Τ.Ε.Ι ΠΕΙΡΑΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Οπτική Αναγνώριση Χαρακτήρων - Εφαρμογές

Σπουδαστής : Πέτικας Φοίβος

Α.Μ. 36964

Επιβλέπων : Έλληνας Ιωάννης

ΙΟΥΝΙΟΣ 2014

Περιεχόμενα

Σκοπός Δημιουργίας της Πτυχιακής Εργασίας

Ο απώτερος σκοπός της πτυχιακής εργασίας είναι εκτός από την κατανόηση και την δημιουργία εφαρμογών που αφορούν την οπτική αναγνώριση κειμένου, η συνειδητοποίηση των λύσεων που ο προγραμματισμός μπορεί να προσφέρει στη ζωή μας αν τον αξιοποιήσουμε σωστά αλλά και η πρόκληση ενδιαφέροντος στο συγκεκριμένο θέμα προς τον κάθε αναγνώστη, ούτως ώστε να μπορέσει ο ίδιος μέσω αυτής να διευρύνει τις γνώσεις του όσον αφορά την επεξεργασία εικόνας και πιο συγκεκριμένα την οπτική αναγνώριση χαρακτήρων.

Απευθύνεται είτε σε φοιτητές είτε σε καθηγητές αλλά και γενικά σε οποιονδήποτε ενδιαφέρεται να μάθει περισσότερα για το συγκεκριμένο θέμα και να τον οδηγήσει σε περαιτέρω αναζήτηση.

Ο σκοπός των εφαρμογών είναι η κατανόηση των τρόπων αξιοποίησης προγραμμάτων (Matlab) αλλά και γλωσσών προγραμματισμού (γλώσσα Matlab,javascript,html,css κλπ) για τη δημιουργία εφαρμογών οι οποίες θα μπορούν μέσω επεξεργασίας με διάφορες εντολές και όχι μόνο να εξάγουν αξιοποιήσιμο αποτέλεσμα κειμένου, έχοντας ως «είσοδο» μία εικόνα ή φωτογραφία κειμένου.

Έτσι η σύγχρονη εποχή όπου η ψηφιοποιημένη φωτογραφία βρίσκεται σχεδόν σε κάθε συσκευή δίπλα μας(κινητά τηλέφωνα, λαπτοπ , Η/Υ ,tablets κλπ) μπορεί να «εκμεταλλευτεί» ακόμα περισσότερο την τεχνολογική πρόοδο της ψηφιοποίησης διευκολύνοντας ακόμα περισσότερο τους ανθρώπους μέσω εφαρμογών όπως αυτές που θα παρουσιάσω παρακάτω .

Στα επόμενα κεφάλαια αναπτύσσεται η ιδέα, το υλικό και η λογική που έκαναν δυνατή και υλοποιήσημη την εκπόνηση της πτυχιακής εργασίας.

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Επεξεργασία εικόνας ονομάζεται κάθε μορφή αλγοριθμικής επεξεργασίας, ανάλυσης και χειρισμού ψηφιακών δεδομένων εικόνας ή βίντεο, όπως και το σχετικό επιστημονικό πεδίο της πληροφορικής. 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 , αλλά εκεί υφίσταται επικάλυψη με το συγγενές γνωστικό πεδίο της τεχνητής νοημοσύνης ονόματι μηχανική όραση. Μεγάλο μέρος του επιστημονικού υποβάθρου της επεξεργασίας εικόνας παρέχεται από την επεξεργασία σήματος, καθώς η ψηφιακή εικόνα μπορεί να θεωρηθεί διδιάστατο χωρικό σήμα και το βίντεο τριδιάστατο χωροχρονικό σήμα.

1.1 Τύποι Επεξεργασίας

Γεωμετρικές μετατροπές: Αλλαγή στο μέγεθος ολόκληρης ή τμήματος της εικόνας, περιστροφή, παραμόρφωση, αλλαγή προοπτικής, αλλαγή ανάλυσης (σε ψηφιογραφικές εικόνες) κτλ.

Χρωματικές μετατροπές και διορθώσεις: Αλλαγή των χρωματικών τόνων μιας εικόνας, ρύθμιση φωτεινότητας, αντίθεσης, αλλαγή του χρωματικού χώρου (μοντέλου), π.χ. από RGB σε CMYK.

Συμπίεση και μετατροπή της μορφής αποθήκευσης μιας εικόνας στον υπολογιστή (file conversion), π.χ. από μορφή .jpg σε μορφή .tif.

Εφαρμογή φίλτρων με στόχο τη βελτίωση της ποιότητας της εικόνας ή τον τονισμό γνωρισμάτων της (αφαίρεση αμυχών, εξάλειψη φαινομένου «κόκκινων ματιών» από εικόνες προσώπων, ανίχνευση ακμών και τονισμός των περιγραμμάτων, μείωση θορύβου κτλ).

Ανάμιξη δύο ή περισσότερων εικόνων ώστε να αποτελούν μία (φωτομοντάζ).

Κατάτμηση της εικόνας σε περιοχές, με στόχο τον καθορισμό των τομέων ενδιαφέροντος στην εικόνα (Regions of Interest, ROI). Ενδεικτικά, μπορεί να γίνει δυαδική κατάτμηση σε προσκήνιο και παρασκήνιο.

Αποκατάσταση, με στόχο την εξαγωγή μίας «ορθής» εκδοχής της εικόνας από μία ενθόρυβη / θολωμένη / παραμορφωμένη εικόνα εισόδου.

Ανεξάρτητα από την κατηγορία επεξεργασίας, όπως αναφέρθηκε πιο πάνω, η διαδικασία σχεδίασης και υλοποίησης των αλγορίθμων επεξεργασίας απαιτεί πολύ καλές γνώσεις μαθηματικών. Το σύγχρονο, έτοιμο λογισμικό επεξεργασίας απαλλάσσει, φυσικά, το χρήστη από την ανάγκη να διαθέτει αυτές τις γνώσεις.

1.2 Εφαρμογές και σχέση με άλλα επιστημονικά πεδία

Εφαρμογές:

- Γραφιστική
- Φωτογραφία
- Επεξεργασία εικόνας για ιατρικούς σκοπούς
- Ανάλυση μικροσκοπικών παρατηρήσεων
- Ταυτοποίηση προσώπων (σύγκριση στα χαρακτηριστικά προσώπου, δακτυλικών αποτυπωμάτων κτλ.)
- Εφαρμογές στην τεχνητή νοημοσύνη: μηχανική όραση

Σε σχέση με άλλα, συγγενή γνωστικά πεδία, η ψηφιακή επεξεργασία εικόνας διαφοροποιείται ως εξής:

Η ψηφιακή επεξεργασία εικόνας εξετάζει αλγορίθμους οι οποίοι δέχονται ως είσοδο εικόνες / βίντεο και παράγουν ως έξοδο εικόνες / βίντεο.

Τα γραφικά υπολογιστή εξετάζουν αλγορίθμους οι οποίοι δέχονται ως είσοδο συμβολικές περιγραφές οπτικών σκηνών και παράγουν ως έξοδο εικόνες / βίντεο (με ή χωρίς αλληλεπίδραση με τον χρήστη).

Η μηχανική όραση εξετάζει αλγορίθμους οι οποίοι δέχονται ως είσοδο εικόνες / βίντεο και παράγουν συμβολικές περιγραφές των εν λόγω οπτικών σκηνών.

Εμείς σε αυτήν την πτυχιακή εργασία θα ασχοληθούμε με την οπτική αναγνώριση χαρακτήρων ή αλλιώς optical character recognition (O.C.R) που είναι εφαρμογή της ψηφιακής επεξεργασίας εικόνας αλλά και πεδίο έρευνας της αναγνώρισης προτύπων, της τεχνητής νοημοσύνης και της μηχανικής όρασης.

ΨΗΦΙΑΚΗ ΕΙΚΟΝΑ

Μία ψηφιακή εικόνα i[m, n] αναπτύσσεται σε ένα διακριτό χώρο δύο διαστάσεων και παράγεται από την ψηφιοποίηση μιας αναλογικής εικόνας i[x,y] που αναπτύσσεται σε ένα συνεχή χώρο με διαστάσεις x και y.

Η συνεχής εικόνα διαιρείται σε N σειρές και M στήλες. Τα σημεία τομής των σειρών με τις στήλες είναι τα pixels. Οι τιμές χρωματικής πληροφορίας που εκχωρούνται στα σημεία αυτά δημιουργούν την ψηφιακή εικόνα a[m, n] όπου m={0, 1, 2,..., M-1} και n={0, 1, 2,..., N-1}.

Μία ψηφιακή εικόνα μπορεί να είναι δυαδική (binary image), μονοχρωματική με αποχρώσεις του γκρι (gray scale image) ή εγχρωμη (color image).

Η μονοχρωματική ψηφιακή εικόνα είναι ένας δισδιάστατος πίνακας MxN εικονοστοιχείων/pixels, που οι τιμές τους αντιπροσωπεύουν τη φωτεινότητα. Η δυαδική από την άλλη, μορφή μιας ψηφιακής εικόνας έχει μονάχα δύο τιμές φωτεινότητας, συνήθως 0 και 1 που αντιστοιχούν στο μαύρο και στο άσπρο αντίστοιχα. Σε δυαδική μορφή μπορούν να αποθηκευτούν διάφορες πληροφορίες για μία εικόνα.

Η έγχρωμη ψηφιακή εικόνα που είναι και η πιο διαδεδομένη ουσιαστικά αποτελείται απο τρεις μονοχρωματικές εικόνες. Η κάθε τέτοια εικόνα αντιπροσωπεύει διαφορετικές φωτεινότητες ή πιο συγκεκριμένα διαφορετικό πίνακα φωτεινοτήτων. Στο χρωματικό μοντέλο RGB (red,green,blue) υπάρχουν τρεις πίνακες που ο κάθε ένας αντιπροσωπεύει και ένα χρώμα, είτε κόκκινο είτε πράσινο είτε μπλε. Έτσι το τελικό χρώμα των εικονοστοιχείων προκύπτει από τον συνδυασμό των τριών συνιστωσών από τους τρεις διαφορετικούς πίνακες φωτεινοτήτων. Εάν οι τρεις αυτοί πίνακες έχουν τις ίδιες τιμές, τότε μιλάμε επί της ουσίας για έναν πίνακα δηλαδή για μία μονοχρωματική εικόνα.

Η οπτική αναγνώριση χαρακτήρων με την οποία θα ασχοληθούμε χρησιμοποιεί δυαδικές εικόνες για να γίνεται ευκολότερα και πιο γρήγορα η αναγνώριση προτύπων και η αντιστοίχισή τους.

2.1 Χρωματικά Μοντέλα

Το χρώμα είναι η υποκειμενική αντίληψη που αναπτύσσουμε στα διάφορα μήκη κύματος του φωτός, στην ορατή περιοχή του φάσματος (από400 μέχρι700nm). Η αίσθησή μας για το χρώμα επομένως είναι μια αυτοματοποιημένη εμηνευτική αντίδραση του ανθρώπινου εγκεφάλου στο μήκος κύματος της ηλεκτρομανητικής ακτινοβολίας και όχι κάποια εξωτερική ουσία.

Υπάρχουν πολλά χρωματικά μοντέλα που το καθένα μπορεί να χρησιμοποιεί διαφορετικές παραμέτρους για την ακριβή περιγραφή των διαφόρων χρωμάτων. Κοινό χαρακτηριστικό τους είναι πως υιοθετούν τρείς παραμέτρους, δηλ. χρειάζονται τρεις ανεξάρτητες τιμές για να προσδιορίσουν μαθηματικά κάποιο χρώμα.

Παραδείγματα τέτοιων μοντέλων:

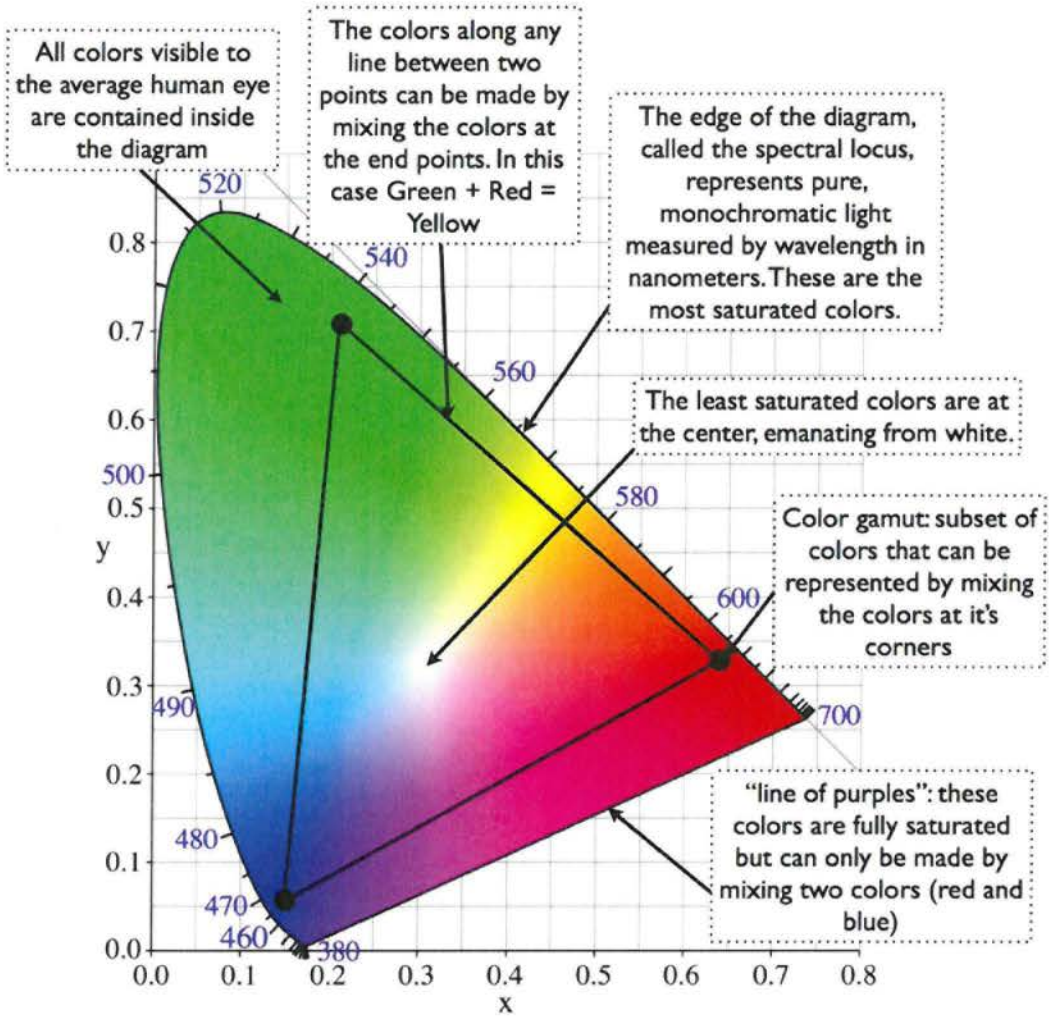
- Red, Green, Blue (RGB)
- Cyan, Magenta, Yellow (CMY)
- Hue, Lightness, aturation (HLS) - απόχρωση, φωτεινότητα, καθαρότητα
- Hue, Saturation, rightness (HSB) - απόχρωση, καθαρότητα λάμψη

2.1.1 Χρωματικό διάγραμμα CIE

Το 1931 η CIE (Comission Internationale de l'Eclairage, International Commision of Illumination) δημιούργησε ένα χρωματικό διάγραμμα σε μια προσπάθεια ακριβούς καταγραφής της τριχρωματικής σύνθεσης των χρωμάτων.

Βάση για τη σύνθεση του διαγράμματος απετέλεσε αυτό που η CIE ονόμασε «τυπικό παρατηρητή» (standard observer). Το διάγραμμα δείχνει το εύρος των χρωμάτων που μπορεί να δει ο τυπικός παρατηρητής.

Ουσιαστικά το χρωματικό διάγραμμα της CIE παρουσιάζει τις αναλογίες των πρωτευόντων χρωμάτων που πρέπει να χρημοποιηθούν ώστε να δημιουργήσουν την αίσθηση ενός συγκεκριμένου χρώματος για τον μέσο παρατηρητή.



Anatomy of a CIE Chromaticity Diagram

Εικόνα 1: Χρωματικό Διάγραμμα CIE

2.1.2 Προσθετικό μοντέλο (RGB)

Στο προσθετικό μοντέλο κάθε άλλο χρώμα δημιουργείται από ανάμιξη (πρόσθεση) των τριών πρωτευόντων χρωμάτων σε ποικίλες αναλογίες.

- Blue + Green= Cyan
- Red + Blue = Magenta
- Green + Red = Yellow
- Red + Blue + Green = White
- Εφαρμογή: όταν οι ακτινοβολίες προσπίπτουν άμεσα στο μάτι του παρατηρητή (πχ. Οθόνες CRT)

2.1.3 Αφαιρετικό Μοντέλο (CMY)

Στο αφαιρετικό μοντέλο τα πρωτεύοντα χρώματα είναι αυτά που σχηματίζονται από ανάμειξη ίσων ποσοτήτων των R, G και B.

- Cyan (Κυανό) (Blue + Green)
- Magenta (Πορφυρό) (Red + Blue)
- Yellow (Κίτρινο) Green + Red)

Οι αποχρώσεις δημιουργούνται αφαιρώντας από το προσπίπτον λευκό τις αποχρώσεις που απορροφά η χρωστική

- Red = white - Green (Yellow+Cyan) - Blue (Magenta+Cyan)

2.2 Ανάλυση Εικόνας και Βάθος Χρώματος

Το μέγεθος που δείχνει από πόσα pixels αποτελείται μια ψηφιακή εικόνα στη μονάδα του μήκους λέγεται «ανάλυση εικόνας» (image resolution) και μετριέται σε ppi(pixels per inch).

Η ανάλυση της εικόνας προκύπτει από τη συχνότητα δειγματοληψίας:

δηλώνει τον αριθμό των δειγμάτων στη μονάδα του μήκους που δημιουργούν τη ψηφιακή εικόνα.

Συνήθεις μονάδες:

- ppi= pixel per inch (οθόνες)
- dpi= dots per inch (εκτύπωση)
- spi= samples per inch

Βάθος χρώματος είναι το εύρος των δυαδικών ψηφίων που θα χρησιμοποιήσει ένας υπολογιστής για να αναπαραστήσει το χρώμα κάθε εικονοστοιχείου (πίξελ, pixel) μιας εικόνας. Το εύρος αυτό εκφράζεται ως δύναμη του 2 (επειδή η αναπαράσταση στον υπολογιστή είναι δυαδική) και, κατά συνέπεια, μια εικόνα μπορεί να έχει βάθος χρώματος:

$2^1 = 2$: Ασπρόμαυρη εικόνα (χωρίς διαβαθμίσεις γκρίζου)

$2^8 = 256$ χρώματα (ή αποχρώσεις του γκρίζου)

$2^{16} = 65536$ χρώματα. Η εικόνα με αυτό το βάθος χρώματος αναφέρεται και ως Highcolor

$2^{24} = 16.777.216$ χρώματα. Η εικόνα με αυτό το βάθος χρώματος αναφέρεται και ως Truecolor

2^{48} = Αυτό το βάθος χρώματος υπερβαίνει την διακριτική ικανότητα του ανθρώπινου οφθαλμού. Χρησιμοποιείται, ωστόσο, για πρακτικούς λόγους, από πολλούς σαρωτές.

Οι εικόνες που αποτελούνται από 256 χρώματα (ή λιγότερα) αποθηκεύονται συνήθως στην μνήμη του υπολογιστή υπό μορφή μιας παλέτας χρωμάτων. Για βάθη μεγαλύτερα από 8 bit, το κάθε εικονοστοιχείο αναπαρίσταται από ανάλογες διαβαθμίσεις των τριών χρωμάτων RGB (κόκκινο, πράσινο και μπλε).

Το βάθος χρώματος των 16 bits "διαιρείται", συνήθως, σε πέντε bits για κάθε ένα από τα χρώματα κόκκινο και μπλε, και έξι bits για το πράσινο, δεδομένου ότι το ανθρώπινο μάτια είναι πιο ευαίσθητα στην διάκριση διαβαθμίσεων του πράσινου σε σχέση με τα άλλα δύο χρώματα. Άλλες φορές το 16ο bit αναπαριστά τυχόν διαφάνεια του χρώματος.

Στα βάθος χρώματος των 24 bits υπάρχουν 8 bits ανά βασικό χρώμα, δηλαδή 28 = 256 διαβαθμίσεις κάθε βασικού χρώματος. Μερικές φορές μπορεί να χρησιμοποιηθεί και βάθος χρώματος των 32 bits. Σε αυτήν την περίπτωση τα 8 επιπλέον bits χρησιμοποιούνται για να δηλωθεί η συνοχή του χρώματος.

ΟΠΤΙΚΗ ΑΝΑΓΝΩΡΙΣΗ ΧΑΡΑΚΤΗΡΩΝ (O.C.R)

Η Οπτική Αναγνώριση Χαρακτήρων (Optical Character Recognition) ή αλλιώς Αυτόματη Αναγνώριση Χαρακτήρων Κειμένου ονομάζεται η διαδικασία μετατροπής σαρωμένων εικόνων χειρογράφων ή έντυπων κειμένων σε κείμενο αναγνώσιμο από ηλεκτρονικό υπολογιστή. Η Οπτική Αναγνώριση Χαρακτήρων καθιστά εφικτή την εκ νέου επεξεργασία του κειμένου, αποφεύγοντας την δακτυλογράφηση του από την αρχή.

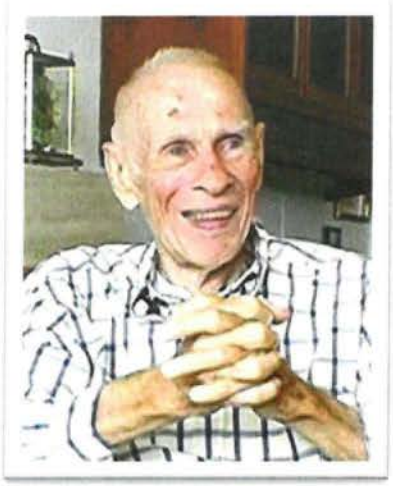
Τα συστήματα Οπτικής Αναγνώρισης Χαρακτήρων απαιτούν βαθμονόμηση για να διαβάσουν μια συγκεκριμένη γραμματοσειρά. Οι πρώτες εκδόσεις ήταν προγραμματισμένες με εικόνες για κάθε χαρακτήρα και δούλευαν μια γραμματοσειρά την φορά. Τα ευφυή συστήματα με υψηλό δείκτη αναγνώρισης είναι πλέον κοινά. Μερικά συστήματα είναι ικανά να αναπαράγουν ακόμη και τις πληροφορίες που δεν είναι κείμενο σε ένα έγγραφο, όπως εικόνες, στήλες, γραμμές, γωνίες κτλ.

3.1 Ιστορική Αναδρομή

Το 1929, ο Gustav Tauschek απέκτησε ευρεσιτεχνία για το OCR στην Γερμανία, ακολουθούμενος από τον Paul W. Handel που απέκτησε την ευρεσιτεχνία για την OCR στις Ηνωμένες Πολιτείες το 1933. Το 1935 ο Tauschek πήρε επίσης την ευρεσιτεχνία στην μέθοδο του στις ΗΠΑ. Το μηχάνημα του Tauschek ήταν μια μηχανική συσκευή που χρησιμοποιούσε πρότυπα και αισθητήρα φωτός.

Το 1949, οι μηχανικοί της RCA δημιούργησαν τον πρώτο OCR σύστημα για να βοηθήσουν τους τυφλούς για το US Veterans Administration, αλλά αντί να μετατρέπουν εκτυπωμένους χαρακτήρες σε χαρακτήρες αναγνώσιμους από υπολογιστή, η συσκευή τους μετέτρεπε και τους διάβασε. Η συσκευή είχε υψηλό κόστος και δεν δόθηκε για παραγωγή.

Το 1950, ο David H. Shepard, ένας κρυπταναλητής των Armed Forces Security Agency των ΗΠΑ δημιούργησε μια συσκευή που μετέτρεπε τα εκτυπωμένα μηνύματα σε κείμενο αναγνώσιμο από ηλεκτρονικό υπολογιστή αφού έκδωσε την δική του πατέντα. Έπειτα, ο Shepard ίδρυσε την Intelligent Machines Research Corporation (IMR), η οποία ήταν η πρώτη που έβαλε σε εμπορική λειτουργία τα συστήματα OCR.



David H. Shepard

Το 1955, το πρώτο εμπορικό σύστημα εγκαταστάθηκε στο Reader's Digest. Το δεύτερο σύστημα πουλήθηκε στην Standard Oil για να διαβάζει αριθμούς πιστωτικών καρτών για λογαριασμούς. Άλλα συστήματα που πουλήθηκαν από την IMR γύρω στο 1950s είχαν αναγνωστέα αποκόμματος λογαριασμού στην Ohio Bell Telephone Company και έναν σαρωτή σελίδας στις United States Air Force για ανάγνωση και μετάδοση χειρόγραφων μηνυμάτων από τον. Η IBM και άλλες αγόρασαν τις άδειες ευρεσιτεχνίας OCR του Shepard.

Το 1965, το Reader's Digest και η RCA συνεργάστηκαν για να φτιάξουν μια συσκευή OCR για να διαβάζει και να ψηφιοποιεί τους σειριακούς αριθμούς από τα κουπόνια του Reader's Digest από τις διαφημίσεις. Οι γραμματοσειρά που χρησιμοποιήθηκε για την εκτύπωση των κουπονιών ήταν η OCR-A font. Η συσκευή ήταν συνδεδεμένη σε ένα RCA 301 υπολογιστή. Η συσκευή επίσης είχε έναν ειδικό αναγνωστέα TWA. Η συσκευή μπορούσε να επεξεργαστεί 1,500 έγγραφα ανά λεπτό, απορρίπτοντας ότι δεν μπορεί να αναγνωρίσει σωστά.

Το Ταχυδρομείο των ΗΠΑ χρησιμοποιεί τεχνολογία οπτικής αναγνώρισης από το 1965 βασιζόμενο σε τεχνολογία που ανέπτυξε ο εφευρέτης Jacob Rabinow. Η πρώτη χρήση της Οπτικής Αναγνώρισης στην Ευρώπη έγινε από το Ταχυδρομείο της Αγγλίας. Το 1965 ξεκίνησε την κατασκευή ενός τραπεζικού συστήματος βασιζόμενο στην τεχνολογία OCR, μια διαδικασία που έφερε επανάσταση στα συστήματα πληρωμής λογαριασμών στην Μ. Βρετανία. Το ταχυδρομείο του Καναδά υιοθέτησε τα συστήματα OCR από το 1971.

Υπολογιστής RCA 301

Το 1974, ο Ray Kurzweil ίδρυσε την εταιρία Kurzweil Computer Products, Inc. και δημιούργησε το πρώτο σύστημα οπτικής αναγνώρισης χαρακτήρων που αναγνώριζε εκτυπωμένο κείμενο διαφόρων γραμματοσειρών. Η εταιρία εστίασε στην δημιουργία μιας συσκευής που θα βοηθήσει τους τυφλούς να διαβάζουν κείμενο με βοήθεια υπολογιστή. Η συσκευή απαιτούσε την εφεύρεση δύο τεχνολογιών – μια συσκευή σάρωσης και ένα σύστημα ανάγνωσης κειμένου από τον υπολογιστή.

Το 1978, η εταιρία Kurzweil Computer Products άρχισε να πουλά εταιρικές εκδόσεις του λογισμικού οπτικής αναγνώρισης. Η LexisNexis ήταν από τους πρώτους πελάτες που αγόρασαν το λογισμικό για να μεταφορτώνουν έγγραφα στην online βάση δεδομένων τους. Δύο χρόνια μετά, ο Kurzweil πούλησε την εταιρία στην Xerox, που έδειξε ενδιαφέρον για την επέκταση της τεχνολογίας οπτικής αναγνώρισης.

3.2 Πως λειτουργεί

Υπάρχουν δύο κύριοι τρόποι εφαρμογής της Οπτικής Αναγνώρισης, η "Αντιστοίχηση με Πρότυπα" και η "Εξαγωγή Χαρακτηριστικών". Η πρώτη μέθοδος είναι πιο διαδεδομένη και κοινή αλλά περιορίζεται αρκετά σε σχέση με την 2η τεχνική. Η σημερινή τεχνολογία χρησιμοποιεί τον συνδυασμό και των δύο τεχνολογιών για την καλύτερη επίτευξη αποτελεσμάτων, κυρίως σε χειρόγραφα έγγραφα.

- Αντιστοίχηση με πρότυπα

Η αντιστοίχηση με πρότυπα αφορά την αναγνώριση χαρακτήρων από έτοιμα πρότυπα ή περιγράμματα χαρακτήρων. Ο σαρωτής ψηφιοποιεί την εικόνα ενός εγγράφου στον υπολογιστή και το λογισμικό Οπτικής Αναγνώρισης προσπαθεί να ταιριάξει, με ένα βαθμό πιθανότητας, τους χαρακτήρες από το σαρωμένο αρχείο εικόνας με τα πρότυπα που έχει αποθηκευμένα. Αν η εικόνα ενός χαρακτήρα αντιστοιχεί με αναγνωρισμένο χαρακτήρα, τότε αντιστοιχίζεται με χαρακτήρα κειμένου για τον ηλεκτρονικό υπολογιστή.

Τα περισσότερα εκτυπωμένα έγγραφα κειμένου ήταν με γραμματοσειρές Times, Courier ή Helvetica με μέγεθος 10 ως 14. Ένα πρόγραμμα αναγνώρισης χαρακτήρων έχει εικόνες σε μορφή bitmap για κάθε χαρακτήρα κάθε μεγέθους κάθε γραμματοσειράς. Το λογισμικό διάβαζε την εικόνα που σάρωνε ο σαρωτής γραμμή-γραμμή και προσπαθούσε να αντιστοιχήσει κάθε χαρακτήρα με την αντίστοιχη εικόνα. Για παράδειγμα αν το πρόγραμμα εντόπιζε ένα χαρακτήρα "Γ" τότε το πρόγραμμα έψαχνε όλα τα πρότυπα από το Α μέχρι το ω σε όλα τα αποθηκευμένα μεγέθη και αν εντόπιζε κάποια εικόνα που έμοιαζε το Γ, το αντιστοίχιζε.

Η όλη διαδικασία είναι χρονοβόρα γιατί απαιτούνται πολλές επαναλήψεις για κάθε χαρακτήρα.

- Εξαγωγή Χαρακτηριστικών

Η εξαγωγή χαρακτηριστικών είναι επίσης γνωστή ως Ευφυής Αναγνώριση Χαρακτήρων (Αγγλ. Intelligent Character Recognition – ICR), ή τοπολογική ανάλυση χαρακτηριστικών. Πρόκειται για ένα είδος οπτικής αναγνώρισης που δεν βασίζεται σε ακριβείς αντιστοιχήσεις με πρότυπα. Το λογισμικό λειτουργεί με ένα πιο σοφιστικό τρόπο αναγνώρισης χαρακτήρων, όπως ανίχνευση επιμέρους συστατικών στοιχείων ενός χαρακτήρα, όπως γωνίες, γραμμές, ενώσεις κτλ) Η εφαρμογή των αντιστοιχήσεων γίνεται με μορφή κανόνων.

Ένας κανόνας θα μπορούσε να είναι ως έξης: Αν εντοπιστούν δύο κάθετες που κλίνουν οι μια στην άλλη "/" και "\" και η κορυφές τους ενώνονται και στο κέντρο υπάρχει μια γραμμή "-" τότε είναι το γράμμα "A". Η εφαρμογή αυτού του κανόνα θα μπορούσε να εντοπίσει όλα τα "A" ανεξάρτητα από την μέγεθος ή τον τύπο γραμματοσειράς που χρησιμοποιήθηκε στο έγγραφο.

- Υβριδική Αναγνώριση

Οι παραπάνω μέθοδοι χρησιμοποιούνται κυρίως για αναγνώριση κειμένου που εκτυπώθηκε από ηλεκτρονικό υπολογιστή ή δακτυλογραφήθηκε. Η αναγνώριση χειρόγραφων χαρακτήρων είναι πιο πολύπλοκη διαδικασία και απαιτεί τον συνδυασμό των παραπάνω τεχνικών, καθώς και στοιχεία όπως γνώσεις για τον συγγραφέα και το περιεχόμενο του κειμένου.

Τα προβλήματα με την αναγνώριση χειρογράφων οφείλονται στην καλλιγραφία (συνεχόμενη γραφή χαρακτήρων χωρίς κενό) διότι δεν μπορούν να ξεχωρίσουν πότε τελειώνει ένα γράμμα και πότε ξεκινάει ένα άλλο. Επίσης, κάθε άνθρωπος έχει διαφορετικό γραφικό χαρακτήρα, δυσχεραίνοντας την διαδικασία εφαρμογής προτύπων ή εξαγωγής χαρακτηριστικών για τον κάθε ένα. Όταν ένα λογισμικό πρέπει να αναγνωρίσει τέτοιες λέξεις, χρησιμοποιεί το νόημα του κειμένου, την γνώση του για τον συγγραφέα και τις λέξεις που ήδη αναγνώρισε.

3.3 Λογισμικό Αναγνώρισης Χαρακτήρων

- Desktop & Server Λογισμικό Αναγνώρισης Χαρακτήρων

Το λογισμικό Οπτικής Αναγνώρισης και Ευφυούς Αναγνώρισης χαρακτήρων είναι συστήματα τεχνίτης νοημοσύνης που θεωρούν το κείμενο ως μια ακολουθία χαρακτήρων και όχι μεμονωμένες λέξεις ή φράσεις. Βασιζόμενα στην ανάλυση των γραμμών και των καμπυλών κάθε χαρακτήρα, προσπαθούν να μαντέψουν ποιος χαρακτήρας απεικονίζεται χρησιμοποιώντας βάσεις με πρότυπα που ταιριάζει.

- WebOCR & OnlineOCR

Με την ανάπτυξη της τεχνολογία της πληροφορίας, οι πλατφόρμες χρήσης λογισμικού αναγνώρισης χαρακτήρων άλλαξαν σε πολύ-πλατφόρμες με την χρήση του ηλεκτρονικού υπολογιστή, του διαδικτύου, του υπολογιστικού νέφους και τις κινητές συσκευές. Μετά από 30 χρόνια, το λογισμικό οπτικής αναγνώρισης υιοθετεί νέες μεθόδους όπως χρήση της αναγνώρισης χαρακτήρων ως υπηρεσία ιστού. Χωρίς την χρήση εξειδικευμένο λογισμικού ή την υπολογιστική ισχύ ενός υπολογιστή, ο χρήστης μπορεί να χρησιμοποιήσει την αναγνώριση χαρακτήρων με εξαιρετικά αποτελέσματα.

- OCR Ειδικής Χρήσης

Λόγω του μεγάλου εύρους χρήσης της τεχνολογίας Οπτικής Αναγνώρισης Χαρακτήρων, υπήρξε η ανάγκη ανάπτυξης λογισμικού ειδικής χρήσης. Το λογισμικό ειδικής χρήσης δίνει καλύτερα αποτελέσματα σε συγκεκριμένες περιπτώσεις, παρά σε γενικές. Το λογισμικό χρησιμοποιεί κάποιους κανόνες ή κάποια φίλτρα που αντιστοιχούν μόνο σε ορισμένες εικόνες κειμένων και εξάγει το κείμενο. Για παράδειγμα, κάποιο λογισμικό αναγνώρισης των χαρακτηριστικών μιας ταυτότητας, θα πρέπει να εφαρμόσει ειδικά φίλτρα και να διαβάσει ορισμένες περιοχές για να είναι πιο πετυχημένη η αναγνώριση.

ΠΕΡΙΒΑΛΛΟΝ MATLAB

Το MATLAB (matrix laboratory) είναι ένα περιβάλλον αριθμητικής υπολογιστικής και μια προγραμματιστική γλώσσα τέταρτης γενιάς. Αποθηκεύει και κάνει τις πράξεις με βάση την άλγεβρα μητρών. Η τρέχουσα έκδοσή του είναι η R2013b η οποία κυκλοφόρησε τον Σεπτέμβριο του 2013.

4.1 Εισαγωγή

Χρησιμοποιείται κατά κύριο λόγο για την επίλυση μαθηματικών προβλημάτων, ωστόσο είναι πολύ "ισχυρό" και μπορεί να χρησιμοποιηθεί και για προγραμματισμό καθώς περιέχει εντολές από την C++ όπως την while, την switch και την if. Στον τομέα των γραφικών όσον αφορά τον μαθηματικό κλάδο μπορεί να υλοποιήσει συναρτήσεις πραγματικές, μιγαδικές, πεπλεγμένες συναρτήσεις δύο μεταβλητών και άλλες. Όσον αφορά τον στατιστικό κλάδο μπορεί να υλοποιήσει ιστογράμματα, τομεογράμματα, ραβδοδιαγράμματα, εμβαδογράμματα και άλλα.

4.2 Προγραμματισμός στο Matlab

Όπως υποδηλώνεται και από το όνομά του, το MATLAB είναι ειδικά σχεδιασμένο για υπολογισμούς με πίνακες, όπως η επίλυση γραμμικών συστημάτων, η εύρεση ιδιοτιμών και ιδιοδιανυσμάτων, η αντιστροφή τετραγωνικών πινάκων κλπ. Επιπλέον είναι εφοδιασμένο με πολλές επιλογές για γραφικά (δηλ. την κατασκευή γραφικών παραστάσεων) και προγράμματα γραμμένα στη δική του γλώσσα προγραμματισμού για την επίλυση άλλων προβλημάτων όπως η εύρεση των ριζών μη γραμμικής εξίσωσης, η επίλυση μη γραμμικών συστημάτων, η επίλυση προβλημάτων αρχικών τιμών με συνήθεις διαφορικές εξισώσεις κα.

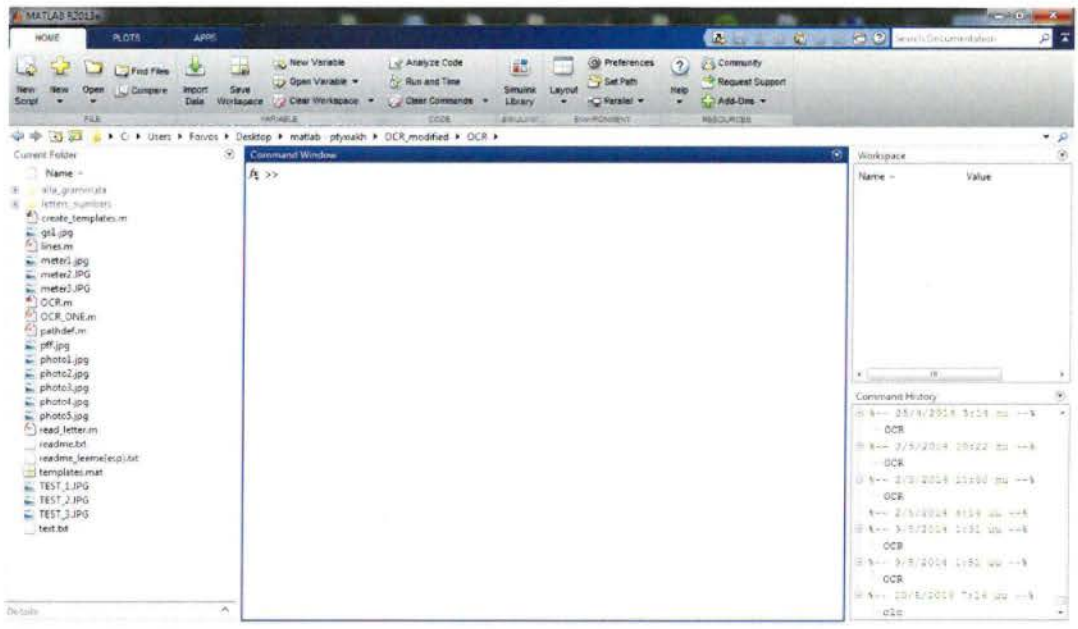
Η γλώσσα προγραμματισμού του MATLAB δίνει την ευχέρεια στον χρήστη να το επεκτείνει με δικά του προγράμματα. Το MATLAB είναι σχεδιασμένο για την αριθμητική επίλυση προβλημάτων σε αριθμητική πεπερασμένης ακρίβειας (finite-precision arithmetic), δηλαδή δεν βρίσκει την ακριβή αλλά μια προσεγγιστική λύση ενός προβλήματος. Αυτή είναι και η βασική του διαφορά από τα συστήματα συμβολικών υπολογισμών όπως η Maple και το Mathematica.

Η γλώσσα προγραμματισμού Matlab, έχει αρκετές ομοιότητες με την C. Διαθέτει και αυτή for, while, if κ.ο.κ. Όμως εκτός από τις μεγάλες δυνατότητες που προγραμματιστικά μας δίνει, το Matlab έχει και πολλές ακόμα εντολές για ψηφιακή επεξεργασία σήματος και συναρτήσεις μαθηματικών, μέχρι επεξεργασίας εικόνας, φίλτρων και εφέ.

Μπορούμε να φτιάξουμε φυσικά και τις δικές μας συναρτήσεις/functions τις οποίες μπορούμε και να τις καλέσουμε. Αυτό μπορεί να γίνει με την δημιουργία m-files ή απλών functions. Οι δυνατότητες του προγράμματος με λίγα λόγια πολλαπλασιάζονται και σε συνάρτηση με την ευκολία που δίνει το ίδιο το περιβάλλον όλα απλοποιούνται αρκετά αλλά και κάνουν την διαδικασία προγραμματισμού αρκετά πιο ευχάριστη και κατανοητή.

Παρακάτω θα δούμε διάφορες συναρτήσεις/εντολές επεξεργασίας εικόνας οι οποίες θα χρησιμοποιηθούν αργότερα στην δημιουργία της πρώτης εφαρμογής O.C.R .

Ο κώδικας μας στην εφαρμογή του Matlab αποτελείται από τέσσερα διαφορετικά m-files. Τα m-files δημιουργούνται εύκολα μέσω του Matlab (πάνω αριστερά New>Script για ένα εντελώς κενό m-file και New>Function για ένα function m-file) και λειτουργούν σαν scripts στην γλώσσα C περίπου. Εμείς θα χρησιμοποιήσουμε και τις δύο μορφές m-file. Παρακάτω μπορείτε να δείτε μία ενδεικτική εικόνα του περιβάλλοντος του Matlab.



Εικόνα 2: Περιβάλλον Matlab

4.3 Συναρτήσεις Επεξεργασίας Εικόνας στο Matlab

Σε αυτό το υποκεφάλαιο θα ασχοληθώ με κάποιες από τις πιο σημαντικές συναρτήσεις (εντολές) επεξεργασίας εικόνας στο Matlab, οι οποίες θα χρησιμοποιηθούν στην πρώτη εφαρμογή της οπτικής αναγνώρισης χαρακτήρων. Δεν θα γίνει επεξήγηση του τρόπου συγγραφής κώδικα αλλά και ούτε της γλώσσας προγραμματισμού Matlab μιας και σκοπός τις πτυχιακής δεν είναι η επεξήγηση του προγράμματος Matlab αλλά των εφαρμογών που δημιουργήθηκαν. Άλλωστε θεωρείται προαπαιτούμενη μία σχετική γνώση όσον αφορά τη χρήση του συγκεκριμένου προγράμματος σε θέματα προγραμματιστικά, μαθηματικά αλλά και όσων αφορούν την επεξεργασία σημάτων και στη δική μας περίπτωση την επεξεργασία εικόνας.

Σημαντικές Συναρτήσεις Επεξεργασίας Εικόνας στην Εφαρμογή O.C.R :

- image = imread('image.type');

  Η συνάρτηση imread χρησιμοποιείται για την ανάγνωση αρχείων με εικόνες διαφορετικών τύπων όπως:

  - BMP (Microsoft Windows Bitmap)
  - JPEG (Joint Photographic Experts Group)
  - PNG (Portable Network Graphics)
  - TIFF (Tagged Image File Format)

- imshow(image);

  Με αυτή τη συνάρτηση απεικονίζουμε μία εικόνα στην οθόνη. Μπορεί να χρησιμοποιηθεί με αρκετούς τρόπους για διαφορετικά αποτελέσματα απεικονίσεων, με κάποιους ενδεικτικούς να είναι:

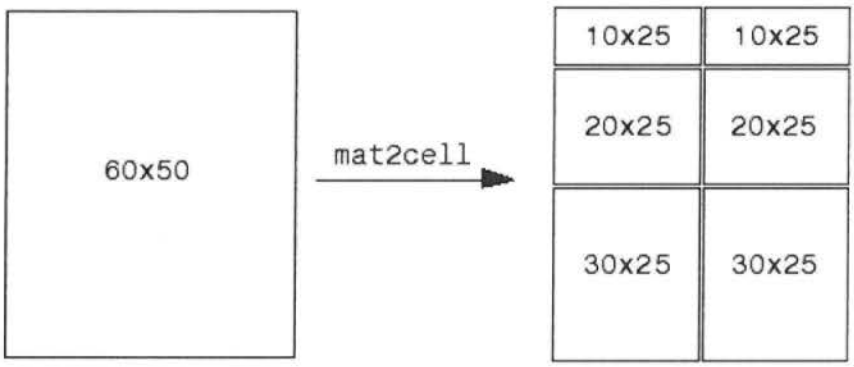
  - Imshow(image) - Απεικόνιση εικόνας φωτεινότητας.
  - Imshow(bitmap_image) - Απεικόνιση δυαδικής εικόνας.
  - Imshow('image.png') - Έτσι απεικονίζουμε ένα αρχείο εικόνας.

- imresize(image,scale);

  Η συνάρτηση αυτή αλλάζει τις διαστάσεις(μέγεθος) μίας εικόνας. Η εικόνα αυτή μπορεί να είναι grayscale,RGB ή binary. Εάν το scale στη συνάρτηση είναι μεταξύ 0 και 1.0, τότε η νέα εικόνα θα είναι μικρότερη από την αρχική(image). Εάν είναι μεγαλύτερο από 1.0 τότε θα είναι μεγαλύτερη.

- mat2cell

  Αυτή η συνάρτηση παίρνει έναν πίνακα και τον χωρίζει σε μικρότερους πίνακες, ανάλογα πάντα με τις μεταβλητές μεγέθους που θα ορίσουμε στην εντολή.Παράδειγμα :
  mat2cell(x, [10,20,30] , [25,25])
  Αν ο πίνακας x έχει διαστάσεις 60x50, το αποτέλεσμα θα είναι :



Εικόνα 3: Εντολή mat2cell

- corr2(A,B);

  Επιστρέφει τον συντελεστή συσχετισμού των A και B, είτε αυτά είναι μήτρες (matrices) είτε διανύσματα (vectors) του ίδιου μεγέθους.

- rgb2gray(image);

  Μετατρέπει μία έγχρωμη εικόνα RGB σε εικόνα φωτεινότητας (grayscale). Αυτό γίνεται αφαιρώντας την χροιά (απόχρωση) αλλά και την πληροφορία κορεσμού της έγχρωμης εικόνας, κρατώντας όμως την πληροφορίας φωτεινότητας.

- graythresh(image);

  Υπολογίζει το κατώφλι μίας εικόνας (image) , το οποίο μετά μπορεί να χρησιμοποιηθεί για να μετατρέψει μία έντονης φωτεινότητας εικόνα σε δυαδική εικόνα (binary image) με την συνάρτηση im2bw που εξηγείται παρακάτω. Χρησιμοποιέι την μέθοδο Otsu για τον υπολογισμό του threshold (κατώφλι). Επιστρέφει μία τιμή μεταξύ του 0.0 και του 1.0.

- im2bw(image,threshold);

  Μετατρέπει μία εικόνα (image) σε δυαδική εικόνα βασιζόμενη όμως στην τιμή του threshold (κατωφλίου). Βασιζόμενη στην τιμή αυτή , η συνάρτηση αυτή μετατρέπει όσα pixel είναι μεγαλύτερα από την τιμή αυτή (η οποία είναι πάντα μεταξύ 0.0 και 1.0) σε 1 (άσπρο) και όσα είναι μικρότερα από τη τιμή αυτή σε 0 (μαύρο). Αν δεν οριστεί το κατώφλι τότε αυτόματα ορίζεται σαν κατώφλι η τιμή 0.5.

Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVASCRIPT

5.1 Εισαγωγή

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, 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.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript.

5.2 Ιστορική Αναδρομή

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. LiveScript ήταν το επίσημο όνομα της γλώσσας όταν για πρώτη φορά κυκλοφόρησε στην αγορά σε βήτα (beta) εκδόσεις με το πρόγραμμα περιήγησης στο Web, Netscape Navigator εκδοχή 2.0 τον Σεπτέμβριο του 1995. LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου, 1995 , όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape εκδοχή 2.0B3.

Η JavaScript απέκτησε μεγάλη επιτυχία ως γλώσσα στην πλευρά του πελάτη (client-side) για εκτέλεση κώδικα σε ιστοσελίδες, και περιλήφθηκε σε διάφορα προγράμματα περιήγησης στο Web. Κατά συνέπεια, η εταιρεία Microsoft ονόμασε την εφάρμογή της σε JScript για να αποφύγει δύσκολα θέματα εμπορικών σημάτων. JScript πρόσθεσε νέους μεθόδους για να διορθώσει τα Y2K-προβλήματα στην JavaScript, οι οποίοι βασίστηκαν στην java.util.Date τάξη της Java. JScript περιλήφθηκε στο πρόγραμμα Internet Explorer εκδοχή 3.0, το οποίο κυκλοφόρησε τον Αύγουστο του 1996.

Τον Νοέμβριο του 1996, η Netscape ανακοίνωσε ότι είχε υποβάλει τη γλώσσα JavaScript στο Ecma International (μια οργάνωση της τυποποίησης των γλωσσών προγραμματισμού) για εξέταση ως βιομηχανικό πρότυπο, και στη συνέχεια το έργο είχε ως αποτέλεσμα την τυποποιημένη μορφή που ονομάζεται ECMAScript.

Η JavaScript έχει γίνει μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αρχικά, όμως, πολλοί επαγγελματίες προγραμματιστές υποτίμησαν τη γλώσσα διότι το κοινό της ήταν ερασιτέχνες συγγραφείς ιστοσελίδων και όχι επαγγελματίες προγραμματιστές (και μεταξύ άλλων λόγων). Με με τη χρήση της τεχνολογίας Ajax, η JavaScript γλώσσα επέστρεψε στο προσκήνιο και έφερε πιο επαγγελματική προσοχή προγραμματισμού. Το αποτέλεσμα ήταν ένα καινοτόμο αντίκτυπο στην εξάπλωση των πλαισίων και των βιβλιοθηκών, τη

βελτίωση προγραμματισμού με JavaScript, καθώς και αυξημένη χρήση της JavaScript έξω από τα προγράμματα περιήγησης στο Web.

Τον Ιανουάριο του 2009, το έργο CommonJS ιδρύθηκε με στόχο τον καθορισμό ενός κοινού προτύπου βιβλιοθήκης κυρίως για την ανάπτυξη της JavaScript έξω από το πρόγραμμα περιήγησης και μέσα σε άλλες τεχνολογίες (π.χ. server-side).

5.3 Προγραμματισμός στη Javascript

Η αρχική έκδοση της Javascript βασίστηκε στη σύνταξη στη γλώσσα προγραμματισμού C, αν και έχει εξελιχθεί, ενσωματώνοντας πια χαρακτηριστικά από νεότερες γλώσσες.

Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν ο φυλλομετρητής (browser) του χρήστη, και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο διακομιστή, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Αντίθετα, άλλες γλώσσες όπως η PHP εκτελούνται στο διακομιστή (server-side γλώσσες προγραμματισμού).

Παρά την ευρεία χρήση της Javascript για συγγραφή προγραμμάτων σε περιβάλλον φυλλομετρητή, αξίζει να σημειωθεί ότι από την αρχή χρησιμοποιήθηκε και για τη συγγραφή κώδικα από την πλευρά του διακομιστή, από την ίδια τη Netscape στο προϊόν LiveWire, με μικρή επιτυχία. Η χρήση της Javascript στο διακομιστή εμφανίζεται πάλι σήμερα, με τη διάδοση του Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα γεγονότα (events).

Ο κώδικας Javascript μιας σελίδας περικλείεται από τις ετικέτες της HTML <script type="text/javascript"> και </script>.

Για παράδειγμα, ο ακόλουθος κώδικας Javascript εμφανίζει ένα πλαίσιο διαλόγου με το κείμενο "Γεια σου, κόσμε!":

```
<script type="text/javascript">

alert('Γεια σου, κόσμε!');

</script>
```

Αν ο κώδικας Javascript περιέχει περισσότερες από μία εντολές, αυτές θα πρέπει να διαχωριστούν μεταξύ τους με το χαρακτήρα του ελληνικού ερωτηματικού ';' (δηλαδή της λατινικής άνω τελείας). Η χρήση του χαρακτήρα αυτού για την τελευταία εντολή δεν είναι απαραίτητη. Η διαχώριση των εντολών στους νεότερους φυλλομετρητές (browsers) δεν είναι απαραίτητη.

Μια άλλη βασική εντολή, η window.prompt("μήνυμα προς το χρήστη"), ζητάει από το χρήστη να συμπληρώσει ένα κομμάτι μιας αίτησης απευθείας ώστε τα δεδομένα να χρησιμοποιηθούν σαν κείμενο:

```
<script>

var FIRSTvariable = window.prompt("PLEASE FILL IN YOUR NAME")

alert("Your name is " + FIRSTvariable + ".")

</script>
```

Η Βάση της Πτυχιακής Εργασίας

Η πτυχιακή εργασία αποτελείται από δύο εφαρμογές οπτικής αναγνώρισης κειμένου (O.C.R όπως αναπτύξαμε σε προηγούμενη ενότητα) γραμμένες και ανεπτυγμένες η κάθεμια σε διαφορετικές γλώσσες. Η πρώτη εφαρμογή ειναι ανεπτυγμένη στο περιβάλλον Matlab και η δεύτερη στη γλώσσα προγραμματισμού Javascript χρησιμοποιώντας τις απαραίτητες βιβλιοθήκες για να φτάσει στο επιθυμητό αποτέλεσμα. Το αποτέλεσμα των δύο αυτών εφαρμογών είναι η εξαγωγή κειμένου από μία εικόνα.

Στο παρακάτω κεφάλαιο θα δούμε αναλυτικά τον κώδικα ο οποίος αναπτύχθηκε για να φτάσουμε στο απαραίτητο αποτέλεσμα σε κάθεμια από τις δύο εφαρμογές. Εκτός από τον κώδικα όμως θα γίνει και αναλυτική επεξήγηση των τρόπων ανάπτυξής του αλλά και των δομών του. Επίσης θα αναδειχθούν τα καίρια σημεία και οι σημαντικότερες εντολές και συναρτήσεις που χρησιμοποιούνται για να φτάσουμε στον επιθυμητό αποτέλεσμα.

Τέλος εκτός από τον σκοπό, τον πηγαίο κώδικα και την επεξήγησή του θα αναφερθούν και τα συμπεράσματα αλλά και τα θετικά και τα αρνητικά της κάθε εφαρμογής μετά την αποτελεσματική ολοκλήρωσή τους.

ΟΙ ΕΦΑΡΜΟΓΕΣ ΠΟΥ ΑΝΑΠΤΥΧΘΗΚΑΝ

6.1 Εφαρμογή O.C.R στο Matlab

Η πρώτη εφαρμογή οπτικής αναγνώρισης κειμένου ανεπτυγμένη στο προγραμματιστικό περιβάλλον του Matlab αποτελείται από τέσσερα m-files. Συγκεκριμένα τα : create_templates, lines, read_letter και OCR. Το πρώτο φτιάχνει ένα αρχείο templates.mat το οποίο θα εξηγήσουμε μετά πώς το κάνει και πού χρησιμεύει, το δεύτερο χωρίζει σε γραμμές την εικόνα που παίρνει η εφαρμογή σαν είσοδο, το τρίτο διαβάζει το κάθε γράμμα που προκύπτει διαδοχικά από την εκάστοτε γραμμή και το αντιστοιχεί με αυτό που είναι και το τελευταίο (OCR) είναι το εκτελέσιμο αρχείο.

Η εφαρμογή χρησιμοποιεί πολλές γνωστές εντολές προγραμματισμού του Matlab αλλά και συναρτήσεις επεξεργασίας εικόνας που αναφέραμε σε προηγούμενη ενότητα. Τα lines.m και read_letter.m λειτουργούν σαν συναρτήσεις , οι οποίες καλούνται αργότερα στο εκτελέσιμο. Το εκτελέσιμο αρχείο και το create_templates λειτουργούν σαν διαφορετικά scripts, το δεύτερο δημιουργώντας το templates.mat που χρησιμοποιείται μετά στο εκτελέσιμο και το εκτελέσιμο που εξάγει το αποτέλεσμα της επεξεργασίας σε ένα αρχείο txt.

6.1.1 Σκοπός ανάπτυξης εφαρμογής

Σκοπός της ανάπτυξης της συγκεκριμένης εφαρμογής εκτός από το αποτέλεσμα είναι και η κατανόηση του κώδικα, των συναρτήσεων αλλά και των δομών που χρειάζονται για την ανάπτυξη της. Στόχος επίσης είναι να λειτουργήσει με τρόπο και χαρακτήρα εκπαιδευτικό, μιας και το προγραμματιστικό περιβάλλον του Matlab έχει τις δυνατότητες να λειτουργήσει με αυτόν τον τρόπο, δίνοντας εκπαιδευτικές και προγραμματιστικές λύσεις σε πολλούς τομείς οι οποίοι συνδέονται με την πληροφορική και τον προγραμματισμό αλλά και επί της ουσίας με πολλούς κλάδους επιστημών οι οποίοι είναι η βάση όλων των σύγχρονων πρακτικών και εφαρμογών.

6.1.2 Κώδικας εφαρμογής και επεξήγηση

Παρακάτω βρίσκεται ο κώδικας του αρχείου **create_templates.m** και η επεξήγησή του.

```
%CREATE TEMPLATES
%Letter
A=imread('letters_numbers\A.bmp');B=imread('letters_numbe
rs\B.bmp');
C=imread('letters_numbers\C.bmp');D=imread('letters_numbe
rs\D.bmp');
E=imread('letters_numbers\E.bmp');F=imread('letters_numbe
rs\F.bmp');
G=imread('letters_numbers\G.bmp');H=imread('letters_numbe
rs\H.bmp');
I=imread('letters_numbers\I.bmp');J=imread('letters_numbe
rs\J.bmp');
K=imread('letters_numbers\K.bmp');L=imread('letters_numbe
rs\L.bmp');
M=imread('letters_numbers\M.bmp');N=imread('letters_numbe
rs\N.bmp');
O=imread('letters_numbers\O.bmp');P=imread('letters_numbe
rs\P.bmp');
Q=imread('letters_numbers\Q.bmp');R=imread('letters_numbe
rs\R.bmp');
S=imread('letters_numbers\S.bmp');T=imread('letters_numbe
rs\T.bmp');
U=imread('letters_numbers\U.bmp');V=imread('letters_numbe
rs\V.bmp');
W=imread('letters_numbers\W.bmp');X=imread('letters_numbe
rs\X.bmp');
Y=imread('letters_numbers\Y.bmp');Z=imread('letters_numbe
rs\Z.bmp');
a2=imread('letters_numbers\a2.bmp');b2=imread('letters_nu
mbers\b2.bmp');
c2=imread('letters_numbers\c2.bmp');d2=imread('letters_nu
mbers\d2.bmp');
e2=imread('letters_numbers\e2.bmp');f2=imread('letters_nu
mbers\f2.bmp');
g2=imread('letters_numbers\g2.bmp');h2=imread('letters_nu
mbers\h2.bmp');
i2=imread('letters_numbers\i2.bmp');j2=imread('letters_nu
mbers\j2.bmp');
k2=imread('letters_numbers\k2.bmp');l2=imread('letters_nu
mbers\l2.bmp');
```

```
m2=imread('letters_numbers\m2.bmp');n2=imread('letters_nu
mbers\n2.bmp');
o2=imread('letters_numbers\o2.bmp');p2=imread('letters_nu
mbers\p2.bmp');
q2=imread('letters_numbers\q2.bmp');r2=imread('letters_nu
mbers\r2.bmp');
s2=imread('letters_numbers\s2.bmp');t2=imread('letters_nu
mbers\t2.bmp');
u2=imread('letters_numbers\u2.bmp');v2=imread('letters_nu
mbers\v2.bmp');
w2=imread('letters_numbers\w2.bmp');x2=imread('letters_nu
mbers\x2.bmp');
y2=imread('letters_numbers\y2.bmp');z2=imread('letters_nu
mbers\z2.bmp');
dotdot=imread('letters_numbers\dotdot.bmp');dot=imread('l
etters_numbers\dot.bmp');
%Number
one=imread('letters_numbers\1.bmp');
two=imread('letters_numbers\2.bmp');
three=imread('letters_numbers\3.bmp');four=imread('letter
s_numbers\4.bmp');
five=imread('letters_numbers\5.bmp');
six=imread('letters_numbers\6.bmp');
seven=imread('letters_numbers\7.bmp');eight=imread('lette
rs_numbers\8.bmp');
nine=imread('letters_numbers\9.bmp');
zero=imread('letters_numbers\0.bmp');


%*-*-*-*-*-*-*-*-*-*-*-*-*-
letter=[A B C D E F G H I J K L M...
    N O P Q R S T U V W X Y Z];
number=[one two three four five...
    six seven eight nine zero ...
    a2 b2 c2 d2 e2 f2 g2 h2 i2 ...
    j2 k2 l2 m2 n2 o2 p2 q2 r2 ...
    s2 t2 u2 v2 w2 x2 y2 z2 dotdot dot];
space=ones(42,24);
character=[letter number space];
templates=mat2cell(character,42,[24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24]);
save ('templates','templates')
clear all
```

Επεξήγηση κώδικα **create_templates.m** :

Το create_templates έχει τη μορφή script. Το αποτέλεσμά του όπως φαίνεται από την πρότελευταία εντολή , είναι η δημιουργία ενός αρχείου templates.mat το οποίο θα περιέχει το σύνολο των απαραίτητων εικονοστοιχείων (γραμμάτων κεφαλαίων, μικρών, σημείων στήξης).

Το κάθε εικονοστοιχείο διαβάζεται από τον φάκελο letters_numbers (ο οποίος βρίσκεται μαζί με όλα τα άλλα αρχεία στον φάκελο της εφαρμογής OCR) με την εντολή **imread** και αποθηκεύεται μετά σε έναν πίνακα (letter, number).

Πριν καταλήξουμε όμως στην δημιουργία των πινάκων αυτών, πρέπει να πούμε και κάποια πιο συγκεκριμένα πράγματα. Το κάθε ένα εικονοστοιχείο του φακέλου letters_numbers είναι μία εικόνα bitmap (δυαδική) με μέγεθος-διαστάσεις 24x42 (πλάτος x ύψος).

Αυτή η πληροφορία είναι πολύ σημαντική διότι χωρίς δυαδικό τύπο εικόνας, δεν μπορεί να γίνει μετά η αντιστοίχιση των εικονοστοιχείων με τα γράμματα τα οποία «εξάγονται» από την εικόνα μας, και πιο συγκεκριμένα από τη γραμμή της εικόνας μας.

Με βάση αυτήν ακριβώς την λογική, μπορούμε να καταλάβουμε για ποιό λόγο ο ορισμός του κενού (space) , μπορεί να γίνει με μία απλή εντολή του Matlab και χωρίς καν να διαβάσουμε κάποιο εικονοστοιχείο από τον φάκελό μας. Συγκεκριμένα με την εντολή :

```
space=ones(42,24);
```

Δημιουργούμε μία μήτρα ή αλλιώς έναν πίνακα με διαστάσεις 24x42 ο οποίος αποτελείται μόνο από άσσους, που στη προκειμένη περίπτωση ο αριθμός 1 στις δυαδικές εικόνες είναι το άσπρο.

Με τον ίδιο τρόπο δημιούργησα τα εικονοστοιχεία ένα ένα ούτως ώστε να μπορεί να γίνει αργότερα ταυτοποίησή τους μέσω του read_letter.

Ακόμα πιο συγκεκριμένα, κάποια γράμματα-εικονοστοιχεία χρειάστηκε να τα εξάγω από εικόνες για να τα δημιουργήσω ούτως ώστε να εμπλουτιστεί ακόμα περισσότερο η εφαρμογή. Σε αυτή την περίπτωση χρησιμοποιώντας την ίδια την εφαρμογή και λειτουργώντας με break points , έβρισκα την τοπεθεσία του γράμματος που ήθελα στη γραμμή που βρισκόταν, και με την εντολή imwrite έγραφα το εικονοστοιχείο με τη μορφή bmp.

Παρακάτω βρίσκεται ο κώδικας του αρχείου **lines.m** και η επεξήγησή του.

```
function [fl re]=lines(im_texto)

% χωριζει το κειμενο της εικονας σε γραμμες
% im_texto->εικονα , fl->first line , re->remain line
% [fl re]=lines(im_texto);
% subplot(3,1,1);imshow(im_texto);
% subplot(3,1,2);imshow(fl);
% subplot(3,1,3);imshow(re);
im_texto=clip(im_texto);
num_filas=size(im_texto,1);
for s=1:num_filas
    if sum(im_texto(s,:))==0
        nm=im_texto(1:s-1, :); % πινακας πρωτης γραμμης
        rm=im_texto(s:end, :);%πινακας γραμμων που μενουν
        fl = clip(nm);
        re=clip(rm);
        %          subplot(2,1,1);imshow(fl);
        %          subplot(2,1,2);imshow(re);
        break
    else
        fl=im_texto;%για οταν εχω μονο μια γραμμη
        re=[ ];
    end
end

function img_out=clip(img_in)
[f c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));%κοβει την
εικονα
```

Επεξήγηση κώδικα **lines.m** :

Το lines.m είναι ουσιαστικά μία συνάρτηση που δημιουργούμε για να χωρίζουμε την εικόνα που εμπεριέχει χαρακτήρες (κείμενο) σε γραμμές κειμένου. Με τον τρόπο αυτό όπως φαίνεται και στο εκτελέσιμο αρχείο, κάνουμε ακόμα ευκολότερη την επεξεργασία κάθε στοιχείου ξεχωριστά μετρώντας τις γραμμές και αντίστοιχα τα στοιχεία κάθε γραμμής.

Δημιουργούμε έτσι :

```
for s=1:num_filas
    if sum(im_texto(s,:))==0
        nm=im_texto(1:s-1, :); % πινακας πρωτης γραμμης
        rm=im_texto(s:end, :);%πινακας γραμμων που μενουν
        fl = clip(nm);
        re=clip(rm);
        %          subplot(2,1,1);imshow(fl);
        %          subplot(2,1,2);imshow(re);
        break
    else
        fl=im_texto;%για οταν εχω μονο μια γραμμη
        re=[ ];
    end
end
```

Με τον παραπάνω κώδικα χωρίζεται η εικόνα σε γραμμές οι οποίες με την συνάρτηση clip που δημιουργείται παρακάτω κόβονται :

```
function img_out=clip(img_in)
[f c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));
```

Έτσι στον πίνακα [fl re] έχουμε σαν ορίσματα την πρώτη γραμμή (fl) και τις εναπομείναντες (re).

Παρακάτω βρίσκεται ο κώδικας του αρχείου **read_letter.m** και η επεξήγησή του.

```
function letter=read_letter(imagn,num_letras)
% υπολογιζει τον συσχετισμο(corellation) μεταξυ της
εικονας και των
% εικονοστοιχειων που βρισκονται στο templates.mat
% το αποτελεσμα του ειναι ενα string που περιεχει το
γραμμα
% το μεγεθος της εικονας(imagn) πρεπει να ειναι 42 x 24
pixels

global templates
comp=[ ];
global counter

%----
a=ones(42,24);
b=sum(sum(imagn-a));
if b==0
    %imagn(1,:)=0;
    counter=counter+1;
else
    counter=0;
end


%----
global n
for n=1:num_letras
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
vd=find(comp==max(comp));
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*


if b==0 && counter==2
    vd=45;
end



global c
global cm

cm=max(c);
```

```
if vd==1
    letter='A';
elseif vd==2
    letter='B';
elseif vd==3
    letter='C';
elseif vd==4
    letter='D';
elseif vd==5
    letter='E';
elseif vd==6
    letter='F';
elseif vd==7
    letter='G';
elseif vd==8
    letter='H';
elseif vd==9
    letter='I';
elseif vd==10
    letter='J';
elseif vd==11
    letter='K';
elseif vd==12
    letter='L';
elseif vd==13
    letter='M';
elseif vd==14
    letter='N';
elseif vd==15
    letter='O';
elseif vd==16
    letter='P';
elseif vd==17
    letter='Q';
elseif vd==18
    letter='R';
elseif vd==19
    letter='S';
elseif vd==20
    letter='T';
elseif vd==21
    letter='U';
elseif vd==22
    letter='V';
elseif vd==23
    letter='W';
elseif vd==24
    letter='X';
elseif vd==25
    letter='Y';
elseif vd==26
```

```
    letter='Z';


    %*-*-*-*-*
elseif vd==27
    letter='1';
elseif vd==28
    letter='2';
elseif vd==29
    letter='3';
elseif vd==30
    letter='4';
elseif vd==31
    letter='5';
elseif vd==32
    letter='6';
elseif vd==33
    letter='7';
elseif vd==34
    letter='8';
elseif vd==35
    letter='9';
elseif vd==36
    letter='0';
%---
elseif vd==37
    letter='a';
elseif vd==38
    letter='b';
elseif vd==39
    letter='c';
elseif vd==40
    letter='d';
elseif vd==41
    letter='e';
elseif vd==42
    letter='f';
elseif vd==43
    letter='g';
elseif vd==44
    letter='h';

elseif vd==45
    letter='i';
elseif vd==46
    letter='j';

elseif vd==47
    letter='k';
elseif vd==48
```

```
    letter='l';
elseif vd==49
    letter='m';
elseif vd==50
    letter='n';
elseif vd==51
    letter='o';
elseif vd==52
    letter='p';
elseif vd==53
    letter='q';
elseif vd==54
    letter='r';
elseif vd==55
    letter='s';
elseif vd==56
    letter='t';
elseif vd==57
    letter='u';
elseif vd==58
    letter='v';
elseif vd==59
    letter='w';
elseif vd==60
    letter='x';
elseif vd==61
    letter='y';
elseif vd==62
    letter='z';
elseif vd==63
    letter=':';
elseif vd==64
    letter='.';

else letter='';



end
```

Επεξήγηση κώδικα **read_letter.m** :

Πριν πούμε για τον υπολογισμό των συσχετισμών, στην αρχή του κώδικα του m-file read_letter ορίζουμε σαν global εκτός από το templates και έναν counter. Παρακάτω θα τον χρησιμοποιήσουμε για τον υπολογισμό του γράμματος i , μιας και το συγκεκριμένο γράμμα έχει κάποιες ιδιαιτερότητες. Πρώτον με βάση τον αλγοριθμό μας το i αποτελείται από δύο στοιχεία, την τελεία που έχει και το υπόλοιπό του και αυτό γιατί το γράμμα δεν είναι ενιαίο. Και στην περίπτωση της τελείας όμως και στην περίπτωση του «κορμού» του γράμματος, θα πάρουμε έναν πίνακα γεμάτο άσσους και μόνο και είναι το μόνο γράμμα (μιας και δεν έχουμε σημεία στήξης) στο οποίο γίνεται αυτό δύο συνεχόμενες φορές. Οπότε :

```
global counter

%----
a=ones(42,24);
b=sum(sum(imagn-a));
if b==0
    %imagn(1,:)=0;
    counter=counter+1;
else
    counter=0;
end
```

Και πιο κάτω:

```
if b==0 && counter==2
    vd=45;
end
```

Αν ισχύει αυτό τότε να γυρίσει πίσω το γράμμα i (όπου vd=45 , ή αλλιώς 45° εικονοστοιχείο στο templates.mat).

Η function read_letter παίρνει σαν ορίσματα , πρώτα την εικόνα imagn η οποία είναι το εξαγώμενο γράμμα από την εκάστοτε γραμμή της εικόνας, και το num_letras το οποίο είναι ο αριθμός των γραμμάτων που περιέχει η γραμμή.

Αρχικά ορίζουμε σαν global την templates, και μετά έχουμε :

```
for n=1:num_letras
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
```

Εδώ γίνεται μία επανάληψη ώστε να συσχετιστούν όλα τα γράμματα της κάθε γραμμής. Στο sem αποθηκεύεται το αποτέλεσμα του συσχετισμού (όπως εξηγήσαμε τη λειτουργία της corr2 σε προηγούμενο κεφάλαιο) μεταξύ του στοιχείου που μόλις έχουμε εξάγει (imagn) με βάση την γραμμή στην οποία βρισκόμαστε και -όλων- των εικονοστοιχείων που έχουμε αποθηκευμένα στο templates.mat.

Δημιουργούμε έναν πίνακα comp μέσα στον οποίο θα αποθηκευτεί και το sem και ουσιαστικά θα εμπεριέχει όλους τους συσχετισμούς οι οποίοι συντελέστηκαν μέσα στην επανάληψη (δηλαδή για όλα τα εικονοστοιχεία που εξάχθηκαν από την εκάστοτε γραμμή) μεταξύ όλων των αποθηκευμένων εικονοστοιχείων που περιέχει το templates.mat.

Η επόμενη εντολή :

```
vd=find(comp==max(comp));
```

Βρίσκει και αποθηκεύει στο vd τον αριθμό με τον μεγαλύτερο συσχετισμό που βρίσκει στον πίνακα comp. Μετά με βάση αυτόν τον αριθμό δίνει και το αντίστοιχο γράμμα ή αριθμό.

Έπειτα, οι δηλώσεις ως global κάποιων μεταβλητών και συγκεκριμένα των :

```
global c
global cm
```

```
n1=imgn(min(r):max(r),min(c):max(c));
```

Η συγκεκριμένη γραμμή κώδικα βρίσκεται στο εκτελέσιμο αρχείο OCR.m. Κάθε εξαγόμενο γράμμα έχει δύο πίνακες που το συνθέτουν, τον πίνακα r και τον πίνακα c. Ο πίνακας r έχει ίδια περιεχόμενα για κάθε ξεχωριστό εικονοστοιχείο αρκεί να είναι το ίδιο εικονοστοιχείο (πχ ίδιο για κάθε γράμμα i).

Τέλος ο πίνακας c περιέχει τιμές οι οποίες διαμορφώνονται κάθε φορά με βάση το σημείο της γραμμής στο οποίο βρίσκεται ο αλγόριθμος. Ξεκινάει δηλαδή από το 0 και αυξάνεται συνεχώς με βάση το μήκος της γραμμής που κόπηκε από την function lines.m, φτάνοντας μέχρι το τελευταίο pixel της.

Άρα έτσι μπορεί να υπολογιστεί το πλάτος κάθε γράμματος αλλά και η απόσταση. Γι' αυτόν ακριβώς τον λόγο η εύρεση των κενών γίνεται με αυτόν τον τρόπο, και αφού έχουμε ορίσει ως global τις μεταβλητές cm και c. Η πρώτη ορίζεται κάθε φορά που εισέρθει η εφαρμογή στο read_letter με την εντολή :

```
cm=max(c);
```

Αφού έχει αρχικοποιηθεί βέβαια ως cm=0 και global στο OCR.m. Η μεταβλητή c ορίζεται στο OCR.m :

```
cc=min(c);
```

Ορίζεται μέσα στην συνθήκη for στην οποία γίνεται η εξαγωγή του εικονοστοιχείου από τη γραμμή, και ακολουθεί η συνθήκη :

```
if (cc-cm)>=12
            letter=read_letter(img_r,num_letras);
            word=[word ' ' letter];

        else
        letter=read_letter(img_r,num_letras);
        % Letter concatenation
        word=[word letter];
        end
```

Η τιμή 12 είναι μετά από εύρεση η καταλληλότερη δήλωση ώς απόσταση «κατώφλι» η οποία ορίζει ένα κενό.

Για να γίνεται κατανοητή η λειτουργία:

Το cm το οποίο αρχικοποιήσαμε ως 0 , ορίζεται κάθε φορά από το μέγιστο (max(c)) του εκάστοτε εικονοστοιχείου (ορίζεται στο read_letter), με λίγα λόγια είναι το δεξιότερο σημείο του κάθε εικονοστοιχείου.

Ας υποθέσουμε ότι είμαστε στο πρώτο εικονοστοιχείο της πρώτης γραμμής. Το cc που θα οριστεί στο OCR.m μέσα στην συνθήκη for που γίνεται εξαγωγή του εικονοστοιχείου, είναι το ελάχιστο του πίνακα c (min(c)) , ο οποίος όπως είπαμε είναι επί της ουσίας δείκτης του μήκους της εκάστοτε γραμμής. Με την αφαίρεση λοιπόν του δεξιότερου σημείου του πρώτου εικονοστοιχείου από το αρίστερότερο σημείο του δεύτερου εικονοστοιχείου, βρίσκουμε το μέγεθος του κενού!

Όταν υπερβαίνει τον αριθμό 12, συνήθως (δυστυχώς όχι πάντα) σημαίνει πως έχουμε κενό. Διαφορετικά δεν έχουμε.

Παρακάτω βρίσκεται ο κώδικας του εκτελέσιμου αρχείου **OCR.m** και η επεξήγησή του.

```
% O.C.R (Optical Character Recognition - Οπτικη
Αναγνωριση Χαρακτηρων)
warning off %#ok<WNOFF>
% clear
clc, close all, clear all
create_templates;
% διαβαζω την εικονα
imagen=imread('test2.jpg');
% δειχνω την εικονα
imshow(imagen);
title('Εικόνα προς επεξεργασία')
% την μετατρεπω σε εικονα φωτεινοτητας(grayscale)
if size(imagen,3)==3 %RGB image
    imagen=rgb2gray(imagen);
end

% τη μετατρεπω σε εικονα bw
threshold = graythresh(imagen);
imagen =~im2bw(imagen,threshold);
% αφαιρω ολα τα αντικειμενα που ειναι μικροτερα απο
30pixel, αν χρειαστει
%imagen = bwareaopen(imagen,30);
% αποθηκευω την μητρα της λεξης-word απτην εικονα
word=[ ];
re=imagen;
% ανοιγω το text.txt ως αρχειο για εγγραφη
fid = fopen('text.txt', 'wt');
% φορτωνω το templates
load templates
global templates
global cm
cm=0;
global counter
counter=0;
% υπολογιζω τον αριθμο των γραμματων στο αρχειο templates
num_letras=size(templates,2);
while 1
    % διαχωριζω τις γραμμες-lines στο κειμενο της εικονας
    [fl re]=lines(re);
    imgn=fl;
    % αν βγαλω απο σχολιο την απο κατω γραμμη θα δω τις
γραμμες μια μια
    %imshow(fl);pause(0.5)
    %-----------------------------------------------
    % Label and count connected components
    [L Ne] = bwlabel(imgn);
    global n
    for n=1:Ne
```

```
        global c
        [r,c] = find(L==n);
        % εξαγωγη γραμματος
        n1=imgn(min(r):max(r),min(c):max(c));

        num=numel(n1);
        if n1(:)==1
            if num>123
            word=[word '1'];
            end
        end


        % κανω resize το γραμμα, να ειναι 42x24 οπως τα
εικονοστοιχεια μου
        img_r=imresize(n1,[42 24]);

        cc=min(c);
        % αν βγαλω απο σχολιο την απο κατω γραμμη θα δω
τα γραμματα ενα ενα
         %imshow(img_r);pause(0.5)
        %-----------------------------------------------
------------------
        % Call fcn to convert image to text
        % υπολογιζω που υπαρχει κενο
        if (cc-cm)>=12
            letter=read_letter(img_r,num_letras);
            word=[word ' ' letter];

        else
        letter=read_letter(img_r,num_letras);
        % Letter concatenation
        word=[word letter];
        end
    end
    %fprintf(fid,'%s\n',lower(word));%Write 'word' in
text file (lower)
    fprintf(fid,'%s\n',word);%Write 'word' in text file
(upper)
    % Clear 'word' variable
    word=[ ];
    %*When the sentences finish, breaks the loop
    if isempty(re)  %See variable 're' in Fcn 'lines'
        break
    end
end
fclose(fid);
% και τελος ανοιγω το αρχειο text.txt
winopen('text.txt')
clear all
```

Επεξήγηση κώδικα **OCR.m** :

Το OCR.m είναι το εκτελέσιμο αρχείο της εφαρμογής αυτής στο Matlab. Αρχικά διαβάζουμε και προβάλουμε την εικόνα που πρόκειται να επεξεργαστούμε. Ύστερα μετατρέπουμε την RGB εικόνα σε grayscale (εικόνα φωτεινότητας όπως έχουμε προσδιορίσει και σε προηγούμενο κεφάλαιο) και ακριβώς μετά την εικόνα φωτεινότητας την μετατρέπουμε σε μαυρόασπρη ή αλλιώς bw με την εντολή :

```
imagen =~im2bw(imagen,threshold);
```

Και αυτό αφού έχουμε ορίσει πριν την τιμή κατωφλίου (threshold).

Φτιάχνουμε έναν πίνακα word[] που μετά θα αποθηκευτεί η λέξη και ανοίγουμε το αρχείο txt στο οποίο θα εγγραφεί το αποτέλεσμα της οπτικής αναγνώρισης χαρακτήρων με την εντολή :

```
fid = fopen('text.txt', 'wt');
```

Μετά φορτώνουμε το αρχείο templates.mat στο οποίο όπως είπαμε αποθηκεύσαμε σε δυαδική μορφή και σε μορφή πινάκων όλη την βάση των εικονοστοιχείων μας και το ορίζουμε ως global ώστε να «φαίνεται» σε όλα τα διαφορετικά αρχεία που δημιουργήσαμε και χρειάζεται (όπως στο read_letter). Ακόμα ώς global ορίζεται και η τιμή cm η οποία όπως αναφέραμε εξηγώντας τον κώδικα για το read_letter.m έχει χρησιμότητα για τον προσδιορισμό των κενών διαστημάτων.

Στη συνθήκη while που ακολουθεί «διαβάζουμε» κάθε γραμμή που έχει κοπεί από το lines.m. Μετράμε επίσης τα στοιχεία της γραμμής τα οποία μετά στην επόμενη συνθήκη του if αντιπροσωπεύουν την μεταβλητή Ne.

Το κάθε εικονοστοιχείο (γράμμα,αριθμός) διαμορφώνεται με τον πίνακα [r,c] και εξάγεται στο n1 με την εντολή :

```
n1=imgn(min(r):max(r),min(c):max(c));
```

Πριν γίνει η προσαρμογή του μεγέθους (resize) του εξαγώμενου στοιχείου, έχουμε ένα κομμάτι κώδικα το οποίο εντοπίζει τις περιπτώσεις που έχουμε είτε το γράμμα l (L μικρό), είτε το γράμμα I (i κεφαλαίο). Είναι εύκολα αντιληπτό ότι στις περισσότερες γραμματοσειρές τα γράμματα αυτά είναι ακριβώς ίδια! Στην προκειμένη περίπτωση δεν είναι αυτό το μόνο πρόβλημα, καθώς θα πρέπει αρχικά να απεικονιστούν. Αυτό θα γίνεται κάθε φορά που βρίσκεται και εξάγεται ένα στοιχείο από την εικόνα (και εκάστοτε γραμμή) του οποίου ο πίνακας είναι γεμάτος άσσους (πράγμα που σημαίνει ότι είναι μία «μαύρη» γραμμή) και έχει μέγεθος μεγαλύτερο από μία τιμή (η οποία δυστυχώς είναι μεταβλητή με βάση διαφορετικά μεγέθη γραμμάτων). Αυτό

γίνεται μιας και το ίδιο ισχύει και για το γράμμα i (αφού εντοπίζεται σαν δύο ξεχωριστά εικονοστοιχεία), όμως το γράμμα l που θέλουμε θα έχει μεγαλύτερο πίνακα, λόγω μεγαλύτερου μεγέθους :

```
num=numel(n1);
if n1(:)==1
   if num>123
   word=[word 'l'];
   end
end
```

Ακριβώς μετά γίνεται η προσαρμογή του μεγέθους του εξαγώμενου στοιχείου με την εντολή :

```
img_r=imresize(n1,[42 24]);
```

Και ακολουθούν εντολές που εξηγήθηκαν στη προηγούμενη ενότητα τι κάνουν, μέχρι να κλείσουν και οι δύο συνθήκες (πρώτα το if και μετά το while).

Τέλος με την εντολή :

```
fprintf(fid,'%s\n',word);
```

Γράφει την λέξη η οποία προφανώς ειναι τύπου string , στο αρχείο κειμένου txt αδείαζουμε τον πίνακα word για την επόμενη λέξη που θα ακολουθήσει , εάν ακολουθήσει πράγμα που εξετάζεται εδώ :

```
if isempty(re)
        break
end
```

Με την fclose κλείνουμε το αρχείο ώστε να μπορεί να διαβαστεί και μετά το ανοίγουμε ακριβώς στο τέλος της εφαρμογής για να δούμε τα αποτελέσματα της οπτικής αναγνώρισης της όποιας εικόνας χρησιμοποιήσαμε για είσοδο.

6.1.3 Συμπεράσματα

Τα συμπεράσματα που βγαίνουν από την ολοκλήρωση της πρώτης εφαρμογής στο Matlab είναι πολλά. Ένα από αυτά είναι ότι για τη δημιουργία μίας εφαρμογής οπτικής αναγνώρισης κειμένου χρειαζόμαστε οπωσδήποτε βασικές γνώσεις προγραμματισμού αλλά και επεξεργασίας εικόνας. Πέρα από τις γνώσεις όμως συμπαιρένουμε ακόμα ότι η οπτική αναγνώριση κειμένου είναι ένας πολύπλοκος τομέας ο οποίος για να έχει τα μεγαλύτερα δυνατά αποτελέσματα χρειάζεται συνεχή ανανέωση..

Αυτό τι σημαίνει; Σημαίνει ότι μία παγιωμένη βάση δεδομένων η οποία εμπεριέχει έναν αριθμό εικονοστοιχείων θα έχει αποτέλεσμα μόνο σε συγκεκριμένο τύπο εικόνων ή καλύτερα σε συγκεκριμένο τύπο κειμένου αποτυπωμένου σε ψηφιακές εικόνες. Εάν αναλογιστούμε τον τεράστιο αριθμό διαφορετικών γραμματοσειρών καταλαβαίνουμε πόσο αυξάνεται η πολυπλοκότητα στο να εξάγουμε χρήσιμο και σωστό αποτέλεσμα σε εικόνες με εντελώς διαφορετικούς τύπους κειμένου.

Από την άλλη όμως η τεχνική αναγνώρισης ενός εικονοστοιχείου και η ταυτισή του με κάποιο υπάρχον δεν θα πέσει ποτέ έξω από τη στιγμή που η βάση μας είναι επαρκής. Επίσης όσο η ψηφιακή εικόνα που επεξεργαζόμαστε βρίσκεται σε καλή κατάσταση και δεν είναι φωτογραφία (με θόρυβο, ανάκλαση και θολούρα) τόσο τα αποτελέσματα πρόκειται να είναι σωστά.

Παρακάτω αναλύονται περισσότερες σκέψεις για τα θετικά και τα αρνητικά μίας τέτοιας εφαρμογής.

6.1.4 Θετικά – Αρνητικά

Στα θετικά της 1^{ης} αυτής εφαρμογής είναι η ευκολία χρησιμοποίησης των συναρτήσεων επεξεργασίας εικόνας μέσα στον κώδικα, και τα αποτελέσματά τους τα οποία διακρίνονται και οδηγούν στο τελικό αρχείο κειμένου. Έπειτα η ταχύτητα ολοκλήρωσης είναι αρκετά καλή για εικόνες που περιέχουν μία λογική ποσότητα κειμένου.

Η εξαγωγή αρχείου κειμένου επίσης είναι πολύ σημαντική , μιας και αργότερα μπορεί να χρησιμοποιηθεί πολύ εύκολα για περισσότερη επεξεργασία του κειμένου που προέκυψε αλλα ακόμα και σύγκριση των περιεχομένων του αρχείου αυτού με κάποιο άλλο.

Θετικό επιπλέον είναι το γεγονός ότι μπορούν να προστεθούν περισσότερα εικονοστοιχεία πάρα πού εύκολα αλλά και κώδικας για ακόμα μεγαλύτερη επεξεργασία εικόνας για οπτική αναγνώριση σε δυσκολότερες εικόνες.

Στα αρνητικά είναι το γεγονός ότι η «βάση» ή φάκελος που περιέχονται τα εικονοστοιχεία θα πρέπει να είναι πολύ μεγαλύτερη και να ανανεώνεται για ευκολότερη ταύτιση.

Επίσης η μέθοδος της αντιστοίχησης με πρότυπα (μία εκ των μεθόδων OCR όπως αναφέρεται και σε προηγούμενο κεφάλαιο) δεν είναι η ταχύτερη και δεν θα μπορεί να εξάγει αποτελέσματα σε χαρακτήρες οι οποίοι ξεφεύγουν αρκετά από αυτά τα πρότυπα. Άρα θα είναι αδύνατη μία οπτική αναγνώριση κειμένου σε εικόνα με γραφικό χαρακτήρα. Αντίθετα με τη μέθοδο εξαγωγής χαρακτηριστικών ή με κάποια υβριδική αναγνώριση τα αποτελέσματα σίγουρα θα είναι υπαρκτά και κοντά σε αυτό που επιθυμείται.

Εν τέλη όμως η μελέτη αυτής της μεθόδου, είναι πολύ σημαντική στην αρχική κατανόηση της έννοιας της οπτικής αναγνώρισης και βοηθάει στην διαμόρφωση νέων ιδεών και μεθόδων για ακόμα καλύτερα αποτελέσματα, με στόχο έναν «ιδανικό» αλγόριθμο ο οποίος θα λειτουργεί σαν μεταφραστής ακόμα και για τις δυσκολότερες γραμματοσειρές.

6.2 Εφαρμογή O.C.R στην Javascript

Η δεύτερη εφαρμογή οπτικής αναγνώρισης χαρακτήρων έχει αρκετά διαφορετική μορφή από την προηγούμενη αλλά και διαφορετικό χαρακτήρα. Συμβαδίζει πολύ περισσότερο με τις απαιτήσεις της σημερινής εποχής μιας και η συμβατότητά της είναι πολύ μεγαλύτερη σε σχέση με την πρώτη εφαρμογή όπου είναι απαραίτητη η κατοχή του προγραμματιστικού περιβάλλοντος Matlab. Αντίθετα στην εφαρμογή αυτή που βασίζεται στην γλώσσα προγραμματισμού Javascript, το μόνο που είναι απαραίτητο είναι ένας εγκατεστημένος browser (Mozilla Firefox, Google Chrome κλπ).

Για το τελικό αποτέλεσμα χρησιμοποιήσαμε τις βιβλιοθήκες javascript : jquery αλλά και ocrad, όπου η πρώτη χρειάζεται για την εφαρμογή του jcrop στην εικόνα και η δεύτερη για την οπτική αναγνώριση κειμένου στην τελική ψηφιακή εικόνα μας.

6.2.1 Σκοπός ανάπτυξης εφαρμογής

Σκοπός ανάπτυξης της δεύτερης αυτής εφαρμογής είναι η ακόμα μεγαλύτερη ευκολία και αμεσότητα ενός προγράμματος στην οπτική αναγνώριση κειμένου. Η γλώσσα Javascript όπως γνωρίζουμε έχει απίστευτη συμβατότητα με όλες τις σύγχρονες εκδόσεις των web browsers και αυτό την κάνει απαραίτητη στη δημιουργία νέων εφαρμογών οι οποίες θα μπορούν εύκολα να χρησιμοποιηθούν αλλά και να μεταδοθούν μέσω του internet σε μεγάλο αριθμό χρηστών, ακόμα και να αποκτήσουν εμπορική αξία ανάλογα πάντα με την επιτυχία τους.

Μεγαλύτερος στόχος με λίγα λόγια στην δεύτερη αυτή εφαρμογή είναι η ευκολία χρήσης και διάδοσης της εφαρμογής αυτής με βάση τα σημερινά δεδομένα στο χώρο του διαδικτύου και λιγότερο η εμβάθυνση στον τρόπο οπτικής αναγνώρισης σε σχέση κιόλας με την πρώτη εφαρμογή η οποία έδινε μεγαλύτερο βάρος σε αυτό.

Τέλος ως σκοπός μπορεί να προστεθεί και η κατανόηση του τρόπου ανάπτυξης μίας εφαρμογής στη γλώσσα Javascript , των παραμέτρων που πρέπει να ληφθούν υπόψην και των προβλημάτων που πρέπει να ξεπεραστούν ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα.

6.2.2 Κώδικας εφαρμογής και επεξήγηση

Αρχικά ο κώδικας «εμφάνισης» ο οποίος βρίσκεται μέσα στον φάκελο css με όνομα **styles.css** :

```
<style type="text/css">


html { height: 100%; }


body {
  margin: 0 auto;
  padding: 0;
  background-color: #eeeeee;
}


button {
  font-family:Arial;
  font-size:16px;
  font-weight:bold;
  display:inline-block;
  width:80px;
  height:40px;
  padding:5px;
  border:2px #79bbff; solid;
```

```
    text-align:center;
    text-decoration:none;
    background-color:#3d94f6;
    color:#ffffff;
}
button:hover {cursor: hand; cursor: pointer;}


textarea {
      border: 2px solid #3366FF;
      border-radius: 10px;

}</style>
```

Επεξήγηση κώδικα **styles.css** :

Η γλώσσα css ανήκει στην κατηγορία γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μία γλώσσα σήμανσης όπως η html ή xhtml. Η CSS είναι μια γλώσσα υπολογιστή προορισμένη να αναπτύσσει στυλιστικά μια ιστοσελίδα δηλαδή να διαμορφώνει περισσότερα χαρακτηριστικά, χρώμματα, στοίχιση και δίνει περισσότερες δυνατότητες σε σχέση με την html. Για μια όμορφη και καλοσχεδιασμένη ιστοσελίδα η χρήση της CSS κρίνεται ως απαραίτητη.

Ακριβώς για αυτόν τον λόγο τη χρησιμοποιούμε και στην εφαρμογή αυτή που βασίζεται στην html, ώστε το εμφανισιακό και στυλιστικό αποτέλεσμα να είναι όσο καλύτερο γίνεται και παράλληλα να είναι όσο πιο ευδιάκριτο και λιτό γίνεται.

Κώδικας εκτελέσιμου αρχείου **jtest.html** :

```
<!DOCTYPE html>

<html lang="en">

<html>



<head>

<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/j
query.min.js"></script>

<script src="js/jquery.min.js"></script>

<script src="js/jquery.Jcrop.js"></script>

<script src="js/jquery.Jcrop.min.js"></script>

<script src="js/ocrad.min.js"></script>


<link rel="stylesheet" href="css/jquery.Jcrop.css"
type="text/css" />

<link rel="stylesheet" href="css/jquery.Jcrop.min.css"
type="text/css" />


<link rel="stylesheet" href="css/styles.css"
type="text/css"/>



</head>
```

```
<body>


<!--

     Ideally these elements aren't created until it's
confirmed that the

     client supports video/camera, but for the sake of
illustrating the

     elements involved, they are created with markup (not
JavaScript)

-->


<div align="center"><video id="video" width="640"
height="480" autoplay></video></div>

<div align="center" ><button id="snap"
>Snap</button></div>

<br><div align="center" ><canvas id="myCanvas"
width="640" height="480" style="background-
color:#eeeeee;" ></canvas></div>

<!-- το υψος του myCanvas ειναι 480 λογω της αναλυσης της
καμερας που ειναι τετοια συνηθως -->

<br><div align="center"><canvas id="cropped"
></canvas></div>

<br><div align="center"><textarea id="myText" rows="4"
cols="50"></textarea></div>

<canvas id="canvas" ></canvas>
```

```
<script type="text/javascript">
// Put event listeners into place
window.addEventListener("DOMContentLoaded", function() {
     // Grab elements, create settings, etc.
     var canvas = document.getElementById("canvas"),
          context = canvas.getContext("2d"),
          video = document.getElementById("video"),
          videoObj = { "video": true },
          errBack = function(error) {
               console.log("Video capture error: ",
error.code);
          };

     // Put video listeners into place
     if(navigator.getUserMedia) { // Standard
          navigator.getUserMedia(videoObj,
function(stream) {
               video.src = stream;
               video.play();
          }, errBack);
     } else if(navigator.webkitGetUserMedia) { // WebKit-
prefixed
          navigator.webkitGetUserMedia(videoObj,
function(stream){
               video.src =
window.webkitURL.createObjectURL(stream);
               video.play();
          }, errBack);
     }
```

```
    else if(navigator.mozGetUserMedia) { // Firefox-
prefixed

        navigator.mozGetUserMedia(videoObj,
function(stream){

            video.src =
window.URL.createObjectURL(stream);

            video.play();

        }, errBack);

    }



}, false);



var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

// Trigger photo take

document.getElementById("snap").addEventListener("click",
function() {

    context.drawImage(video, 0, 0 ) ;

});



//jcrop

$(function(){ $('#myCanvas').Jcrop({ aspectRatio: 0,
onSelect: updateCoords }); });

function updateCoords(c) { $('#x').val(c.x);
$('#y').val(c.y);  $('#w').val(c.w); $('#h').val(c.h);
myF(c);};
```

```
function checkCoords() { if (parseInt($('#w').val()))
return true; alert('Select to crop.');

return false; };

//αν θελω να δω τις συντεταγμενες βαζω μεσα στην function
updateCoords(c)

//πριν την myF(c); το window.alert
(c.x+','+c.y+','+c.w+','+c.h);

//ουσιαστικα με βαση αυτες τις συντεταγμενες του jcrop θα
γινει το κοψιμο της εικονας μετα


function myF(c){

$('#x').val(c.x); $('#y').val(c.y); $('#w').val(c.w);
$('#h').val(c.h);

var canvass = document.getElementById('cropped');

canvass.width = c.w;

canvass.height = c.h;

var ctx = canvass.getContext('2d');

ctx.drawImage(myCanvas,c.x,c.y,c.w,c.h,0,0,c.w,c.h);


//OCRAD

var resultText = OCRAD(ctx);

//resultText = resultText.trim();

//textarea

var text = document.getElementById("myText");

text.innerHTML = resultText;

};

</script>

</body>

</html>
```

Επεξήγηση κώδικα αρχείου **jtest.html** :

Το jtest είναι το εκτελέσιμο αρχείο μας, γι'αυτό και στην αρχή αρχή του και συγκεκριμένα μέσα στις ετικέτες <head> ορίζουμε τα scripts τα οποία λειτουργούν ως βιβλιοθήκες για συγκεκριμένες ενέργειες που μετέπειτα θα χρειαστεί να γίνουν.

Τα jquery θα μας χρειαστούν για την λειτουργία jcrop και το script ocrad.js για τη λειτουργία της οπτικής αναγνώρισης κειμένου.

Επίσης ακριβώς μετα τα scripts δηλώνονται ως links τα αρχεία css που θα χρειαστούν να γίνουν import , όπως το css που δείξαμε ακριβώς πριν.

Ακριβώς μετά ξεκινάει το <body> και στην αρχή αυτού έχουμε ορίσει τα βασικά elements που θα χρησιμοποιήσουμε στην εφαρμογή :

```
<div align="center"><video id="video" width="640"
height="480" autoplay></video></div>

<div align="center" ><button id="snap"
>Snap</button></div>

<br><div align="center" ><canvas id="myCanvas"
width="640" height="480" style="background-
color:#eeeeee;" ></canvas></div>

<br><div align="center"><canvas id="cropped"
></canvas></div>

<br><div align="center"><textarea id="myText" rows="4"
cols="50"></textarea></div>

<canvas id="canvas" ></canvas>
```

Όπως βλέπουμε ορίσαμε ένα video element, ένα button (μονο ένα μας χρειάζεται) , τρία canvas elements και ένα textarea.

Ποία είναι η ιδέα των canvas :

Τα canvas είναι ουσιαστικά ο χώρος στη σελίδα στον οποίο θα βρίσκονται τα elements που θέλουμε, όπως το video stream μέσω της κάμερας του υπολογιστή.

Το πρώτο κατα σειρά απεικόνισης canvas με id=canvas δεν έχει καθορισμένο μέγεθος και θα μας χρησιμεύσει ακριβώς μετά στην τοποθέτηση του video element το οποίο και αυτό ορίζεται με id=video. Το ορισμένο μέγεθος πλάτους και ύψους δεν παίζει ρόλο μιας και θα καθοριστεί από την ανάλυση της

εκάστοτε κάμερας (laptop,pc κλπ) και ανάλογο λοιπόν θα είναι και το μέγεθος του πρώτου κατά σειρά καμβά.

Πλέον είμαστε έτοιμοι να γράψουμε ουσιαστικά Javascript (και οχι html) ξεκινώντας με αυτό τον τρόπο :

```
<script type="text/javascript">
```

Παρακάτω θα δημιουργήσουμε ακροατές γεγονότων ή καλύτερα event listeners οι οποίοι περιέχουν μεθόδους που εκτελούνται κάθε φορά που εμφανίζεται το γεγονός που εκπροσωπούν.

Στη προκειμένη περίπτωση το γεγονός αυτό είναι το getUserMedia με το οποίο «ανιχνεύεται» η κάμερα του navigator ή αλλίως του περιηγητή (firefox, google chrome κλπ) που βασισμένοι σε αυτήν θα βγάλουμε την φωτογραφία κειμένου οπου θέλουμε να κάνουμε οπτική αναγνώριση.

Το getUserMedia είναι ένα API (application programming interface) το οποίο ξεκίνησε ως μέρος της HTML5. Επιτρέπει στην Javascript την πρόσβαση στη κάμερα και το μικρόφωνο μίας συσκευής.

Υπάρχουν περιπτώσεις-συνθήκες με βάση το ποιόν περιηγητή έχουμε διότι υπάρχει διαφορετικότητα στο getUserMedia ανάλογα με τον περιηγητή, ώστε να δημιουργήσουμε το αντικείμενο του video και να παίξει ακριβώς μετά :

```
video.src = window.URL.createObjectURL(stream);

video.play();
```

Στις επόμενες εντολές και αφού έχουμε το video element το οποίο τρέχει από τη συσκευή που βρισκόμαστε, χρησιμοποιούμε τον δεύτερο καμβά με id=myCanvas που βρίσκεται κάτω από τον πρώτο καμβά στον οποίο γίνεται stream το βίντεο. Σε αυτόν τον καμβά θα «ζωγραφίσουμε» την εικόνα που θέλουμε να κάνουμε Snap από το video stream (που στην συγκεκριμένη θα γίνει και η επεξεργασία OCR).

```
var canvas = document.getElementById('myCanvas');

var context = canvas.getContext('2d');

// Trigger photo take

document.getElementById("snap").addEventListener("click",
function() {

     context.drawImage(video, 0, 0 ) ;

});
```

Από τις παραπάνω εντολές, στην πρώτη εντολή δημιουργούμε ένα variable canvas στο οποίο θα αποθηκευτεί το element με id=myCanvas, αυτός ο χώρος δηλαδή στη σελίδα που θα εμφανιστεί η αποτυπωμένη εικόνα ή αλλιώς η φωτογραφία μας αφού πατήσουμε το Snap.

Στην επόμενη μεταβλητή (variable-var) με όνομα context βάζουμε το αντικείμενο που περιέχει τις μεθόδους και τις ιδιότητες για να «ζωγραφίσουμε» στον καμβά. Αυτό γίνεται με το :

```
getContext('2d')
```

Και στο τέλος με έναυσμα το πάτημα του πλήκτρου Snap :

```
document.getElementById("snap").addEventListener("click"
```

ζωγραφίζουμε στο context την εικόνα που αποτυπώσαμε από το video :

```
function() {

     context.drawImage(video, 0, 0 ) ;

});
```

Κάπου εδώ , και αφού κάτω από το streaming του εκάστοτε video (με βάση τι περιηγητή έχουμε) βρίσκεται το Snapshot που θέλουμε να επεξεργαστούμε έρχεται και το σημείο του javascript κώδικα για το jcrop.

Το jcrop το εντάσουμε στην εφαρμογή για να γίνει όσο ακριβής πρέπει η τελική εικόνα που θα επεξεργαστούμε για να εξάγουμε το κείμενο της. Εάν σε μία εικόνα που φωτογραφίσαμε το κείμενο βρίσκεται σε κάποιο σημείο χαμηλά ή στα άκρα, το jcrop θα μας βοηθήσει να αποκόψουμε το σημείο αυτό ώστε να διευκολύνουμε την απόδοση της εφαρμογής και εν τέλη την εξαγωγή κειμένου.

Φυσικά και θα είναι αναγκαία στην πράξη η λειτουργία του jcrop γι'αυτόν ακριβώς τον λόγο συμπεριλαμβάνεται. Ο παρακάτω κώδικας καλεί το javascript jcrop και κάποιες από τις συναρτήσεις του ώστε να κόψει σωστά την εικόνα :

```
$(function(){ $('#myCanvas').Jcrop({ aspectRatio: 0,
onSelect: updateCoords }); });

function updateCoords(c) { $('#x').val(c.x);
$('#y').val(c.y);  $('#w').val(c.w); $('#h').val(c.h);
myF(c);};
```

```
function checkCoords() { if (parseInt($('#w').val()))
return true; alert('Select to crop.');

return false; };
```

Το #myCanvas είναι κάτι σαν δείκτης (με βάση το id) για την εφαρμογή της λειτουργίας αυτής στον καμβά αυτόν, που συγκεκριμένα είναι ο καμβάς που βρίσκεται η φωτογραφία. Το :

```
onSelect: updateCoords
```

δείχνει ότι η συνάρτηση-function updateCoords που μετά καλούμε θα κληθεί όταν γίνει το select ή αλλιώς η επιλογή των συντεταγμένων όπου θα γίνει crop της φωτογραφίας.

Αμέσως μετά καλείται η updateCoords που έχει ορίσματα τις συντεταγμένες όπου θα γίνει το κόψιμο (x,y,width,height) αλλά εκτός από αυτές (και ένα προαιρετικό παράθυρο που δείχνει ποιές είναι) καλέι μέσα της μία άλλη συνάρτηση, την :

```
myF(c);
```

Παρακάτω θα εξηγηθεί ο ρόλος αυτής της συνάρτησης :

```
function myF(c){


$('#x').val(c.x); $('#y').val(c.y); $('#w').val(c.w);
$('#h').val(c.h);

var canvass = document.getElementById('cropped');

canvass.width = c.w;

canvass.height = c.h;


var ctx = canvass.getContext('2d');

ctx.drawImage(myCanvas,c.x,c.y,c.w,c.h,0,0,c.w,c.h);


//OCRAD

var resultText = OCRAD(ctx);

//resultText = resultText.trim();
```

```
//textarea

var text = document.getElementById("myText");

text.innerHTML = resultText;


};
```

Αυτή η συνάρτηση καλείται μέσα στην updateCoords ώστε να μην χαθούν οι συντεταγμένες. Κρατάει τις συντεταγμένες όπως βλέπουμε στην πρώτη εντολή και μετά λειτουργώντας με τον ίδιο τρόπο όπως πριν σχεδιάσαμε τον δεύτερο καμβά με τη φωτογραφία που βγάλαμε έτσι και τώρα χρησιμοποιώντας άλλες μεταβλητές (canvass, ctx) σχεδιάζουμε ή «ζωγραφίζουμε» στον τρίτο και τελευταίο καμβά (όπως τους ορίσαμε στην αρχή αρχή στην html) την εικόνα που προκύπτει από τον κόψιμο της φωτογραφίας με βάση τις συντεταγμένες που ορίστηκαν στο jcrop.

```
ctx.drawImage(myCanvas,c.x,c.y,c.w,c.h,0,0,c.w,c.h);
```

Η παραπάνω εντολή κάνει το ίδιο λοιπόν πράγμα με πιο πριν, απλά παίρνει σαν ορίσματα και τις συντεταγμένες που μας επιστρέφει τη συνάρτηση updateCoords τις οποίες δεν θέλαμε να χάσουμε ώστε να γίνει το κόψιμο.

Τέλος κάνουμε την οπτική αναγνώριση κειμένου με το αρχείο javascript με όνομα ocrad.js με το οποίο στην αρχή ορίσαμε σύνδεσμο :

```
var resultText = OCRAD(ctx);
```

Και πολύ εύκολα το εξαγώμενο κείμενο το δείχνουμε σε μία περιοχή κειμένου που επίσης ορίσαμε στην αρχή :

```
var text = document.getElementById("myText");

text.innerHTML = resultText;
```

Το αποτέλεσμα είναι κείμενο το οποίο μπορεί να γίνει copy paste αλλά και να ανανεωθεί κάνοντας οπτική αναγνώριση σε μία άλλη εικόνα.

6.2.3 Συμπεράσματα

Η οπτική αναγνώριση χαρακτήρων δεν έχει περιορισμούς όσον αφορά τη γλώσσα προγραμματισμού που θα χρησιμοποιηθεί για να γίνει. Είναι όμως σημαντική η λήψη και η εκχώρηση για επεξεργασία μίας εικόνας όσον το δυνατόν λιγότερο παραμορφωμένης και με τη μεγαλύτερη δυνατή ευκρίνεια.

Στην εφαρμογή αυτή βλέπουμε το πόσο εύκολα γίνεται η προσαρμογή μίας δύσκολης λειτουργίας όπως το O.C.R σε μία γλώσσα όπως η Javascript. Εκτός όμως από την λειτουργία αυτή, χαρακτηριστικά εύκολα μπορούμε μέσω της Javascript να «χτίσουμε» μία δική μας εφαρμογή, η οποία θα είναι γρήγορη και εύχρηστη ούτως ώστε να χρησιμοποιηθεί από τον κόσμο μαζικά.

Αυτό ακριβώς είναι η εφαρμογή αυτή. Εκμεταλλευόμαστε τις ευκολίες της Javascript ως προς ώφελός μας και αυτό έχει σαν αποτέλεσμα μία άμεσα αξιοποιήσημη εφαρμογή με θέμα την οπτική αναγνώριση χαρακτήρων.

6.2.4 Θετικά - Αρνητικά

Θετικά στην 2η αυτή εφαρμογή οπτικής αναγνώρισης χαρακτήρων μπορούν να χαρακτηριστούν πολλά. Αρχικά όπως έχει προαναφερθεί η ευκολία χρήσης της εφαρμογής, μιας και το μόνο που χρειάζεται είναι ένας περιηγητής, πράγμα που κάθε υπολογιστής διαθέτει.

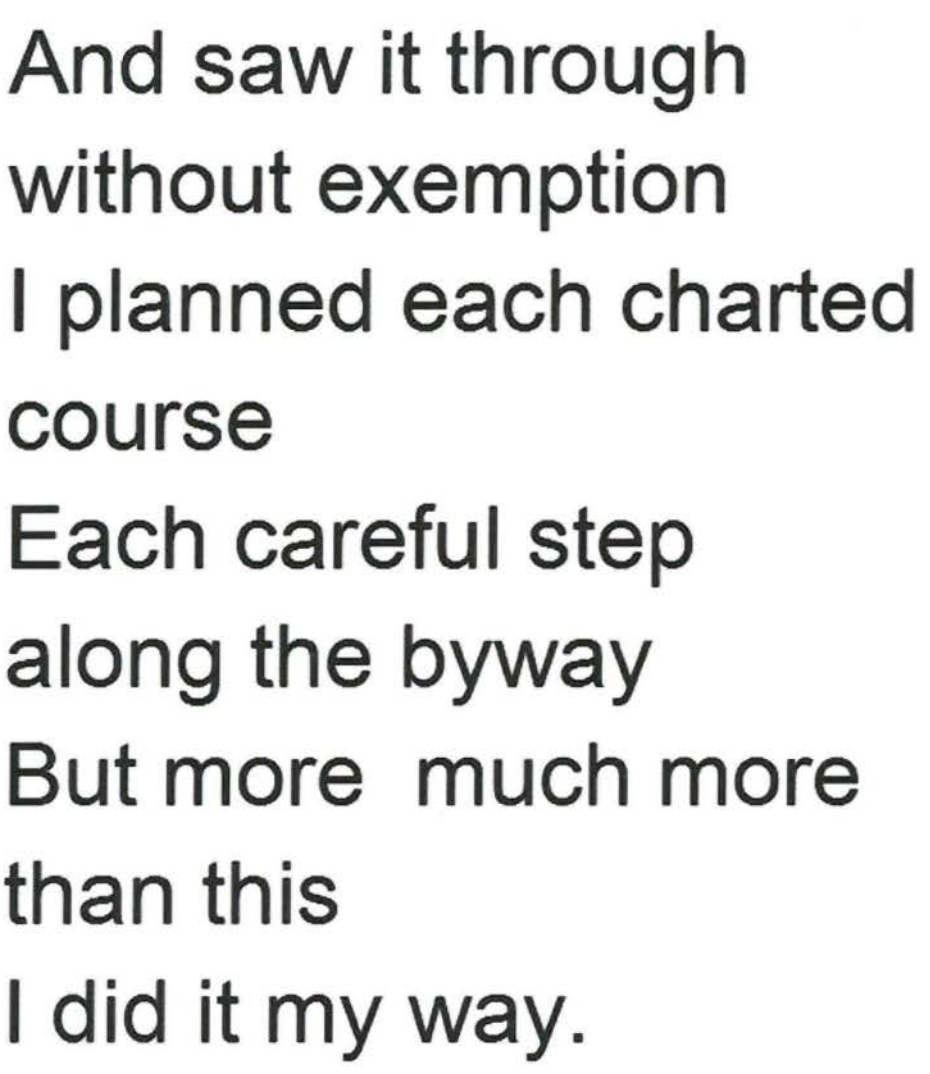
Έπειτα η ταχύτητα εξαγωγής του κειμένου από την εικόνα και το ποσοστό επιτυχίας που θα δούμε και παρακάτω (ξεπερνάει το 96%). Ακόμα να προσθέσουμε την προσβασιμότητα της javascript σε οποιαδήποτε σύγχρονη συσκευή, σκεπτόμενοι φυσικά και μελλοντικές χρήσεις και επεκτάσεις της εφαρμογής σε κινητά.

Τέλος το αποτέλεσμά της, δηλαδή η δημιουργία κειμένου σε μία περιοχή κειμένου, δίνει την δυνατότητα αξιοποίησης του αλλά και πιθανών επεκτάσεων που επίσης θα δούμε και παρακάτω.

Αρνητικό μπορεί να χαρακτηριστεί μόνο το γεγονός της μη εμβάθυνσης στον κώδικα της οπτικής αναγνώρισης (γίνεται χρησιμοποίηση βιβλιοθήκης – ocrad.js- σε javascript) αντίθετα με την πρώτη εφαρμογή.

ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΤΩΝ ΕΦΑΡΜΟΓΩΝ

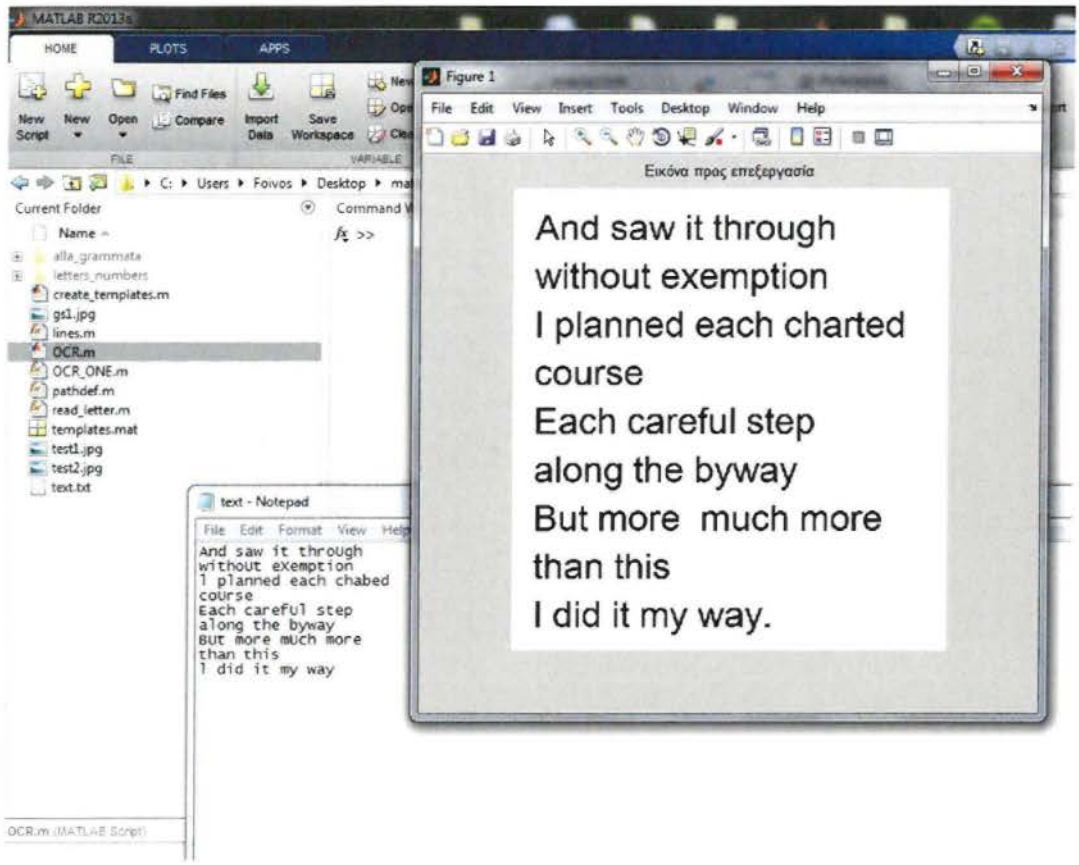
Θα δούμε εδώ σύγκριση των αποτελεσμάτων των δύο εφαρμογών σε μία ίδια εικόνα και συγκεκριμένα στην :



Εικόνα 4 : Κοινή εικόνα για επεξεργασία και με τις δύο εφαρμογες (στίχοι από το τραγούδι My Way)

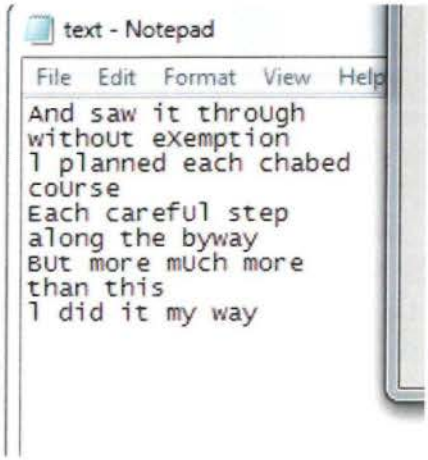
Εφαρμογή 1η (Matlab)

Πατώντας OCR στη γραμμή εντολών του περιβάλλοντος Matlab, παίρνουμε (για τη συγκεκριμένη εικόνα) :



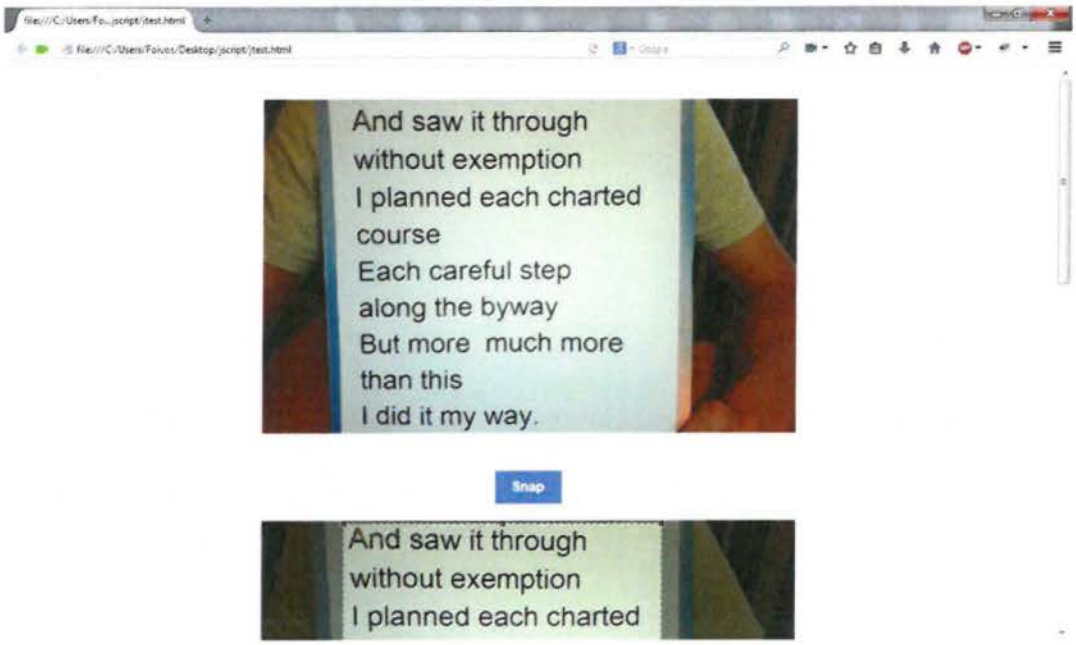
Εικόνα 5 : Αποτέλεσμα εφαρμογής Matlab

Και πιο κοντά στην εικόνα κειμένου text.txt :

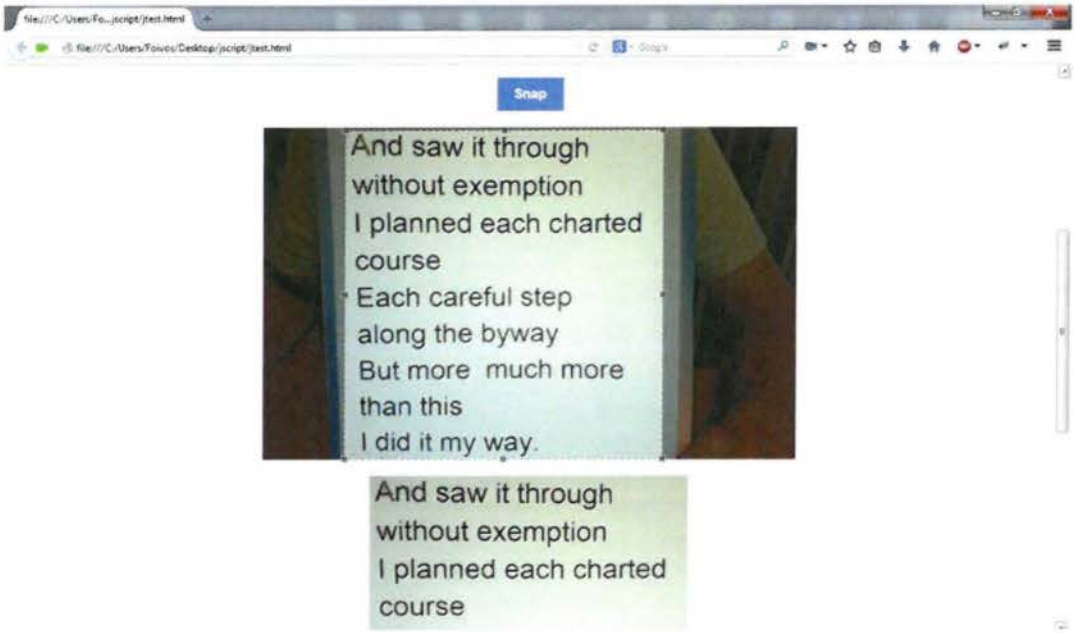


Εφαρμογή 2η (Javascript)

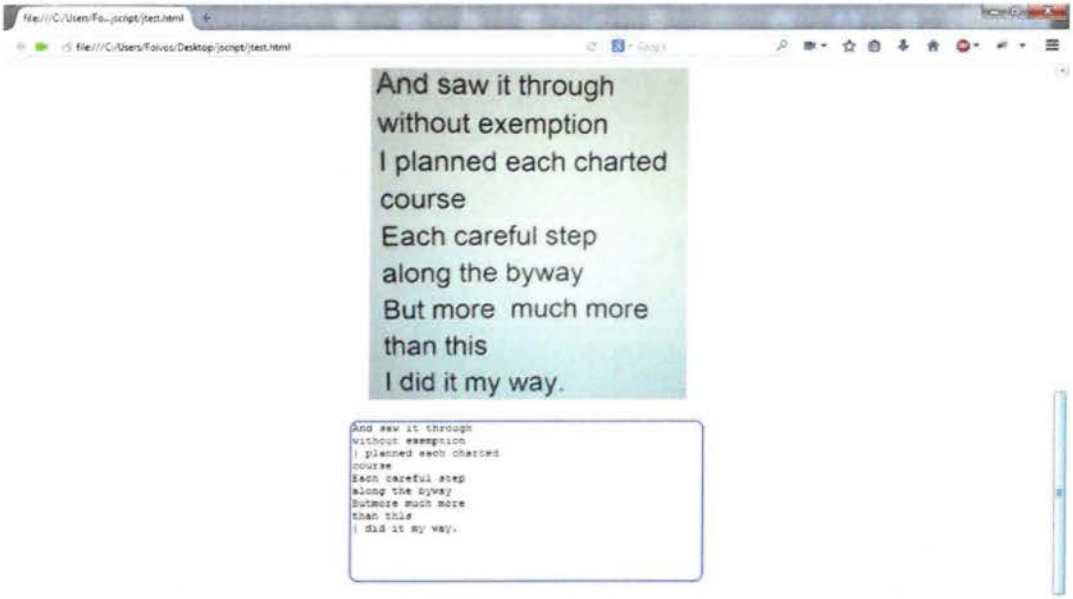
Τρέχουμε το αρχείο html με όνομα jtest.html :



Εικόνα 6 : Πρώτη εικόνα εφαρμογής javascript



Εικόνα 7 : Δεύτερη εικόνα εφαρμογής javascript



Εικόνα 8 : Τρίτη εικόνα εφαρμογής javascript

```
And saw it through
without exemption
| planned each charted
course
Each careful step
along the byway
Butmore much more
than thls
| did it my way.
```

Εικόνα 9 : Τέταρτη εικόνα εφαρμογής javascript

Με βάση την ποσότητα των γραμμάτων του κειμένου αυτού (118 πέραν της τελείας η οποία δεν αναγνωρίζεται στον κώδικα της 1ης εφαρμογής) βγάζουμε ποσοστό επιτυχίας των δύο εφαρμογών μας:

- Στην 1η εφαρμογή το ποσοστό επιτυχίας είναι **>90%** (107/118 εικονοστοιχεία) όπου από αυτά τα 11 γράμματα τα περισσότερα που τα εκλαμβάνω λάθος βγαίνουν ως μεγάλα αντί για μικρά (πχ το γράμμα U). Το μόνο πραγματικό και ουσιαστικό λάθος είναι στη λέξη charted. Εκεί τα γράμματα r και t επειδή σχεδόν τέμνονται μεταξύ τους, ο αλγόριθμος μπερδεύεται από αυτό και τα αντιμετωπίζει σαν ένα εικονοστοιχείο γι'αυτό βγάζει το γράμμα b.

- Στην 2η εφαρμογή το ποσοστό επιτυχίας είναι **>96%** (114/118 εικονοστοιχεία) όπου από αυτά τα 4 γράμματα το ένα λάθος είναι ένα κενό που δεν εντοπίζεται και το γράμμα I σε τρείς περιπτώσεις.

Εν τέλη βλέπουμε ποσοστά 90% και 96% αντίστοιχα τα οποία μπορούν να γίνουν ακόμα μεγαλύτερα και να αγγίξουν το τέλειο. Το σφάλμα είναι έτσι και αλλιώς πολύ μικρό και στις δύο εφαρμογές. Στην 2η εφαρμογή τα αποτελέσματα είναι καλύτερα και αυτό λόγω της καλύτερης αντιμετώπισης των εικονοστοιχείων στην επεξεργασία της εικόνας.

ΑΞΙΟΠΟΙΗΣΗ ΤΩΝ ΕΦΑΡΜΟΓΩΝ ΚΑΙ ΠΙΘΑΝΕΣ ΕΠΕΚΤΑΣΕΙΣ

Η αξιοποίηση των εφαρμογών μπορεί να είναι και πέρα από το πεδίο της πληροφορικής. Η οπτική αναγνώριση χαρακτήρων μπορεί να χρησιμοποιηθεί για διάφορους σκοπούς, όπως η ασφάλεια, η επιβεβαίωση στοιχείων και η μετατροπή όγκου κειμένου σε ψηφιακή μορφή.

Οι δύο εφαρμογές που δείξαμε και αναπτύξαμε στα προηγούμενα κεφάλαια έχουν τον ίδιο προσανατολισμό παρά τη διαφορετική αντιμετώπιση. Η πρώτη εφαρμογή βασισμένη στο περιβάλλον του Matlab, μπορεί να αξιοποιηθεί για την περαιτέρω κατανόηση όπως έχω ξανααναφέρει του θέματος της οπτικής αναγνώρισης αλλά και γενικά της χρήσης της επεξεργασίας εικόνας. Η δεύτερη εφαρμογή δείχνει τη δυνατότητα που μας δίνουν οι σύγχρονες γλώσσες web προγραμματισμού και η δυνατότητα αξιοποίησής τους για τη δημιουργία εφαρμογών που βασίζονται στην επεξεργασία εικόνας και στη προκειμένη περίπτωση συγκεκριμένα στην οπτική αναγνώριση.

Πιθανές επεκτάσεις για την πρώτη εφαρμογή είναι παραδείγματος χάρην η εκπαιδευτική αξιοποίησή της αλλά ακόμα και η δημιουργία συγκεκριμένης συνάρτησης στο Matlab που να εξάγει κείμενο από εικόνα και να είναι μέρος μεγαλύτερων προγραμμάτων και εφαρμογών.

Στην δεύτερη εφαρμογή η μεγαλύτερη και πιο άμεση επέκτασή της είναι στον χώρο των κινητών συσκευών. Με λίγα λόγια μίας web application η οποία θα ανταποκρίνεται στις απαιτήσεις της εποχής και θα είναι απόλυτα προσβάσιμη στις περισσότερες συσκευές. Με την ευρεία χρήση της φωτογραφικής μηχανής στα κινητά και βλέποντας ότι η ίδια η εφαρμογή βασίζεται σε αυτή τη δυνατότητα, καταλαβαίνουμε την αμεσότητα που οι σύγχρονες τεχνολογίες προσφέρουν σε τέτοιες εφαρμογές αλλά και την μεγάλη ταύτιση της ίδιας της εφαρμογής στην εποχή μας.

Άλλη αξιοποίηση αλλά και επεκτάσεις αφορούν την βιομηχανική εφαρμογή της σε αναγνώριση προτύπων π.χ στην βιομηχανία φαρμάκων μετατρέποντας τους κωδικούς άμεσα σε ψηφιακή μορφή και αποτρέποντας την πιθανότητα σφαλμάτων στην τύπωση ή άλλων τυχόν λαθών.

ΠΕΡΙΛΗΨΗ

Στα πρώτα κεφάλαια έγινε μία θεωρητική εισαγωγή στην έννοια της επεξεργασίας εικόνας, σε σημαντικές γνώσεις που αφορούν την ψηφιακή εικόνα και έπειτα στην οπτική αναγνώριση χαρακτήρων. Έγινε ιστορική αναδρομή και διαχωρισμός των λογισμικών αναγνώρισης χαρακτήρων. Επίσης έγινε εισαγωγή στο περιβάλλον του Matlab αλλά και στην γλώσσα σήμανσης Javascript και επεξηγήθηκαν κάποια βασικά πράγματα που θα χρειαζόντουσαν αργότερα στην υλοποίηση. Έγινε η ανάπτυξη του κώδικα των εφαρμογών ξεχωριστά και η λεπτομερής επεξήγησή του. Αναπτύχθηκαν τα συμπεράσματα αλλά και τα θετικά και τα αρνητικά της κάθε εφαρμογής. Τέλος έγινε η σύγκριση των δύο εφαρμογών με το αποτέλεσμά τους και αναφορά στην αξιοποίηση τους και τις πιθανές επεκτάσεις τους.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην πτυχιακή αυτή εργασία συμπεραίνουμε τις δυνατότητες που έχουμε μέσω του προγραμματισμού και της σωστής αλλά και ευφάνταστης χρησιμοποίησης και αξιοποίησής του μέσω των σύγχρονων τεχνολογιών. Η οπτική αναγνώριση χαρακτήρων είναι μόνο μία από τις άπειρες ιδέες και εφαρμογές που μπορεί να έχει η επεξεργασία εικόνας στη ζωή μας μέσω των ευκολιών που ο προγραμματισμός δημιουργεί. Οι τρόποι ανάπτυξης και επίλυσης των διάφορων στόχων που βάζουμε είναι στο χέρι μας μιας και οι δυνατότητες που μας δίνονται με προγράμματα όπως το Matlab ή γλώσσες προγραμματισμού όπως η Javascript αλλά και ακόμα η C, C++,Java κλπ που δεν χρησιμοποιούμε εδώ είναι απεριόριστες.

Το μόνο που μπορεί να σταθεί εμπόδιο είναι η φαντασία μας και η δημιουργικότητά μας. Όμως αν αυτά τα δύο υπάρχουν, τότε οι λειγουργίες και οι εφαρμογές που θα μπορούμε να πραγματοποιήσουμε θα περιορίζονται μονάχα από τον παράγοντα εξέλιξης της τεχνολογίας. Οι εξελίξεις στον προγραμματισμό και γενικά στην τεχνολογία είναι ασταμάτητες. Αν μόνο μπορούμε να διαχειριστούμε σωστά τις εξελίξεις αυτές και τις δυνατότητες που εμείς οι ίδιοι έχουμε, τότε θα μπορούμε να πετύχουμε πολλά παραπάνω απ'όσα στο παρελθόν έχουμε καταφέρει.

ΒΙΒΛΙΟΓΡΑΦΙΑ

11.1 Ελληνική και Ξένη Βιβλιογραφία

- Γ.Γεωργίου – Χ.Ξενοφώντος , Εισαγωγή στο Matlab 2007
- Chris Solomon – Toby Breckon , Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab
- Alasdair McAndrew , An Introduction to Digital Image Processing with Matlab
- Gerard Blanchet – Maurice Charbit , Digital Signal and Image Processing using Matlab
- David Flanagan , Javascript: The Definitive Guide
- Jonathan Chaffer , Learning jQuery – Fourth Edition

11.2 Αναφορές

- http://en.wikipedia.org/wiki/Image_processing
- http://compus.uom.gr/MIS118/document/Dialeksh_03_-_Eikona_-Eisagwgh_sto_photoshop/07-SYSPOL-U5_Image-A.pdf
- http://en.wikipedia.org/wiki/Optical_character_recognition
- http://en.wikipedia.org/wiki/Color_depth
- http://el.wikipedia.org/wiki/MATLAB
- http://www2.ucy.ac.cy/~georgios/bookfiles/MATLABbook.pdf
- http://el.wikipedia.org/wiki/JavaScript
- http://el.wikipedia.org/wiki/CSS
- http://www.caam.rice.edu/~timwar/CAAM210/OCR.html

- http://html5hub.com/using-the-getusermedia-api-with-the-html5-video-and-canvas-elements/#i.1dxeumx6sgdxuy

- http://davidwalsh.name/browser-camera

- http://pulkitgoyal.in/javascript-image-manipulation-using-html5-canvas-element-tutorial/

- http://mrbool.com/jquery-crop-cropping-images-with-jcrop/26243
- http://docs.webplatform.org/wiki/concepts/programming/drawing_images_onto_canvas

- https://developer.mozilla.org/en-US/docs/Web/HTML/Canvas/Drawing_DOM_objects_into_a_canvas