η διαδικασία πολυκατωφλίωσης με χρήση νευρωνικού δικτύου, υλοποιημένη σε JavaScript. Δέχεται δύο ορίσματα. Το πρώτο όρισμα είναι ένα αντικείμενο εικόνας και το δεύτερο όρισμα είναι ένας ακέραιος αριθμός, στο διάστημα 1-256, όπου εκφράζει το πλήθος των νευρώνων. Επιστρέφει τα βάρη.

**function** kohonen1d(im, neuronNum)  
{   
*// var neuron = new Array(neuronNum);***var** w=[neuronNum],x=[neuronNum],out=[neuronNum];  
**var** a0 = 0.4, a = a0; *// learning rate parameters.***var** d0 = neuronNum/2.0, d=d0; *// vicinity radius.***var** i, c ; *// neuron indexes.***var** g ; *// gray level variable.***var** epochs = 4, t, T;  
  
*// Neuron initializations.***for**(i = 0; i < neuronNum; i++)  
 {

w[i] = Math.floor((Math.random() \* 256));  
 x[i] = i;   
 out[i]=0;  
 }  
  
*// Training stage.*T = epochs \*(im.data.length/4) ;   
**for**( t = 0; t<T; t++)  
 {  
*// input selection  
 // var k=Math.floor(Math.random()\*K);  
 // var j=Math.floor(Math.random()\*J);* **var** pos=4\*Math.floor(Math.floor(Math.random() \* im.data.length)/4);  
 g = im.data[pos];  
  
 *// neuron output computation.* **for**(i = 0; i < neuronNum; i++) {out[i] = Math.abs(w[i] - g);}  
  
 *// winner computation.* c=0;  
 **for**(i=0; i<neuronNum; i++)  
 {  
 **if**( out[i] < out[c]){c=i;}  
 }  
 *// winner update.* **for**(i = 0; i < neuronNum; i++)  
 **if**(Math.abs(x[c]-x[i]) <d) w[i] +=a\*(g-w[i]);  
  
 *// a , d variable updating.* a = a0 \* (1.0 - t / T) ; d = d0 \* (1.0 - t/T);  
 }  
*// Neuron sorting according weight value.*w.sort(**function**(a,b) { **return** a - b; });  
*/\* the following code is optional for cases of n variable computation  
 // Classification stage  
 for( int j = 0; j < im.J; j++)  
 for( int k = 0; k < im.K; k++)  
 { g = im.pel[ j][ k] ;*

*// neuron output computation  
 for(i = 0; i < neuronNum; i++) neuron[ i].out = fabs(neuron[ i].w - g);  
// winner computation  
 c=0; for(i=0; i<neuronNum; i++) if( neuron[ i].out < neuron[ c].out)c=i;  
 neuron[ min].n++;  
 }\*/***for**(i=0;i<neuronNum;i++){w[i]=Math.floor(w[i]);}  
**return** w ;  
}

# 

## Τεχνικά θέματα υλοποίησης

Η ιστοσελίδα προβάλλεται μέσω κάποιου φυλλομετρητή, γεγονός που περιορίζει ένα μέρος των διαθέσιμων πόρων του εκάστοτε υπολογιστή. Παρόλα αυτά δεν απαιτούνται υψηλά τεχνικά χαρακτηριστικά για την λειτουργία της. Λειτουργεί ορθά και ανταποκρίνεται βέλτιστα σ’ ένα μέσο ηλεκτρονικό υπολογιστή. Υποστηρίζει ποικίλους τύπους αρχείων εικόνας και μέγεθος αρχείου μέχρι 2 MB. Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε στο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) PhpStorm της εταιρείας JetBrains και δοκιμάστηκε στους δυο πιο δημοφιλείς φυλλομετρητές του έτους 2018, Google Chrome και Mozilla Firefox. Επιπροσθέτως υποστηρίζει την λειτουργία από κινητές συσκευές και tablets. Για να την σχεδίαση της ιστοσελίδας χρησιμοποιήθηκε η αρχιτεκτονική MVC.

## Βασική δομή της εφαρμογής

Στο σημείο αυτό θα αναφερθούμε σε βασικές αλγοριθμικές έννοιες της εφαρμογής που χρήζουν επεξήγηση.

Το αντικείμενο gray δημιουργείται με την φόρτωση της ιστοσελίδας. Τα δεδομένα που περιέχει τα λαμβάνει από την συνάρτηση uploadImage. Συγκεκριμένα δέχεται τον αριθμητικό πίνακα ακεραίων histogram και το αντικείμενο εικόνας imgGray. Ο histogram περιέχει τιμές για την δημιουργία ιστογράμματος, ενώ ο imgGray πληροφορίες εικόνας στην κλίμακα του γκρι. Με λίγα λόγια το αντικείμενο gray διατηρεί τα σημαντικότερα δεδομένα που χρειαζόμαστε για να επεξεργαστούμε μία ψηφιακή εικόνα.

Η εφαρμογή δέχεται ένα ευρύ φάσμα τύπων εικόνας όπως επίσης και ποικίλων χρωματικών παλετών. Κατά την φόρτωση μίας εικόνας στην εφαρμογή οι διαδικασίες που εκτελούνται στο προσκήνιο είναι α) ο έλεγχος ορθότητας των εισαχθέντων δεδομένων και β) η αποθήκευση αυτών, όπως και άλλων σημαντικών πληροφοριών, για τις μετέπειτα εφαρμογές αλγορίθμων ΨΕΕ. Παρακάτω παρουσιάζεται το σενάριο που εκτελείται κάθε φορά που ένας χρήστης φορτώνει μία εικόνα στην εφαρμογή. Το σενάριο επιτυγχάνει:

α) Να ελέγξει αν το αρχείο που φορτώθηκε είναι εικόνα. Διαφορετικά εμφανίζει μήνυμα σφάλματος στην οθόνη του χρήστη.

β) Έπειτα στέλνει το αρχείο για μετατροπή στη χρωματική κλίμακα του γκρι. Την διαδικασία αυτή αναλαμβάνει το σενάριο *grayscale* (imToGray) επιστρέφοντας δύο πίνακες. Έναν στη μεταβλητή histogram, ο οποίος περιέχει τις τιμές του ιστογράμματος της εικόνας, για κάθε χρωματική τιμή στην κλίμακα του γκρι. Επιπλέον ένα αντικείμενο εικόνας, στη μεταβλητή imgGray, με τις χρωματικές τιμές των εικονοστοιχείων στην κλίμακα του γκρι.

γ) Οι δύο παραπάνω πίνακες εκχωρούνται μέσα στο αντικείμενο gray το οποίο δημιουργείται με την φόρτωση της σελίδας. Ακολούθως το σενάριο αποθηκεύει τα δεδομένα που έχει λάβει στο local storage. Τα δεδομένα θα χρησιμοποιηθούν για την εφαρμογή οποιουδήποτε αλγορίθμου επιλέξει ο χρήστης να εκτελέσει.

δ) Γνωρίζοντας πως η γλώσσα προγραμματισμού JavaScript φορτώνει τα σενάρια ασύγχρονα, δημιουργεί μία σταθερή δομή για την εισαγωγή των δεδομένων μειώνοντας τις περιπτώσεις σφάλματος.

**function** uploadImage(input) *// Το όρισμα που δέχεται η συνάρτηση αφορά στο αρχείο που έχει επιλέξει ο χρήστης.*{  
**if** (input) *// Έλεγχος αν υπάρχει αρχείο.*{  
**if** (!(input.files[0].type.match(**'image.\*'**))) *// Έλεγχος αν το αρχείο είναι εικόνα.*{  
 input.value = **''**;  
 $(**"#error-msg"**).html(**"Is not an image"**);   
 snackBar();*// Εκτυπώνει μήνυμα σφάλματος στην οθόνη του χρήστη.*  
  
 } **else**{  
 showLoader(); *// Γραφικό στοιχείο. Εμφανίζει μια ροδέλα καθώς ο χρήστης περιμένει την φόρτωση του αρχείου.***var** reader = **new** FileReader(); *// Δημιουργεί έναν file Reader ο οποίος ενεργοποιείται ασύγχρονα από τον φυλλομετρητή  
//* [*https://developer.mozilla.org/en-US/docs/Web/API/FileReader*](https://developer.mozilla.org/en-US/docs/Web/API/FileReader) *.*reader.onload = **function** preview() *// Η συνάρτηση θα εκτελεσθεί μόλις εκτελεσθεί η εντολή reader.readAsDataURL(input.files[0]) στο τέλος της συνάρτησης.*{  
 addStorItem(**"imgData"**, reader.result); *// Αποθήκευση δεδομένων στο Local Storage για μεταγενέστερη χρήση.*  
  
*// Η εικόνα που έχει επιλέξει ο χρήστης ακόμα δεν έχει εμφανιστεί στην οθόνη. Αν προσπαθήσουμε να την γραπώσουμε από την θέση της (top-left-image) τότε θα παρουσιαστεί πρόβλημα. Χρησιμοποιούμε τη μέθοδο addEventListener η οποία θα ενεργοποιήσει την συνάρτηση που έχει μέσα της όταν εμφανιστεί η εικόνα στην οθόνη.*document.getElementById(**'top-left-image'**).addEventListener(**'load'**,**function**()  
 { *// Η grayscale βρίσκεται στο αρχείο grayscale.js. Θα πάρει ως //όρισμα την εικόνα που επέλεξε ο χρήστης και θα την μετατρέψει σε //ασπρόμαυρη στο παρασκήνιο. Επίσης θα παράξει τον πίνακα με τις αποχρώσεις τις εικόνας.*gray = grayscale(document.getElementById(**'top-left-image'**));  
 showPage();  
 },**false**);  
 clear(**'upload'**);  
 showImage(reader.result, **'top-left-image'**);  
  
**if**(getStorItem(**"imgData"**)==**null**) *// Έλεγχος αν η εικόνα δεν επεξεργάσθηκε λόγο μεγέθους.*{  
 showPage();  
 clear(**'all'**);  
 $(**"#error-msg"**).html(**"image is big"**);  
 snackBar();  
 }**else** *// Αν η επεξεργασία ήταν επιτυχής θα εμφανιστεί το παρακάτω μήνυμα.*{  
 setTimeout(**function** ()  
 {  
 $(**"#error-msg"**).html(**"The colour images are converted<br> into gray level in the background processing"**);  
 snackBar();  
 }, 0);  
 }  
 };  
 reader.readAsDataURL(input.files[0]); *// Μόλις διαβάσει το αρχείο η ιδιότητα result θα περιέχει τα δεδομένα του αρχείου σε URL. Επίσης θα ξεκινήσει η συνάρτηση preview().  
 //https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readAsDataURL.*}  
 }  
}

Παρακάτω παρουσιάζεται το σενάριο *grayscale* (imToGray) για το οποίο έγινε αναφορά προηγουμένως. Δέχεται ως όρισμα ένα στοιχείο εικόνας (έγχρωμο ή ασπρόμαυρο). Έπειτα δημιουργεί έναν κενό καμβά ίδιων διαστάσεων με την εικόνα που επιλέχθηκε ως όρισμα στη συνάρτηση. Η μέθοδος drawImage αντιγράφει την εισαχθείσα εικόνα στον νεοσύστατο καμβά, στη μορφή PNG. Μ αυτήν την διαδικασία καταφέρνουμε να δουλέψουμε με διαφορετικούς τύπους εικόνας (bitmap, JPEG, κτλ) και να εφαρμόσουμε αλγορίθμους ΨΕΕ σ αυτούς. Πλέον γνωρίζουμε πως κάθε φορά που συλλέγουμε στοιχεία από τον καμβά, μπορούμε να τα δουλέψουμε με τον ίδιο τρόπο.

Την μετατροπή της έγχρωμης εικόνας στην χρωματική κλίμακα του γκρι αναλαμβάνει η μέθοδος getImageData. Η μέθοδος αυτή διαβάζει το κάθε εικονοστοιχείο ως συνδυασμό 4 επιπέδων. Το κάθε επίπεδο εκφράζει μια χρωματική τιμή στο χρωματικό μοντέλο RGBA, όπου Κόκκινο, Πράσινο, Μπλε και Αδιαφάνεια. Όλα τα επίπεδα μαζί αποτελούν την σύνθεση του κάθε εικονοστοιχείου. Η εξίσωση με την οποία επιτυγχάνεται ο αποχρωματισμός της εικόνας είναι η εξής:

Κλίμακα Γκρι = Κοκ \* 0.2126 + Πρασ \* 0 .7152 + Μπλε \* 0.0722

Τέλος δημιουργείται ο πίνακας histogram και η συνάρτηση επιστρέφει το αντικείμενο εικόνας imgGray και τον πίνακα histogram στη συνάρτηση που αναλύσαμε παραπάνω.

**function** *grayscale*(imToGray)  
{  
  
**var** cnv = **document**.createElement(**'canvas'**), *// Δημιουργία καμβά.*ctx = cnv.getContext(**'2d'**);  
  
cnv.**width** = imToGray.**naturalWidth**; *// Φυσικές διαστάσεις εικόνας.*cnv.**height** = imToGray.**naturalHeight**;  
ctx.drawImage(imToGray, 0, 0); *// Αποτύπωση της εικόνας στον καμβά.***var** imgGray = ctx.getImageData(0, 0, cnv.**width**, cnv.**height**) *// Συλλογή δεδομένων της εικόνας από τον καμβά.*, histogram = **new** Array(256) *// Ο πίνακας που θα φιλοξενήσει τις τιμές για το ιστόγραμμα.*, i  
, red  
, green  
, blue  
 , graysc;  
**for** (i = 0; i <256; ++i)  
histogram[i] = 0;  
**for** (i = 0; i <imgGray.**data**.**length**; i += 4)  
 {  
red = imgGray.**data**[i]; *// Πρώτο στοιχείο του pixel κόκκινο.*green = imgGray.**data**[i + 1]; *// Δεύτερο στοιχείο του pixel πράσινο.*blue = imgGray.**data**[i + 2];  
*// Η αδιαφάνεια δεν μας χρειάζεται σ αυτήν την περίπτωση.  
 // alpha = imgGray.data[i + 3];  
 // Τρόπος μετατροπής της έγχρωμης εικόνας σε ασπρόμαυρη υπάρχει στο παρακάτω σύνδεσμο  
 // https://en.wikipedia.org/wiki/Grayscale.*graysc = red \* .2126 + green \* .7152 + blue \* .0722; *// or Y=0.299R+0.587G+0.114B*histogram[**Math**.round(graysc)] += 1;  
imgGray.**data**[i]=imgGray.**data**[i+1]=imgGray.**data**[i+2]=**Math**.round(graysc);  
imgGray.**data**[i + 3]=255;  
 }  
  
**return**{  
**histogram**: histogram,  
**imgGray**: imgGray  
};  
}

Σημαντική μπορεί να φανεί η επικοινωνία με το local storage κατά την ανάπτυξη του αλγορίθμου ΨΕΕ που θα θελήσουμε να ενσωματώσουμε στην εφαρμογή. Η addLocItem(key, value) αποθηκεύει, με όνομα key την τιμή value στο local storage. Αντίστοιχα για να αντλήσουμε μία τιμή μπορούμε να χρησιμοποιήσουμε την getLocItem(key), ώστε να μας επιστρέψει την τιμή που βρίσκεται στο όνομα key. Για την αποθήκευση πινάκων μπορούμε να χρησιμοποιήσουμε την addStorArray(key, value, length), όπου key είναι το όνομα που θέλουμε να δώσουμε στην αποθηκευμένη τιμή, value ο πίνακας και length το μέγεθος του πίνακα. Αντίστοιχα με την κλήση της getStorArray(key) θα μας επιστρέψει τον πίνακα. Οι παραπάνω συναρτήσεις βρίσκονται στο αρχείο storage.js.

## Εισαγωγή αλγορίθμου ΨΕΕ στην εφαρμογή

Όλα τα παρακάτω βήματα προϋποθέτουν την υλοποίηση του αλγορίθμου ΨΕΕ σε JavaScript και την αποθήκευση του στο φάκελο js. Θα παρουσιάσουμε τον τρόπο ενσωμάτωσης του στην εφαρμογή. Τα προηγούμενα σενάρια που εξετάσαμε λειτουργούν αυτομάτως κάθε φορά που φορτώνεται μία εικόνα στην εφαρμογή. Αφού λοιπόν φορτωθεί η εικόνα και συγκεντρωθούν τα απαραίτητα δεδομένα που χρειαζόμαστε, σειρά έχει η επεξεργασία της με αλγορίθμους ΨΕΕ. Παρακάτω θα αναλύσουμε τον τρόπο με τον οποίο μπορούμε να εισάγουμε έναν τέτοιο αλγόριθμο στην εφαρμογή.

Αρχικά θα πρέπει να σχεδιάσουμε τις απαραίτητες επιλογές (κουμπιά, κείμενα, κτλ) στο περιβάλλον του χρήστη, με τις οποίες θα είναι δυνατό να επιλέγει τον αλγόριθμο που θα εισάγουμε. Για να γίνει πιο κατανοητός ο τρόπος εισαγωγής ενός αλγορίθμου θα χρησιμοποιήσουμε το όνομα «algorithm( )», υποθέτοντας πως είναι ο επιθυμητός αλγόριθμος ΨΕΕ που θέλουμε να εισάγουμε και «algorithm» όπου χρειάζεται να δοθεί ως όρισμα σε συνάρτηση ή να γίνει αναφορά σε αυτόν.

Ας θεωρήσουμε ότι έχουμε τον algorithm και επιθυμούμε να τον εισάγουμε στην κατηγορία Process. Κάθε νέα κατηγορία στο μενού είναι εφικτό να εισαχθεί, τοποθετώντας την κάτω από το εξής κομμάτι κώδικα index.HTML:

*<!—Δημιουργία της κάθε κατηγορίας και υποκατηγορίας στο μενού-->*<**div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1"**>  
<**ul class="nav navbar-nav"**>

Στα παρακάτω </a> tags μπορούμε να βρούμε την κατηγορία Process (σημειωμένο με κόκκινο). Όμως αν θέλουμε να εισάγουμε μια νέα κατηγορία, τότε θα πρέπει να κάνουμε αντιγραφή-επικόλληση το παρακάτω κομμάτι κώδικα και να του δώσουμε το όνομα που επιθυμούμε.

*<!--Κατηγορία -->*<**li class="dropdown"**>  
  
<**a href="#" class="dropdown-toggle" data-toggle="dropdown" data-submenu="" role="button"  
aria-haspopup="true" aria-expanded="false"**> Process <**span class="caret"**></**span**>  
</**a**>  
<**ul class="dropdown-menu"**>

Για εισαγωγή υποκατηγορίας, στο παράδειγμα μας είναι ο algorithm, θα πρέπει να εισάγουμε τα εξής στοιχεία HTML:

*<!—Διαχωριστική γραμμή ανάμεσα στις υποκατηγορίες-->*<**li role="separator" class="divider"**></**li**>  
  
*<!—υποκατηγορία της κατηγορίας-->*<**li class="dropdown-submenu"**>  
<**a tabindex="0"**> algorithm </**a**>  
  
<**ul class="dropdown-menu"**>  
<**li**><**a onclick="**algorithm()**" tabindex="0"**><**span class="glyphicon glyphicon-play"**></**span**>Use</**a**></**li**>  
<**li**><**a tabindex="0" onclick="***goToPractice*(**'**algorithm**'**)**"**><**span class="glyphicon glyphicon-pencil"**></**span**> Practice</**a**></**li**>  
<**li**><**a tabindex="0" onclick="***goToTheory*(**'**algorithm**'**)**"**><**span class="glyphicon glyphicon-book"**></**span**> Theory</**a**></**li**>  
</**ul**>  
</**li**>

Ας δούμε αναλυτικά τι περιγράφει το λεκτικό algorithm σε κάθε μια γραμμή κώδικα παραπάνω. Το συναντάμε σε τέσσερα </a> tags.

α) Το πρώτο εμπεριέχει το όνομα που θα εμφανίζεται στο κουμπί που βλέπει ο χρήστης. Όταν πατηθεί θα εμφανίσει τρείς επιλογές. Τα επόμενα τρία </a> tags περιγράφουν τις επιλογές που θα προβληθούν.

β) Η πρώτη επιλογή που θα εμφανιστεί ονομάζεται «use» και είναι το δεύτερο </a> tag στο παραπάνω κώδικα. Στο συγκεκριμένο υπάρχει μια ενέργεια onclick η οποία ενεργοποιείται όταν επιλεγεί από τον χρήστη. Η ενέργεια onclick θα ψάξει για την συνάρτηση algorithm(), ώστε να την εκτελέσει και να προβάλει τα αποτελέσματα στην οθόνη.

γ) Η επόμενη επιλογή έχει όνομα «Practice». Υπάρχει και σε αυτήν μία ενέργεια onclick η οποία καλεί την συνάρτηση goToPractice ('algorithm')με όρισμα το algorithm. Αν πατηθεί θα ανοίξει μια δεύτερη σελίδα στην οποία ο χρήστης έχει την δυνατότητα να εξασκηθεί πάνω στον αλγόριθμο.

δ) Η τελευταία επιλογή λειτουργεί όπως η προηγούμενη. Ονομάζεται «Theory» και όταν πατηθεί καλεί την συνάρτηση goToTheory ('algorithm') με όρισμα algorithm. Με την επιλογή αυτή ο χρήστης μεταφέρεται σε μία νέα σελίδα όπου του δίνει την δυνατότητα να διαβάσει το θεωρητικό υπόβαθρο του αλγορίθμου.

Αυτές οι τρείς επιλογές αναλύονται παρακάτω με λεπτομέρεια. Με κόκκινο έχει σημειωθεί η λέξη algorithm. Επειδή η παρακάτω διαδικασία που θα αναλύσουμε, θα προσπαθήσει να προβάλει στην οθόνη την αντίστοιχη θεωρητική επεξήγηση του αλγορίθμου που θα ενσωματώσουμε, μεταβαίνουμε στο αρχείο theories.js. Εκεί υπάρχει το αντικείμενο theories. Όλες οι θεωρίες είναι εκφρασμένες ως συμβολοσειρά με στοιχεία μορφοποιημένα σε HTML, καταχωρημένα στο αντίστοιχο όνομα μεταβλητής. Με την τεχνική αυτή καταφέρνουμε να χρησιμοποιούμε τις ιδιότητες της JavaScript και συγχρόνως να προβάλλεται ως κείμενο HTML στην οθόνη. Ετοιμάζουμε το κείμενο μας με τον αντίστοιχο τρόπο που είναι και τα υπόλοιπα υλοποιημένα και είμαστε έτοιμοι να προχωρήσουμε στην ενσωμάτωση του αλγορίθμου.

Παρακάτω παρουσιάζεται μια πρότυπη συνάρτηση που μπορούμε να χρησιμοποιήσουμε για να εισάγουμε τον αλγόριθμο μας στην εφαρμογή. Κάθε αλγόριθμος ΨΕΕ στην εφαρμογή έχει μία τέτοια συνάρτηση με το όνομα του. Βρίσκονται στο αρχείο /js/index/clickFunctions.js και εκτελείται όταν πατηθεί το κουμπί «use».

**function** algorithm()  
{  
**if** (getStorItem(**'imgData'**) == **null**) *// ελέγχει αν ο πίνακας είναι γεμάτος. Δηλαδή αν έχει φορτωθεί κάποια εικόνα στην εφαρμογή.*{  
 $(**"#error-msg"**).html(**"You must first upload an image!"**);  
 snackBar();) *// εμφανίζει μήνυμα στην οθόνη αν δεν έχει φορτωθεί εικόνα.*  
 }  
**else** {  
**var** im = document.getElementById(**'top-left-image'**); ) *// γραπώνει την εικόνα*   
**var** cloneImg=clone(gray.histogram,gray.imgGray);   
**var** thresh = algorithm(cloneImg.histogram);  
 print(thresh, cloneImg.cloneImgGray, im);  
 addStorItem(**"thresh"**,thresh);  
 addStorItem(**"HistTlt"**, **"threshold"**);  
 histogram(gray.histogram);  
 showImage(getStorItem(**"editedImg"**), **"top-right-image"**);  
 setText(**"algorithm"**,**'index'**);  
 addLocItem(**"practiceAlg"**, **" algorithm "**);  
 $(**'#save-btn'**).prop(**'disabled'**, **false**);  
 $(**'#change-img-btn'**).prop(**'disabled'**, **false**);  
 document.getElementsByClassName(**'zoom-btn'**)[1].style.visibility = **"visible"**;  
**if**(getStorItem(**"imgData"**)==**null** || getStorItem(**"editedImg"**)==**null**) *// Ελέγχει αν έχει δημιουργηθεί error 404 από μεγάλες εικόνες.*{  
 clear(**'all'**); *// Επαναφέρει την εφαρμογή στην αρχή της κατάσταση*  
 $(**"#error-msg"**).html(**"image is huge for this algorithm"**);  
 snackBar();  
 }  
 }   
}

Πρώτο μέλημα του σεναρίου είναι να πραγματοποιήσει έλεγχο αν έχει φορτωθεί κάποια εικόνα στην εφαρμογή. Για να το επιτύχει καλεί την getStorItem (**'imgData')** η οποία ελέγχει αν έχουν αποθηκευτεί δεδομένα στο local storage για την τιμή imgData. Η τιμή imgData περιέχει τα δεδομένα της εικόνας τα οποία έχουν συλλεχτεί κατά την εισαγωγή της και έχουν αποθηκευτεί στην τιμή αυτή. Έπειτα αν όντως έχει φορτωθεί κάποια εικόνα στην εφαρμογή την καταχωρεί στην μεταβλητή im. Καλείται η συνάρτηση clone(gray.histogram, gray.imgGray) με πρώτο όρισμα τον αριθμητικό πίνακα ακεραίων τιμών ιστογράμματος της εικόνας και ως δεύτερο το αντικείμενο εικόνας. Η συνάρτηση clone( ) θα κατασκευάσει ακριβές αντίγραφα αυτών των ορισμάτων. Δεν χρησιμοποιούμε τα πρωτότυπα διότι οποιαδήποτε επεξεργασία πραγματοποιήσουμε σ αυτά θα είναι οριστική, με αποτέλεσμα να αλλοιωθούν τα δεδομένα μας και στην επόμενη επεξεργασία της ίδιας εικόνα να λάβουμε ψευδή αποτελέσματα. Εν συνεχεία καλείται ο αλγόριθμος επεξεργασίας εικόνας, algorithm στο παράδειγμα μας, με τα απαραίτητα κλωνοποιημένα ορίσματα. Η επιστρεφόμενη τιμή καταχωρείται στην μεταβλητή thresh. Η συνάρτηση print, που καλείται αμέσως μετά, λαμβάνει την τιμή κατωφλίωσης, το κλωνοποιημένο αντικείμενο εικόνας και τα δεδομένα της εισαχθείσας εικόνας για να σχεδιάσει το αποτέλεσμα που προκύπτει μετά την χρήση του algorithm. Θα αποθηκεύσει το αποτέλεσμα στο local storage με όνομα editedImg. Οι τιμές thresh και HistTlt θα αποθηκευτούν κι αυτές στο local storage για να χρησιμοποιηθούν από την επόμενη συνάρτηση, την histogram. Η histogram θα σχεδιάσει το ιστόγραμμα της εικόνας και με την βοήθεια των αποθηκευμένων τιμών θα σημειώσει την τιμή κατωφλίωσης με κόκκινο πάνω στο διάγραμμα και θα εμφανίσει την τιμή ως κείμενο. Αμέσως μετά γίνεται άντληση των αποθηκευμένων τιμών και προβάλλεται η εικόνα μετά την εφαρμογή του algorithm πάνω σ αυτήν. Εμφανίζεται η θεωρία του αλγορίθμου. Αποθηκεύεται στο practiceAlg ο αλγόριθμος που χρησιμοποιήσαμε, έτσι είναι εφικτή η επικοινωνία με τις άλλες σελίδες, πρακτικής εξάσκησης και θεωρίας. Οι επόμενες γραμμές εμφανίζουν κάποια κουμπιά για μεγέθυνση και σμίκρυνση της εικόνας. Τέλος ελέγχει αν έχει δημιουργηθεί σφάλμα κατά την παραπάνω διαδικασία, λόγω μεγάλου όγκου δεδομένων κι εμφανίζει μήνυμα σφάλματος αν έχει όντως συμβεί.

Συνεχίζοντας με την δεύτερη επιλογή, αυτή της πρακτικής εξάσκησης, οι ενέργειες που θα πρέπει να πραγματοποιηθούν, στο αρχείο index.HTML, για να ανοίξει η σελίδα για τον εκάστοτε αλγόριθμο θα πρέπει να είναι οι εξής:

<**li**><**a tabindex="0" onclick="***goToPractice*(**'algorithm'**)**"**><**span class="glyphicon glyphicon-pencil"**></**span**> Practice</**a**></**li**>

Στον παραπάνω κώδικα, στην συνάρτηση goToPractice, πρέπει να δώσουμε ως όρισμα το όνομα του αλγορίθμου. Η συνάρτηση καλείται όταν πατηθεί το κουμπί με όνομα Practice. Θα ελέγξει ποιο όνομα αλγορίθμου ΨΕΕ έχει αποθηκευτεί στην τιμή practiceAlg και αν υπάρχει θα ανοίξει την σελίδα πρακτικής άσκησης για τον αλγόριθμο που επιλέχθηκε. Παρακάτω παρουσιάζεται η συνάρτηση. Βρίσκεται στο αρχείο /js/index/clickFunctions.js.

**function** goToPractice(alg)  
{  
**if** (alg === undefined)  
 {  
**if** (getLocItem(**"practiceAlg"**) == **null**)  
 {  
 $(**"#error-msg"**).html(**"You must first choose an algorithm"**);  
 snackBar();  
 } **else**{  
 openWnd(**'practice.html'**); *\*}  
 }**else**{  
 addLocItem(**'practiceAlg'**, alg);  
 openWnd(**'practice.html'**);  
 }  
}

Είναι κατανοητό ότι ο έλεγχος γίνεται από τις αποθηκευμένες τιμές που υπάρχουν στο local storage, συγκεκριμένα για την τιμή practiceAlg. Τα σενάρια που θα εκτελεστούν θα μας οδηγήσουν στην καρτέλα της πρακτικής άσκησης (Practice.html). Εκεί θα πρέπει να προσθέσουμε στο header την διαδρομή αρχείου που βρίσκεται αποθηκευμένος ο αλγόριθμος μας ΨΕΕ, όπως παρακάτω.

<**script src="js/practice/generateNums.js"**></**script**>  
<**script src="js/otsu.js"**></**script**>  
<**script src="js/kapur.js"**></**script**>  
<**script src="js/eqHistogram.js"**></**script**>  
<**script src="js/algorithm.js"**></**script**>  
  
Έπειτα στην παρακάτω λίστα είναι απαραίτητο να ενσωματώσουμε το όνομα του αλγορίθμου, ώστε να το επιλέγει ο χρήστης από το μενού αλλά και η εφαρμογή να μπορεί να καλεί τα απαραίτητα σενάρια για την προβολή των επιθυμητών αντικειμένων.  
  
<**li class="info-link"**><**a onclick="***setPractice*(**'histogram'**)**"**>Histogram</**a**></**li**>  
<**li class="info-link"**><**a onclick="***setPractice*(**'eqHistogram'**)**"**>Histogram equalization</**a**></**li**>  
<**li class="info-link"**><**a onclick="***setPractice*(**'otsu'**)**"**>Otsu</**a**></**li**>  
<**li class="info-link"**><**a onclick="***setPractice*(**'kapur'**)**"**>Kapur</**a**></**li**>

<**li class="info-link"**><**a onclick="***setPractice*(**'algorithm'**)**"**>algorithm</**a**></**li**>

Σειρά έχουν τα σενάρια τα οποία εκτελούνται εδώ. Βασικότερο είναι το σενάριο generateNums. Βρίσκεται στο αρχείο js/practice/generateNums.js. Εφόσον παράξει πέντε τυχαίους διαφορετικούς ακέραιους αριθμούς στο διάστημα 1-256, τους αποθηκεύει σε μορφή πίνακα, στην μεταβλητή array, στο local storage και δημιουργεί την εικόνα των 25 εικονοστοιχείων που εμφανίζεται στην πάνω δεξιά πλευρά της οθόνης, στην εφαρμογή. Τα δεδομένα αποθηκεύονται σ αυτήν την μορφή για να μπορούν να επεξεργασθούν από τους αλγορίθμους ΨΕΕ.

**function** generateNums()  
{  
**var** arrayNums = **new** Array(5);  
**var** mArray =**new** Array(256); *// Χρησιμοποιείται για την αποθήκευση των 5 τιμών. Δημιουργήθηκε για να χρησιμοποιείται στον έλεγχο των απαντήσεων.  
 // Λόγω του ότι οι αλγόριθμοι kapur, otsu, κτλ επεξεργάζονται πίνακες μεγέθους 256 στοιχείων.***var** grayImgValues=**new** Array(25); *// Περιέχει τις τιμές των αποχρώσεων με την σειρά που παραχθήκαν.  
 // Χρησιμοποιείται ως όρισμα στην κλήση της συνάρτησης createPracticeImg για την εμφάνιση της εικόνας.***var** i,num;  
**for**(i=0;i<256;i++){mArray[i]=0;}  
  
**for** ( i=0;i<5;i++)  
 {  
 num=Math.floor((Math.random() \* 256) ); *// Τυχαίες τιμές μεταξύ 0 και 255.***while**(arrayNums.includes(num)) *// Αν ο τυχαίος αριθμός υπάρχει ήδη στον πίνακα τότε ξανά διάλεξε τυχαίο αριθμό.*{  
 num = Math.floor((Math.random() \* 256) );  
 }  
 arrayNums[i] = num;  
 }  
 addStorArray(**"fiveArrColoVal"**,arrayNums,arrayNums.length); *// Αποθηκεύει τιμές για την συνάρτηση setDisplayProperties.***var** count;  
 count=[0,0,0,0,0]; *// Αρχικοποίηση***for**(i=0;i<25;i++) *// Κάνε 25 επαναλήψεις και διάλεγε μια από τις 5 τυχαίες τιμές κάθε φορά.*{  
 num = Math.floor((Math.random() \* 5) );  
 count[num]++;  
 mArray[arrayNums[num]]++;  
 grayImgValues[i]=arrayNums[num];  
 }  
**for** (i=0;i<5;i++) *// Έλεγχος αν από τις πέντε τυχαίες τιμές εμφανίζονται και οι πέντε. Αν όχι ξανατρέχει η συνάρτηση.*{  
**if**(count[i]===0){generateNums();}  
 }  
 addStorArray(**"array"**,mArray,mArray.length);  
 createPracticeImg(grayImgValues); *// Η συνάρτηση βρίσκεται στο παρόν αρχείο.*}

Σύμφωνα με τον αλγόριθμο ΨΕΕ που έχει επιλέξει ο χρήστης για να εξασκηθεί εμφανίζεται και ο κατάλληλος χώρος που θα τοποθετηθεί η απάντηση της άσκησης. Δηλαδή μπορεί να είναι ένας ακέραιος αριθμός οπότε χρειάζεται μόνο ένα πεδίο ή ένας πίνακας ακεραίων άρα περισσότερα πεδία για την εισαγωγή των τιμών. Η συνάρτηση που ρυθμίζει αυτές τις παραμέτρους είναι η setDisplayProperties και βρίσκεται στο αρχείο js/practice/clickFunctionsPr.js.

Όταν ο χρήστης πληκτρολογήσει την απάντηση και πατήσει το κουμπί answer and check τότε ανάλογα με το αν έχει εισαχθεί πίνακας ή όχι θα εκτελεστούν τα εξής σενάρια:  
α) για έναν ακέραιο αριθμό η συνάρτηση checkAnswer().

β) για έναν πίνακα ακεραίων η συνάρτηση checkAnswerHists().

Και στις δύο περιπτώσεις αυτό που θα κάνουν τα σενάρια είναι να πάρουν την απάντηση του χρήστη. Έπειτα καλούν τον αλγόριθμο ΨΕΕ στον οποίο εξασκείται ο χρήστης με όρισμα τους ακεραίους που δημιούργησαν την εικόνα εξ αρχής. Τους αριθμούς αυτούς τους αντλεί από το local storage. Στη συνέχεια αποθηκεύει το σωστό αποτέλεσμα και επιστρέφει για να ελέγξει αν συμφωνεί με αυτό που πληκτρολόγησε ο χρήστης. Αν ναι τότε εμφανίζει μήνυμα επιτυχίας και φορτώνει νέα άσκηση στον ίδιο αλγόριθμο, διαφορετικά εμφανίζει μήνυμα λάθους. Τα σενάρια βρίσκονται στο αρχείο js/practice/clickFunctionsPr.js.

Αν ο χρήστης επιλέξει να δει την απάντηση τότε θα γίνει κλίση του σεναρίου printAnswer( ). Ελέγχει για ποίον αλγόριθμο ζητήθηκε η απάντηση. Εμφανίζει το κείμενο. Αντλεί τις τιμές των εικονοστοιχείων που κατασκεύασαν την εικόνα από το local storage, εκτελεί τον αλγόριθμο και ο αλγόριθμος αποθηκεύει τις τιμές που θα χρειαστεί το σενάριο για να προβάλει στην απάντηση. Εφόσον αποθηκευτούν το σενάριο τις αντλεί από το local storage και τις προσαρμόζει στις ανάγκες της εκάστοτε απάντησης. Το σενάριο βρίσκεται στο αρχείο js/practice/clickFunctionsPr.js.

Ας δούμε αναλυτικότερα όσα αναφέραμε παραπάνω. Στην συνάρτηση setDisplayProperties θα πρέπει να κάνουμε τη παρακάτω προσθήκη του ονόματος και να δώσουμε τον κατάλληλο αύξοντα αριθμό για να εμφανίζεται η επιλογή στο μενού.

**function** setDisplayProperties(){  
**var** prop=getLocItem(**'practiceAlg'**); *// Ελέγχει τον τρέχον αλγόριθμο.***var** i;  
**var** title=$(**'.info-link'**); *// Γραπώνει την κλάση info-link.***var** switchInputs=$(**'.switchInputs'**);  
**switch** (prop) {  
**case "histogram"**:  
 title.removeClass(**"active"**); *// Αφαιρεί από όλη την κλάση την ιδιότητα active.*title.eq(0).addClass(**"active"**); *// Προσθέτει μόνο στο πρώτο στοιχείο την ιδιότητα active. Το φόντο του ονόματος του αλγορίθμου στην επάνω μπάρα επιλογής θα σκουρύνει.*switchInputs.html(switchInps.arrayGrid); *// Αλλάζει τα inputs. Υπάρχει στο αρχείο switchInputs.js***var** txtInpsName=getStorArray(**'fiveArrColoVal'**);  
 txtInpsName.sort(**function**(a,b) { **return** a - b; });  
 console.log(txtInpsName);  
**for** (i=0;i<txtInpsName.length;i++) *// Τοποθετεί στα inputs τις τιμές των αποχρώσεων της εικόνας για να μπορεί ο χρήστης να καταλάβει που θα πληκτρολογήσει το αποτέλεσμα του.*{  
 $(**'input'**).eq(i).attr(**'placeholder'**,txtInpsName[i]);  
 }  
**break**;  
**case "otsu"**:  
 title.removeClass(**"active"**);  
 title.eq(2).addClass(**"active"**);  
 switchInputs.html(switchInps.oneGrid);  
**break**;  
**case "kapur"**:  
 title.removeClass(**"active"**);  
 title.eq(3).addClass(**"active"**);  
 switchInputs.html(switchInps.oneGrid);  
**break**;  
**case "eqHistogram"**:  
 title.removeClass(**"active"**);  
 title.eq(1).addClass(**"active"**);  
 switchInputs.html(switchInps.arrayGrid);  
**break**;

**case "algorithm"**:  
 title.removeClass(**"active"**);  
 title.eq(4).addClass(**"active"**);  
 switchInputs.html(switchInps.arrayGrid);  
**break**;  
  
**default**:  
  
 }  
 $(**'#displayAnswer'**).html(**' '**); *// Αδειάζει την περιοχή που έχει τυπωθεί η λύση της άσκησης.*}

Εν συνεχεία παρουσιάζονται οι περιπτώσεις ελέγχου. Όπως δηλώσαμε και παραπάνω ο αλγόριθμος θα ανήκει σε μία από τις δύο κατηγορίες. Αν η απάντηση αποτελείται από έναν ακέραιο τότε θα καλέσουμε την checkAnswer. Παρακάτω έχουμε εισάγει τον αλγόριθμο algorithm από το παράδειγμα μας.

**function** checkAnswer()  
{  
**var** message=$(**"#error-msg"**); *// Γραπώνει το div με id error-msg και το καταχωρεί στην μεταβλητή message. Είναι υπεύθυνο για να τυπώνει τα μηνύματα στην οθόνη.***try**{  
**var** intNum=parseInt(document.getElementById(**"input-answer"**).value,10); *// Παίρνει την τιμή που υπάρχει στο input με id='input-answer' την μετατρέπει σε ακέραιο και την εκχωρεί στην μεταβλητή intNum.***if**(Number.isInteger(intNum)) *// Ελέγχει αν η τιμή είναι πράγματι ακέραιος.*{  
**var** compareNum; *// Στην μεταβλητή αυτή θα εκχωρηθεί η σωστή απάντηση και θα συγκριθεί με την απάντηση του χρήστη.***switch** (getLocItem(**"practiceAlg"**)) *// Ελέγχει ποιος αλγόριθμος έχει επιλεγεί.*{  
*// Αυτό το καταλαβαίνει κοιτάζοντας στο storage στο οποίο υπάρχει κάθε φορά το όνομα του τρέχον αλγορίθμου.***case "kapur"**:  
 compareNum=kapur(getStorArray(**"array"**)); *// Παίρνει τον πίνακα με όνομα array από το storage. Ο πίνακας αυτός περιέχει τις τιμές που παράχθηκαν από το κουμπί 'generate new exercise data'. Δημιουργήθηκε και αποθηκεύτηκε από την συνάρτηση generateNums βρίσκεται στο αρχείο generateNums.js.  
 // Έπειτα καλεί την συνάρτηση otsu ή kapur για να λάβει τα σωστά αποτελέσματα.  
 // Η συνάρτηση getStorArray βρίσκεται στο αρχείο storagePr.js.***break**;  
**case "otsu"**:  
 compareNum=otsu(getStorArray(**"array"**));  
**break**;

**case "algorithm"**:  
 compareNum=algorithm(getStorArray(**"array"**));  
**break**;  
  
**default**:  
 compareNum= **""**;  
 }  
**if**(compareNum===intNum)  
 {  
 generateNums(); *// Αν η απάντηση είναι σωστή θα γίνει ανανέωση της άσκησης με νέα εικόνα. Η συνάρτηση generateNums βρίσκεται στο αρχείο generateNums.js.*message.css(**"background-color"**, **"green"**); *// Βάφει πράσινο το φόντο του μηνύματος.*message.html(**"Correct"**); *// Εκχωρεί το κείμενο που θα εμφανιστεί.*snackBar(); *// Εμφανίζει το μήνυμα. Η συνάρτηση snackBar βρίσκεται στο παρόν αρχείο.*}**else**{  
 message.css(**"background-color"**, **"red"**);  
 message.html(**"False"**);  
 snackBar();  
 }  
 }**else**{  
 message.css(**"background-color"**, **"#333"**);  
 message.html(**"This is not an integer"**);  
 snackBar();  
 }  
 }**catch**(err)  
 {  
 message.css(**"background-color"**, **"#333"**);  
 message.html(**"This is not a number"**);  
 snackBar();  
 }  
}

Αν βρισκόμαστε στην κατηγορία όπου η απάντηση αποτελείται από περισσότερα στοιχεία του ενός τότε θα κινηθούμε όπως παρακάτω. Θεωρούμε πως ο αλγόριθμος algorithm έχει λύση με περισσότερους του ενός ακεραίου.

**function** checkAnswerHists()  
{  
**var** message=$(**"#error-msg"**); *// Γραπώνει το div με id error-msg και το καταχωρεί στην μεταβλητή message.  
// Είναι υπεύθυνο για να τυπώνει τα μηνύματα στην οθόνη.***try**{  
**var** intArray=**new** Array(5); *// Πίνακας στον οποίο θα εκχωρηθούν οι τιμές που πληκτρολόγησε ο χρήστης.***var** i;  
**var** pass; *// Boolean μεταβλητή. Αν είναι σωστή η απάντηση του χρήστη παίρνει την τιμή true.***for** (i=0;i<intArray.length;i++)  
 {  
 intArray[i]=parseInt($(**'input'**).eq(i).val(),10); *// Οι απαντήσεις του χρήστη μετατρέπονται σε ακέραιοι και εκχωρούνται στον πίνακα intArray.*}  
**switch** (getLocItem(**"practiceAlg"**))  
 {  
**case "histogram"**:  
 pass=checkHist(intArray); *// Η ckeckHist βρίσκεται στο παρόν αρχείο.***break**;  
**case "eqHistogram"**:  
 pass=checkEqHist(intArray); *// Η ckeckEqHist βρίσκεται στο παρόν αρχείο.***break**;

**case "algorithm"**:  
 pass=checkAlgorithmHist(intArray); *// Η ckeckEqHist βρίσκεται στο παρόν αρχείο.***break**;  
  
**default**:  
 pass= **false**;  
 }  
**if**(pass)  
 {  
 generateNums(); *// Έχει γίνει αναφορά παραπάνω.*message.css(**"background-color"**, **"green"**);  
 message.html(**"Correct"**);  
 snackBar();  
 }**else**{  
 message.css(**"background-color"**, **"red"**);  
 message.html(**"False"**);  
 snackBar();  
 }  
 }**catch**(err)  
 {  
 message.css(**"background-color"**, **"#333"**);  
 message.html(**"some input value is not a number"**);  
 snackBar();  
 }  
}

Η pass=checkAlgorithmHist(intArray); που συναντήσαμε στον κώδικα θα πρέπει να σχεδιαστεί όπως οι αντίστοιχες **checkEqHist(intArray)** και **checkHist(intArray)** για τον έλεγχο τιμών του αλγορίθμου μας ΨΕΕ.

Η printAnswer ελέγχει για ποιόν αλγόριθμο ζητήθηκε η απάντηση και την σχεδιάζει. Αφού παράξει και τοποθετήσει τα νούμερα στις σωστές θέσεις, έπειτα τα εμφανίζει στην οθόνη. Οι γραμμές κώδικα που θα προστεθούν στην περίπτωση του κάθε αλγορίθμου δεν ακολουθούν κάποια δεδομένη σειρά, διότι κάθε ένας έχει διαφορετικά κατασκευασμένη την απάντηση. Υπολογίζει διαφορετικές παραμέτρους, εμφανίζει διαφορετικές μαθηματικές μεταβλητές όπως και διαφορετικό κείμενο. Οπότε είναι κατανοητό πώς θα πρέπει να δομηθεί με παρόμοιο τρόπο που δομήθηκαν και τα υπόλοιπα.

**function** printAnswer()  
{  
**var** expName=getLocItem(**'practiceAlg'**); *// Έλεγχος για τον τρέχον αλγόριθμο.***var** text=$(**"#displayAnswer"**); *// Γραπώνει το σημείο όπου θα εμφανιστεί το κείμενο με την απάντηση.***var** setVal; *// Δέχεται την τοποθεσία που θα εκτυπώσει λεπτομερή αποτελέσματα όπως μέσους όρους, αθροιστικές πιθανότητες, κτλ.***var** originalArray; *// Δέχεται τον πίνακα array από το storage. Χρησιμοποιεί τα δεδομένα του για να εκτυπώσει την λύση της άσκησης στην οθόνη.***var** i,j,k; *// Μετρητές.***var** Argmnts; *// Δέχεται τις επιστρεφόμενες τιμές από τις συναρτήσεις kapur, otsu, κτλ.***switch** (expName)  
 {  
**case "histogram"**:  
 text.html(explainTxt.histogram); *// Το αντικείμενο explainTxt βρίσκεται στο αρχείο explainTxt.js και περιέχει κείμενο για την επεξήγηση των ασκήσεων.*setVal=$(**" #displayAnswer > table > tbody >tr>td "**); *// Το αντικείμενο explainTxt εισάγει έναν πίνακα στον html δέντρο. Η εντολή αυτή γραπώνει τον πίνακα που θα τυπωθούν οι απαντήσεις.*originalArray=getStorArray(**'array'**); *// Ο array δημιουργείται από την συνάρτηση generateNums του αρχείου generateNums.js.*j=2;  
**for**(i=0;i<256;i++)  
 {  
**if**(originalArray[i]!=0)  
 {  
 setVal.eq(j).html(i);j++;  
 setVal.eq(j).html(originalArray[i]);j++;  
 }  
 }  
**break**;  
**case "otsu"**:  
 text.html(explainTxt.otsu);  
 setVal=$(**" #displayAnswer > table > tbody >tr>td "**);  
 Argmnts=otsu(getStorArray(**'array'**),**'practice'**); *// Καλείται η συνάρτηση otsu με όρισμα τον πίνακα που περιέχει τις τιμές της εικόνας και δεύτερο όρισμα practice για να μας επιστρέψει τις τιμές που χρειαζόμαστε.*originalArray=getStorArray(**'array'**);  
**var** span=$(**'#displayAnswer > span'**); *// Το span έχει δημιουργηθεί από το αντικείμενο explainTxt για να τοποθετούνται από κάτω του οι απαντήσεις.***var** p1=[],t=[],m1=[],sb=[],T=Argmnts[3],m=Argmnts[4];  
  
 j=3; *// Το j δείχνει τις θέσεις του html πίνακα. Οι 3 πρώτες θέσεις του πίνακα (όπως μπορεί κανείς να δει αν ανοίξει το αρχείο explainTxt.otsu) περιέχουν τους τίτλους των γραμμών (g, h(g), κτλ) οπότε θα πρέπει α τυπώσουμε τιμές μετά την θέση 2 (μετράμε από το 0).*k=-1;*// Την πρώτη μη αρνητική τιμή του sb δεν την χρειάζομαι. Για να την αποφύγω θέτω τον μετρητή κ=-1.***for**(i=0;i<256;i++)  
 {  
**if**(originalArray[i]!=0)  
 {  
 setVal.eq(j).html(i);j++;  
 setVal.eq(j).html(originalArray[i]);j++;  
 setVal.eq(j).html((originalArray[i]/25));j++;  
**if**(k>=0 )  
 {  
 p1[k]=Argmnts[0][k]; *// Τοποθετεί τις τιμές που έλαβε από την συνάρτηση otsu σε άλλους πίνακες για καλύτερη κατανόηση του κώδικα.*m1[k]=Argmnts[1][k];  
 sb[k]=Argmnts[2][i];  
 t[k]=Argmnts[5][k];  
 }  
 k++;  
 }  
 }  
 span.after(**'Την μεγαλύτερη τιμή σ<sub>b</sub><sup>2</sup> την συναντούμε για Τ='**+T+**'.<br><br>'**); *// Η after προσθέτει κείμενο μετά από το σημείο που της δείχνεις.  
 // Οπότε θα πρέπει να σκεφτούμε αντίστροφα. Πρώτα θα τυπώσουμε τα στοιχεία.  
 // Που θέλουμε να εμφανίζονται τελευταία.***for**(i=2;i>=0;i--)  
 {  
 span.after(**' .Άρα η συνολική διασπορά θα λάβει την τιμή s<sub>b</sub><sup>2</sup>= '**+sb[i].toFixed(2)+ **'.<br><br>'**);  
 span.after(**' και η αντίστοιχη αθροιστική πιθανότητα p<sub>1</sub> με p1= '**+p1[i].toFixed(2));  
 span.after(**' η μέση τιμή m<sub>1</sub> ισούται με m1= '**+m1[i].toFixed(2));  
 span.after(**'Για την τιμή t= '**+t[i]);  
 }  
 span.after(**'Η συνολική μέση τιμή <i>μ</i> ισούται με '**+m.toFixed(2)+**'<br><br>'**);  
**break**;  
**case "kapur"**:  
*// Ίδια λογική που ακολουθείται και στην περίπτωση otsu.*text.html(explainTxt.kapur);  
 setVal=$(**" #displayAnswer > table > tbody >tr>td "**);  
 originalArray=getStorArray(**'array'**);  
 Argmnts=kapur(getStorArray(**'array'**),**'practice'**);  
  
**var** span=$(**'#displayAnswer > span'**);  
**var** E1=[],E2=[],P=[],He=[],t=[],T=Argmnts[5];  
  
 j=9;k=-1;  
**for**(i=0;i<256;i++)  
 {  
**if**(originalArray[i]!=0)  
 {  
*// if(j==3){m1.unshift(i);}*setVal.eq(j).html(i);j++;  
 setVal.eq(j).html(originalArray[i]);j++;  
 setVal.eq(j).html((originalArray[i]/25));j++;  
 setVal.eq(j).html((Math.log(originalArray[i]/25)).toFixed(4));j++;  
**if**(k>=0)  
 {  
 setVal.eq(j).html(Argmnts[2][k].toFixed(2));j++;  
 setVal.eq(j).html(Math.log(Argmnts[2][k]).toFixed(4));j++;  
 setVal.eq(j).html(Argmnts[0][k].toFixed(2));j++;  
 setVal.eq(j).html(Argmnts[1][k].toFixed(2));j++;  
 setVal.eq(j).html(Argmnts[3][i].toFixed(2));j++;  
  
 E1[k]=Argmnts[0][k];  
 E2[k]=Argmnts[1][k];  
 P[k]=Argmnts[2][k];  
 He[k]=Argmnts[3][i];  
 t[k]=Argmnts[4][k];  
  
 }  
**else**{  
 j+=5;  
 }  
 k++;  
 }  
 }  
  
 span.after(**'Την μεγαλύτερη τιμή E την συναντούμε για Τ='**+T+**'.<br><br>'**); *// Η after προσθέτει κείμενο μετά από το σημείο που της δείχνεις.  
 // Οπότε θα πρέπει να σκεφτούμε αντίστροφα. Πρώτα θα τυπώσουμε τα στοιχεία.  
 // Που θέλουμε να εμφανίζονται τελευταία.***for**(i=2;i>=0;i--)  
 {  
 span.after(**' .Άρα η συνολική μέση πληροφορία θα λάβει την τιμή E= '**+He[i].toFixed(2)+ **'.<br><br>'**);  
 span.after(**' και η δεύτερη κλάση έχει μέση πληροφορία E<sub>2</sub> ίση E2= '**+E2[i].toFixed(2));  
 span.after(**' Η πρώτη κλάση έχει μέση πληροφορία (εντροπία) E<sub>1</sub> ίση με E1= '**+E1[i].toFixed(2));  
 span.after(**'Για την τιμή t= '**+t[i]);  
 }  
**break**;  
**case "eqHistogram"**:  
 text.html(explainTxt.eqHistogram);  
 setVal=$(**" #displayAnswer > table > tbody >tr>td "**);  
**var** prob=[];  
 originalArray=getStorArray(**'array'**);  
 j=6;  
 k=0; *// Η μεταβλητή 'κ' χρησιμοποιείται για την εκχώρηση τιμών στον πίνακα prob.***for**(i=0;i<256;i++)  
 {  
**if**(originalArray[i]!=0)  
 {  
 setVal.eq(j).html(i);++j;  
 setVal.eq(j).html(originalArray[i]);++j;  
 setVal.eq(j).html(originalArray[i]/25);j+=4;  
 prob[k]=originalArray[i]/25;k++;  
 }  
 }  
 j=15;  
 setVal.eq(9).html(prob[0].toFixed(2));  
**for**(i=1;i<5;i++) *// Εμφάνιση αθροιστικής πιθανότητας στην οθόνη.*{ *// debugger;*prob[i]=prob[i]+prob[i-1]; *// Υπολογισμός αθροιστικής πιθανότητας.*setVal.eq(j).html(prob[i].toFixed(2));j+=6;  
 }  
 min = Math.min.apply(Math, prob); *// Εύρεση μικρότερης τιμής στον πίνακα prob.*j=10;  
**for**(i=0;i<5;i++) *// Εμφάνιση τιμών στις δύο τελευταίες στήλες.*{ *//debugger;*setVal.eq(j).html(((prob[i]-min)/(1-min)).toFixed(2));  
 ++j;  
 setVal.eq(j).html(Math.round((prob[i]-min)/(1-min)\*255));  
 j+=5;  
 }  
**break**;  
**case "algorithm"**:

**break**;  
**default**:  
  
 }   
}

}

Για την επιλογή «Theory», η οποία ανοίγει μία νέα καρτέλα και προβάλλει την θεωρία του αλγορίθμου ακολουθούμε την παρακάτω διαδικασία:

Εφόσον έχουμε ετοιμάσει το κείμενο στο αρχείο theories.js μπορούμε να προχωρήσουμε στις παρακάτω διαδικασίες ώστε να προβάλλεται το κείμενο. Μεταφερόμαστε στο αρχείο Index.HTML. Όπως αναφερθήκαμε και πιο πάνω η συνάρτηση goToTheory δέχεται ένα όρισμα. Προσαρμόζουμε τη τιμή του ορίσματος ανάμεσα στα </a> tags και γράφουμε το όνομα του αλγορίθμου μας.

<**li**><**a tabindex="0" onclick="***goToTheory*(**'algorithm'**)**"**><**span class="glyphicon glyphicon-book"**></**span**> Theory</**a**></**li**>

Παρακάτω βλέπουμε την συνάρτηση που θα εκτελεστεί όταν ο χρήστης διαλέξει την επιλογή της θεωρίας. Όπως φαίνεται δεν χρειάζεται καμία επιπλέον προσθήκη στα σενάρια τις αρχική σελίδας (index.html). Το παρακάτω αρχείο βρίσκεται στο js/index/clickFunctions.js.

**function** goToTheory(alg)  
{  
**if** (alg === undefined)  
 {  
**if** (getLocItem(**"theoryAlg"**) == **null**)  
 {  
 $(**"#error-msg"**).html(**"You must first choose an algorithm"**);  
 snackBar();  
 }**else**{  
 openWnd(**'theory.html'**);  
 }  
 }**else**{  
 addLocItem(**'theoryAlg'**, alg);  
 openWnd(**'theory.html'**);  
 }  
}

Μεταφερόμαστε στη σελίδα της θεωρίας όπου κι εκεί θα πρέπει να κάνουμε κάποιες ελάχιστες προσθήκες. Στον παρακάτω κώδικα, στο αρχείο js/theory/clickFunctionTr.js θα πρέπει να προστεθεί το όνομα του αλγορίθμου στην εντολή απόφασης switch-case.

**function** setDisplayProperties(){

**var** prop=getLocItem(**'theoryAlg'**); *// Ελέγχει τον τρέχον αλγόριθμο.*  
**var** title=$(**'.info-link'**); *// Γραπώνει την κλάση info-link.*  
**switch** (prop) {  
**case "histogram"**:

title.removeClass(**"active"**);  
 title.eq(0).addClass(**"active"**);  
**break**;  
**case "otsu"**:  
 title.removeClass(**"active"**);  
 title.eq(2).addClass(**"active"**);  
**break**;  
**case "kapur"**:  
 title.removeClass(**"active"**);  
 title.eq(3).addClass(**"active"**);

**break**;

**case "kohonen"**:  
 title.removeClass(**"active"**);  
 title.eq(4).addClass(**"active"**);  
**break**;  
**case "eqHistogram"**:  
 title.removeClass(**"active"**);  
 title.eq(1).addClass(**"active"**);  
**break**;

**case "algorithm"**:  
 title.removeClass(**"active"**);  
 title.eq(5).addClass(**"active"**);  
**break**;  
  
**default**:  
 **text** = **"**undefined**"**;

}  
}

Προσθέσαμε την επιλογή «algorithm» με αύξοντα αριθμό το 5. Αυτή η εισαγωγή πραγματοποιείται για να φαίνεται στο μενού της σελίδας ποιος αλγόριθμος είναι επιλεγμένος. Τέλος στον κώδικα Theory.html θα πρέπει να προστεθεί η εξής γραμμή:

<**li class="info-link"**><**a onclick="***setTheory*(**'algorithm'**)**"**>algorithm</**a**></**li**>

Την γραμμή την προσθέτουμε στο παρακάτω κομμάτι κώδικα:

<**li class="info-link"**><**a onclick="***setTheory*(**'histogram'**)**"**>Histogram</**a**></**li**>  
<**li class="info-link"**><**a**

**onclick="***setTheory*(**'eqHistogram'**)**"**>Histogram equalization</**a**></**li**>  
<**li class="info-link"**><**a**

**onclick="***setTheory*(**'otsu'**)**"**>Otsu</**a**></**li**>  
<**li class="info-link"**><**a**

**onclick="***setTheory*(**'kapur'**)**"**>Kapur</**a**></**li**>

<**li class="info-link"**><**a onclick="***setTheory*(**'kohonen'**)**"**>Kohonen</**a**></**li**>

<**li class="info-link"**><**a onclick="***setTheory*(**'algorithm'**)**"**>algorithm</**a**></**li**>

## Μελλοντικές επεκτάσεις

Ο σκοπός δημιουργίας του διαδικτυακού χώρου είναι να προσφέρει μια ολοκληρωμένη κάλυψη στον εκπαιδευτικό τομέα της ψηφιακής επεξεργασίας εικόνας, για να αντιμετωπιστούν δυσκολίες που συναντούν οι φοιτητών στη διδασκαλία της. Σε ένα συνεχώς εξελισσόμενο επιστημονικό τομέα αναμοχλεύεται διαρκώς η ανάγκη για διδασκαλία των νέων τεχνικών και θεωρητικών ανακαλύψεων της.

Αναλογιζόμενοι όλα τα παραπάνω, κατά την σχεδίαση και ανάπτυξη της ιστοσελίδας, λάβαμε υπόψη μας και αφήσαμε ανοιχτές τις πόρτες για οποιοσδήποτε μελλοντικές ενημερώσεις. Προσπαθήσαμε να απομονώσουμε τις διαδικασίες που συνθέτουν τα γραφικά, με εκείνες που επεξεργάζονται τα δεδομένα και των έλεγχο αυτών. Έπειτα τις εντάξαμε σε συναρτήσεις αυτοτελή και ευανάγνωστες. Έτσι διατηρήσαμε εφικτή κάθε μελλοντική ενημέρωση της εφαρμογής.

Η προσθήκη επιπλέον αλγορίθμων επεξεργασίας εικόνας εντάσσεται στις σημαντικότερες από τις μελλοντικές επεκτάσεις του διαδικτυακού τόπου. Θα προσφέρει στους επισκέπτες μεγαλύτερο εύρος από εφαρμοσμένες τεχνικές κατωφλίωσης ψηφιακής επεξεργασίας εικόνας με αποτέλεσμα να γίνετε πιο κατανοητός ο ρόλος της επιστήμης αλλά και ο τρόπος λειτουργίας αυτών. Μια επιπλέον μελλοντική επέκταση που θα μπορούσαμε να υλοποιήσουμε είναι η εγκατάσταση εξυπηρετητή. Ο εξυπηρετητής θα συμβάλει στην επεξεργασία μεγάλου όγκου δεδομένων. Τέτοια παραδείγματα είναι :

* Εικόνες με μέγεθος μεγαλύτερο των 2MB.
* Ενσωμάτωση και χρήση πολυπλοκότερων αλγορίθμων επεξεργασίας εικόνας .
* Διαδικασίες κατωφλίωσης έγχρωμης εικόνας και προβολή του εκάστοτε ιστογράμματος ή τρισδιάστατη προβολή ιστογράμματος .

Μια ακόμη χρήσιμη επέκταση θα ήταν η ανάπτυξη της ιστοσελίδας σε εφαρμογή για κινητές συσκευές. Το πλεονέκτημα είναι η χρήση της δίχως την προϋπόθεση σύνδεσης στο διαδίκτυο.

Όλες οι προαναφερθείσες επεκτάσεις μπορούν να αναπτυχθούν με την JavaScript. Βιβλιοθήκες όπως η ionic, Mobile Angular UI, ExtJS και πολλές άλλες παρέχουν την δυνατότητα ανάπτυξης εφαρμογών για κινητές συσκευές.

# 

## Εγχειρίδιο χρήσης

### Αρχική σελίδα

Η γενική σκέψη για τις ενέργειες που πραγματοποιούνται στην διαδικασία της εφαρμογής είναι η ακόλουθη .

1. Ο επισκέπτης επιλέγει μια εικόνα από το αρχείο του.
2. Η επιλογή ενός αλγορίθμου επεξεργασίας εικόνας ο οποίο θα εφαρμοστεί στην εικόνα που έχει επιλέξει .
3. Παρουσίαση των αποτελεσμάτων ( γραφήματα, επεξεργασμένη εικόνα, κτλ ) .
4. Επανάληψη διαδικασίας με νέα εικόνα .

Στο κεντρικό μενού συναντάμε 2 κατηγορίες, την **«File»** και **«Process»**. Η κατηγορία «File» εμπεριέχει επιλογές που αφορούν την διαχείριση του αρχείου εικόνας. Η επιλογή «Open» ανοίγει ένα παράθυρο επιλογής, από το οποίο μπορεί ο χρήστης να επιλέξει ένα αρχείο εικόνας, που υπάρχει αποθηκευμένη στον υπολογιστή του, να φορτωθεί για επεξεργασία. Η επιλογή «Save» ενεργοποιείται όταν πραγματοποιηθεί κάποια ΨΕΕ και δίνει την δυνατότητα αποθήκευσης της επεξεργασμένης εικόνας. Ομοίως και η επιλογή «Set resulted image as current» ενεργοποιείται μετά την εκτέλεση κάποιου αλγορίθμου ΨΕΕ επιτρέποντας την αντικατάσταση της αρχικής εικόνας που είχε φορτωθεί με την επεξεργασμένη.

Στην κατηγορία «Process», αν την αναπτύξουμε, συναντάμε τους αλγορίθμους ΨΕΕ που έχουν ενσωματωθεί στην εφαρμογή. Συγκεκριμένα είναι «Histogram», «Hisotgram equalization», «Otsu», «Kapur» και «Kohonen». Κάθε αλγόριθμος στην εφαρμογή αποτελείται από 3 επιλογές. Την «use», «practice» και «theory».

Η «use» για να ενεργοποιηθεί θα πρέπει πρώτα να έχει φορτωθεί κάποια εικόνα στην εφαρμογή. Όταν εκτελείται εφαρμόζει την εκάστοτε διαδικασία ΨΕΕ στην εικόνα που έχει φορτώσει ο χρήστης. Προβάλει το αποτέλεσμα της επεξεργασίας, το ιστόγραμμα της εικόνας με σημειωμένη την τιμή κατωφλιού και την θεωρητική επεξήγηση του αλγορίθμου ΨΕΕ.

Η επιλογή «practice» όταν πατηθεί μεταφέρει τον χρήστη σε μία νέα σελίδα για πρακτική εξάσκηση. Στην παράγραφο 4.2.2 περιγράφεται η λειτουργικότητα της.

Η επιλογή «theory» μεταφέρει τον επισκέπτη σε μία νέα σελίδα και προβάλει τη θεωρητική επεξήγηση του εκάστοτε αλγορίθμου ΨΕΕ.

### Σελίδα πρακτικής εξάσκησης

Στη σελίδα πρακτικής εξάσκησης μεταφερόμαστε μέσω της κεντρικής σελίδας από την επιλογή «Practice». Στα αριστερά υπάρχει μία εικόνα αποτελούμενοι από 25 εικονοστοιχεία. Τα εικονοστοιχεία πλαισιώνονται από 5 τυχαίες τιμές στο διάστημα 1-256 και εκφράζουν τις χρωματικές τιμές που μπορεί να λάβει ένα εικονοστοιχείο στην κλίμακα του γκρι. Σκοπός του επισκέπτη είναι να απαντήσει κάθε φορά στην εκάστοτε άσκηση ΨΕΕ, σύμφωνα με το όσα περιγράφει η θεωρία.

Το κουμπί «generate new exercise data» συνθέτει μία νέα εικόνα, με διαφορετικές τιμές. Το επόμενο κουμπί, «answer and check», λειτουργεί εφόσον ο χρήστης έχει πληκτρολογήσει την απάντηση του, δίπλα ακριβώς από το κουμπί, στον κενό χώρο. Θα ελέγξει αν η απάντηση που δόθηκε είναι σωστή και θα εμφανίσει μήνυμα σφάλματος αν είναι λάθος ή μήνυμα επιτυχίας αν έχει απαντήσει σωστά. Στις ασκήσεις ιστογράμματος και εξισορροπημένου ιστογράμματος η απάντηση αποτελείται από περισσότερα του ενός ακέραια νούμερα, οπότε υπάρχει κατάλληλα διαμορφωμένος χώρος για την τοποθέτηση των απαντήσεων.

Το τελευταίο κουμπί της σειράς, «solution in details», όταν πατηθεί θα εμφανίσει τον τρόπο με τον οποίο λύνεται η εκάστοτε άσκηση.

Από το κεντρικό μενού μπορούμε να αλλάξουμε αλγόριθμο εξάσκησης.

# 

## Βιβλιογραφία

Βιβλία

|  |  |
| --- | --- |
| [1] | BallardPhil, MoncurMichael.ΜάθετετηνJavaScriptσε 24 ώρες , Αθήνα, Ελλάδα, Δεκέμβρης 2013. |
| [2]  [3] | Duckett Jon. JavaScript & Jquery Interactive Front-end Web Development, New York, USA, July2014.  Παναγιώτης Δ. Κεντερλής. Ανάπτυξη διαδικτυακών εφαρμογών θεωρία και πράξη, Αθήνα, Ελλάδα, Ιούλιος 2009. |

Διαδικτυακοί ιστότοποι

|  |  |
| --- | --- |
| [4] | W3Schools.https://www.w3schools.com, Διαδικτυακός ιστότοπος διδασκαλίας ανάπτυξης διαδικτυακών εφαρμογών. |
| [5] | Bootstrap.https://getbootstrap.com, Επίσημος διαδικτυακός ιστότοπος με ελεύθερο υλικό και πληροφορίες για την bootstrap. |
| [6] | Plotly.<https://plot.ly>, Επίσημος διαδικτυακός ιστότοπος με ελεύθερο υλικό και πληροφορίες για την plotly. |
| [7] | JQUERY.<https://jquery.com>, Επίσημος διαδικτυακός ιστότοπος με ελεύθερο υλικό και πληροφορίες για την jquery. |
|  |  |