06/10/2019

ΔΙΑΧΩΡΙΣΜΟΣ ΓΡΑΜΜΩΝ ΚΑΙ ΛΕΞΕΩΝ ΑΠΟ ΧΕΙΡΟΓΡΑΦΟ ΚΕΙΜΕΝΟ

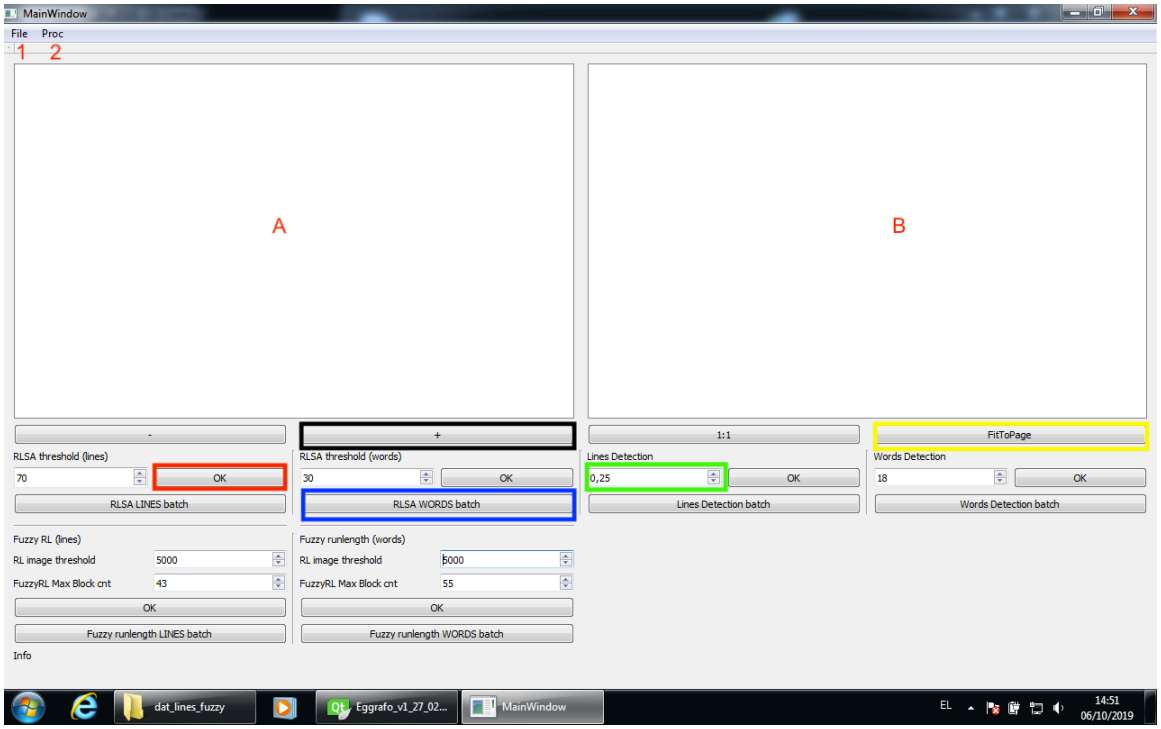
ΕΙΣΑΓΩΓΗ:

Σε αυτή την προσπάθεια για να γίνει αναγνώριση γραμμών και λέξεων από χειρόγραφο κείμενο υλοποιήθηκαν 3 διαφορετικές τεχνικές:

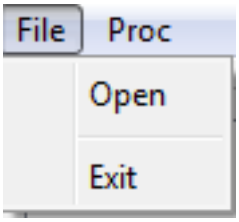
1. Εντοπισμός γραμμών/λέξεων με προβολές
2. Εντοπισμός γραμμών/λέξεων με χρίση του αλγορίθμου RLSA (Run Length Smoothing Algorithm)
3. Εντοπισμός γραμμών/λέξεων με χρίση αλγορίθμου Fuzzy Run Length Algorithm

ΠΕΡΙΒΑΛΛΟΝ ΧΡΙΣΤΗ

Κατά την εκτέλεση του προγράμματος εμφανίζεται ένα παράθυρο με την ονομασία “Main Window”



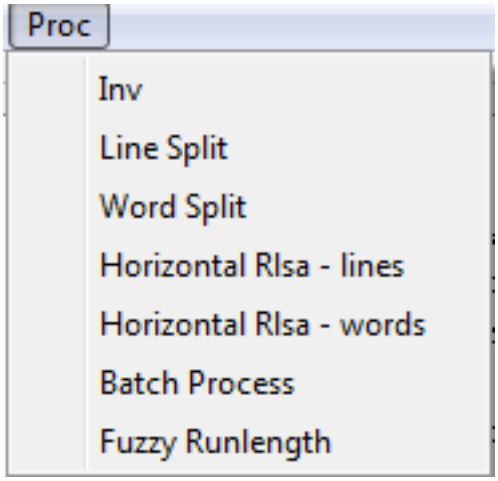
Στην περιοχή «1» που έχει την ονομασία «File» είναι ένα Dropdown menu που έχει μέσα 2 επιλογές:



1. **Open**:
   Πατώντας αυτό το πλήκτρο, ανοίγει ένα παράθυρο πλοήγησης στα αρχεία του υπολογιστή μας, για να διαλέξουμε κάποια εικόνα που πρέπει να φορτώσουμε στο πρόγραμμα μας για να μπορέσει να γίνει η οποιαδήποτε ανάλυση για τις μεθόδους που θέλουμε να χρησιμοποιήσουμε
   ΠΡΟΣΟΧΗ, αν δεν φορτώσουμε κάποια εικόνα δεν θα μπορέσουμε να δούμε οπτικά αποτελέσματα κατά την εκτέλεση της οποιασδήποτε μεθόδου που επιθυμούμε να δούμε αποτελέσματα.
2. **Exit** : Τερματίζει την εκτέλεση του προγράμματος μας.

Στην περιοχή «2» που έχει την ονομασία «Proc» είναι ένα Dropdown menu που έχει μέσα τις εξής επιλογές:

1. **Inv** : εμφάνιση αρνητικών χρωμάτων από την εικόνα που ανοίξαμε
2. **Line Split :** εφαρμογή της μεθόδου αναγνώρισης γραμμών με προβολές
3. **Word Split :** εφαρμογή της μεθόδου αναγνώρισης λέξεων με προβολές
4. **Horizontal Rlsa – lines:** εφαρμογή της μεθόδου Rlsa για τον εντοπισμό γραμμών
5. **Horizontal Rlsa -words:** εφαρμογή της μεθόδου Rlsa για τον εντοπισμό λέξεων
6. **Batch Process:** ανοίγει ένα παράθυρο του συστήματος και ζητάει να δώσουμε έναν φάκελο που θα διαβάσει όλες τις εικόνες που εμπεριέχονται σε αυτόν και θα τρέξει όλες τις μεθόδους (χωρίς οπτικά αποτελέσματα) και μέσα στον φάκελο που διάβασε τις εικόνες θα δημιουργήσει ξεχωριστούς φακέλους με ονομασία τις κάθε μεθόδου και εκεί θα αποθηκεύσει τα αποτελέσματα σε ένα αρχείο **.dat**
7. **Fuzzy Runlength:** εφαρμογή της μεθόδου Fuzzy Runlength για τον εντοπισμό λέξεων



Στην περιοχή «A» θα εμφανιστεί η εικόνα που φορτώσαμε από το dropdown>File>Open
ενώ στην περιοχή «B» θα εμφανιστεί ένα αντίγραφο της εικόνας «A»
με διαφορετικούς χρωματισμούς ανά γραμμή (αν κάνουμε εντοπισμό για γραμμές) η' ανά λέξη (αν κάνουμε εντοπισμό για λέξεις)

Στο κουμπί «+» που είναι με μαύρο περίγραμμα μπορούμε να κάνουμε μεγέθυνση (Zoom In) των εικόνων A και B ταυτόχρονα, για να δούμε συγκριτικά αποτελέσματα από την μια εικόνα στην άλλη, καθώς και

μπορούμε να τις μικρύνουμε τις εικόνες (Zoom out)

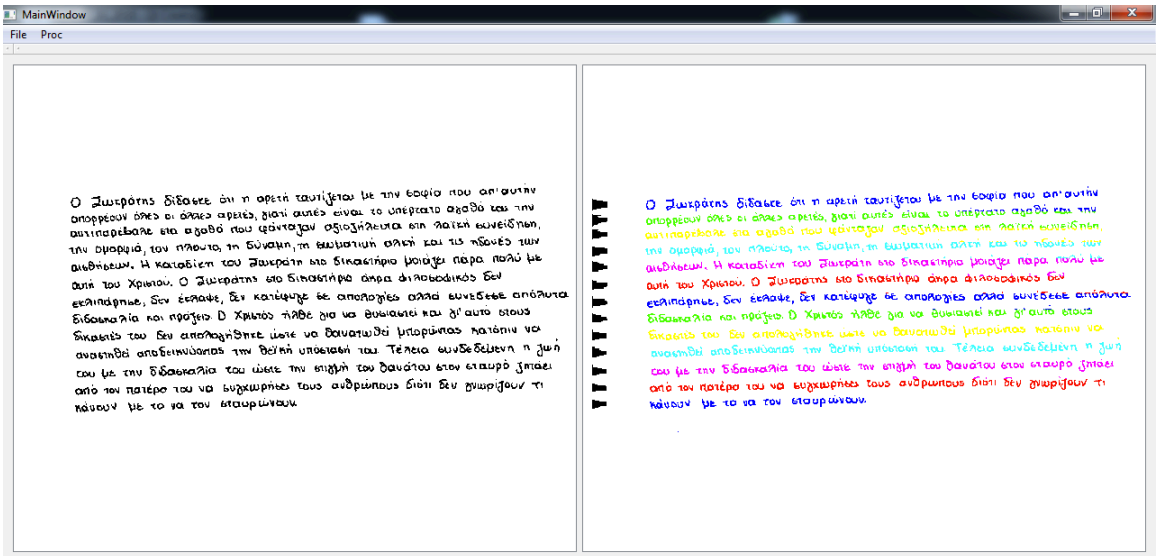
ΠΡΟΣΟΧΗ: για να μην λειτουργήσει αυτό ασύγχρονα θα πρέπει πρώτα να φέρουμε και τις 2 εικόνες στην ιδιά κλίμακα για να μπορέσουν να κάνουν ταυτόχρονα ζουμ.
Αυτό μπορούμε να το πετύχουμε με την εκτέλεση του κουμπιού «FitTo Page» (που είναι μαρκαρισμένο με κίτρινο περίγραμμα)

Τέλος , οι περιοχές που είναι μαρκαρισμένες με κόκκινο(ΟΚ),μωβ(*batch) και πράσινο(καταχώρηση μεταβλητής με μορφή Slingbox) χρώμα εμφανίζονται σε κάθε κουτί που περικλείεται .
όλες οι μέθοδοι έχουν μια κεφαλίδα , ένα κουμπί οκ που μόλις πατηθεί ξανά εκτελείτε η μέθοδος με την νέα κατάχτηση μεταβλητής μέσα στο spin Box,ένα(τουλάχιστον) Spin box που αλλάζοντας το περιεχόμενο του βλέπουμε διαφορετικά αποτελέσματα γιατί γίνονται διαφορετικές μετρήσεις, και ένα Batch button που εκτελεί την συγκεκριμένη μέθοδο για πολλά αρχεία αλλά μόνο για αυτή την μέθοδο
ΠΡΟΣΟΧΗ: στο dropdown proc>Batch Process πατώντας το, θα εκτελεστεί για όλες τις μεθόδους και αν έχουν αλλάξει τα δεδομένα στα Spin Boxes θα λάβει υπόψιν του τα νέα δεδομένα.

**Οι μεταβλητές της κάθε μεθόδου θα εξηγηθούν κατά την ανάλυση της κάθε μεθόδου.

ΔΙΑΧΩΡΙΣΜΟΣ ΓΡΑΜΜΩΝ ΜΕ ΟΡΙΖΟΝΤΙΕΣ ΠΡΟΒΟΛΕΣ

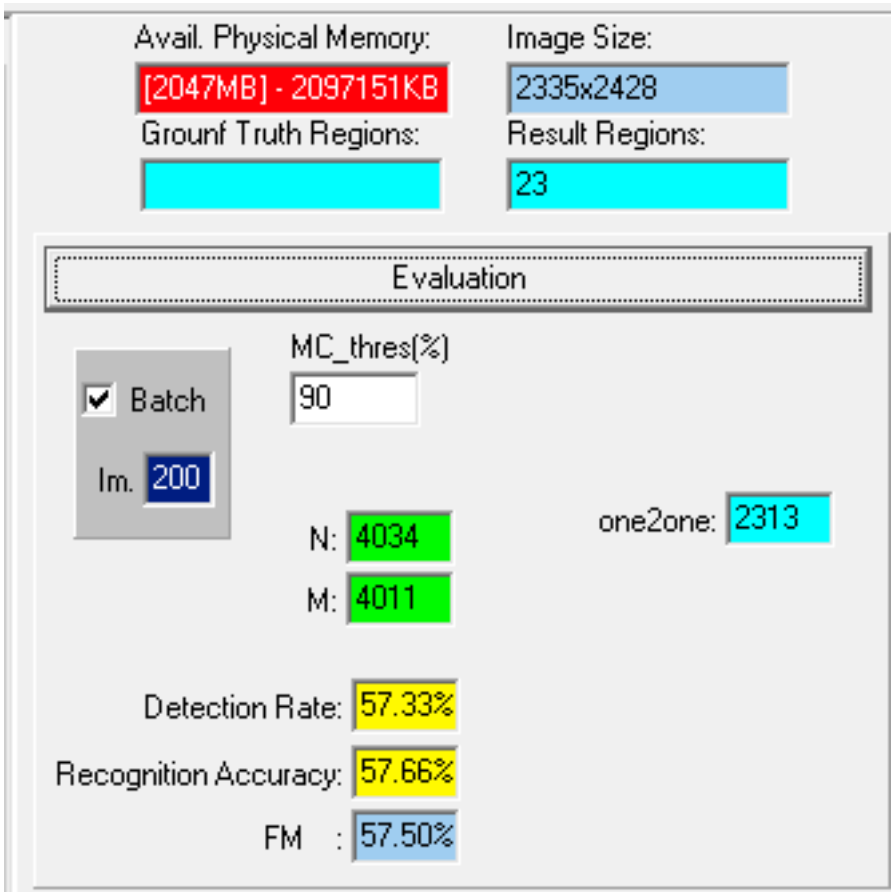
Με αυτή την τεχνική χρησιμοποιούμε οριζόντιες προβολές για να εντοπίσουμε που έχουμε γραμμές και που υπάρχουν κενά ανάμεσα στις γραμμές.
ένα δείγμα είναι στην πάρα κάτω εικόνα:



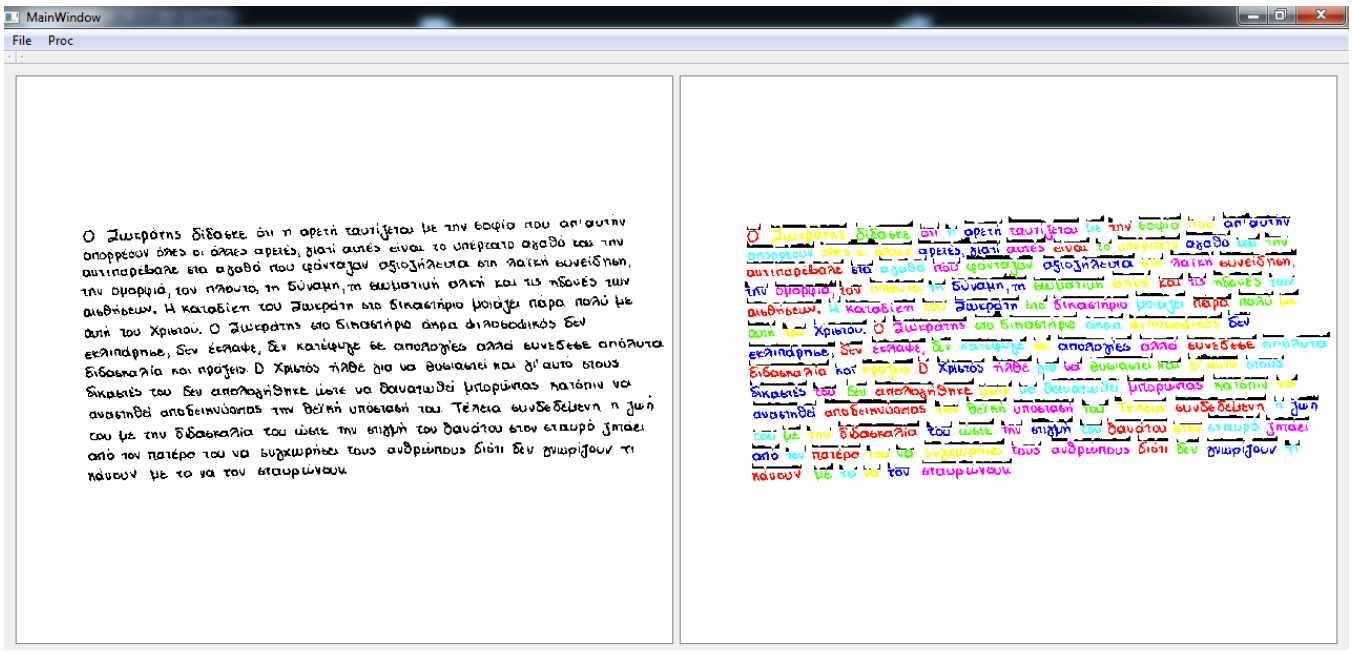
Ανάλυση Μεθόδου:

Υπολογίζονται οι προβολές (άθροισα των pixels σε γραμμή ή στήλη της εικόνας του εγγράφου) σε οριζόντια και κάθετη διεύθυνση.
Στη συνέχεια εντοπίζονται τα διαδοχικά μμηδενικά των προβολών τα οποία θεωρούνται και όρια οριζόντιας ή κάθετης τμηματοποίησης όταν είναι πάνω από ένα ελάχιστο μήκος. Για κάθε τέτοιο όριο, η εικόνα χωρίζεται σε δύο υπο-εικόνες. Για κάθε υπο-εικόνα, η ίδια διαδικασία επαναλαμβάνεται στην άλλη διεύθυνση (αν το πρώτο χώρισμα είναι κάθετο, το επόμενο είναι οριζόντιο) και η όλη διαδικασία επαναλαμβάνεται μέχρι να μην μπορεί να χωριστεί η εικόνα άλλο σε οριζόντια ή κάθετη διεύθυνση. Αν η εικόνα περιέχει θόρυβο, τότε αντι να αναζητούμε μηδενικά στις προβολές, μπορούμε να αναζητούμε μικρό αριθμό pixels, όμως τότε πέφτει η ακρίβεια της μεθόδου. Το αποτέλεσμα της επαναληπτικής εφαρμογής των οριζόντιων και κάθετων προβολών μπορεί να αναπαρασταθεί με την μορφή δέντρων τα οποία ονομάζονται X-Y δένδρα.

Η δειγματοληψία που έδειξε τα καλυτέρα αποτελέσματα ήταν 0.80 στην περιοχή του spinBox και πετύχαμε ποσοστό ακρίβειας 57.33%.
Αυτός ο αλγόριθμος έτρεχε για 200 διαφορετικές εικόνες με διαφορετικά χειρόγραφα κείμενα. Τα αποτελέσματα αναγράφονται στον πάρα κάτω πίνακα


Avail. Physical Memory:
[2047MB] - 2097151KB
Image Size:
2335x2428
Grounf Truth Regions:
Result Regions:
23
Evaluation
MC_thres(%)
90
Batch
Im.
200
N:
4034
M:
4011
one2one:
2313
Detection Rate:
57.33%
Recognition Accuracy:
57.66%
FM :
57.50%

ΔΙΑΧΩΡΙΣΜΟΣ ΛΕΞΕΩΝ ΜΕ ΚΑΘΕΤΕΣ ΠΡΟΒΟΛΕΣ

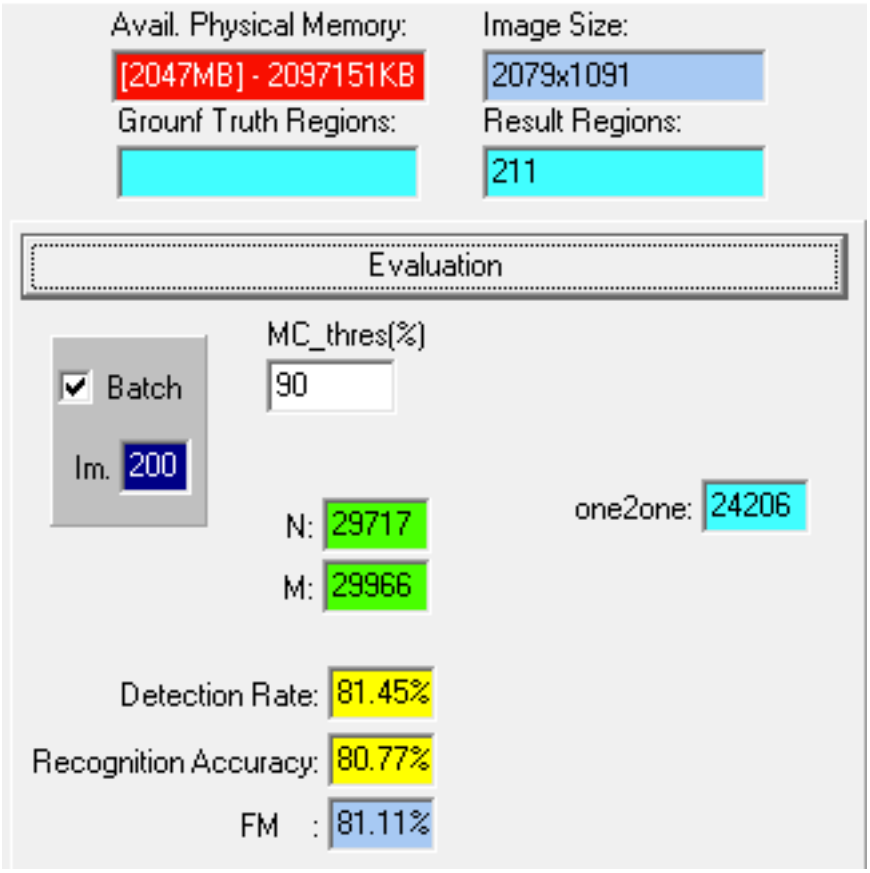


Αυτός ο αλγόριθμος τρέχει συνδικάστηκα με τον πάρα πάνω αλγόριθμο που εντοπίζει γραμμές.
Αφού έχουν αναγνωριστεί και έχουν βρεθεί οι γραμμές, τότε πάνω στις γραμμές μετράμε τα λευκά pixels που υπάρχουν και ανάλογα με το κατώφλι που δίνουμε (αν είναι π.χ. 5 η' π.χ 15 τα λευκά pixels) γίνεται και ο εντοπισμός των λέξεων.

Μετά από διάφορες δοκιμές που έγιναν καταλήξαμε ότι η βέλτιστη τιμη που δόθηκε σαν παράμετρος ήταν : **18**

Threshold	Detection rate	Detection Accuracy
25	44.12%	56.54%
18	47.98%	50.05%
10	34.07%	27.71%

Οπότε με **κατώφλι 18** σε μαζική εκτέλεση 200 χειρόγραφων αρχείων εικόνας ,πέτυχαμε ποσοστό ακρίβειας 81.45 %



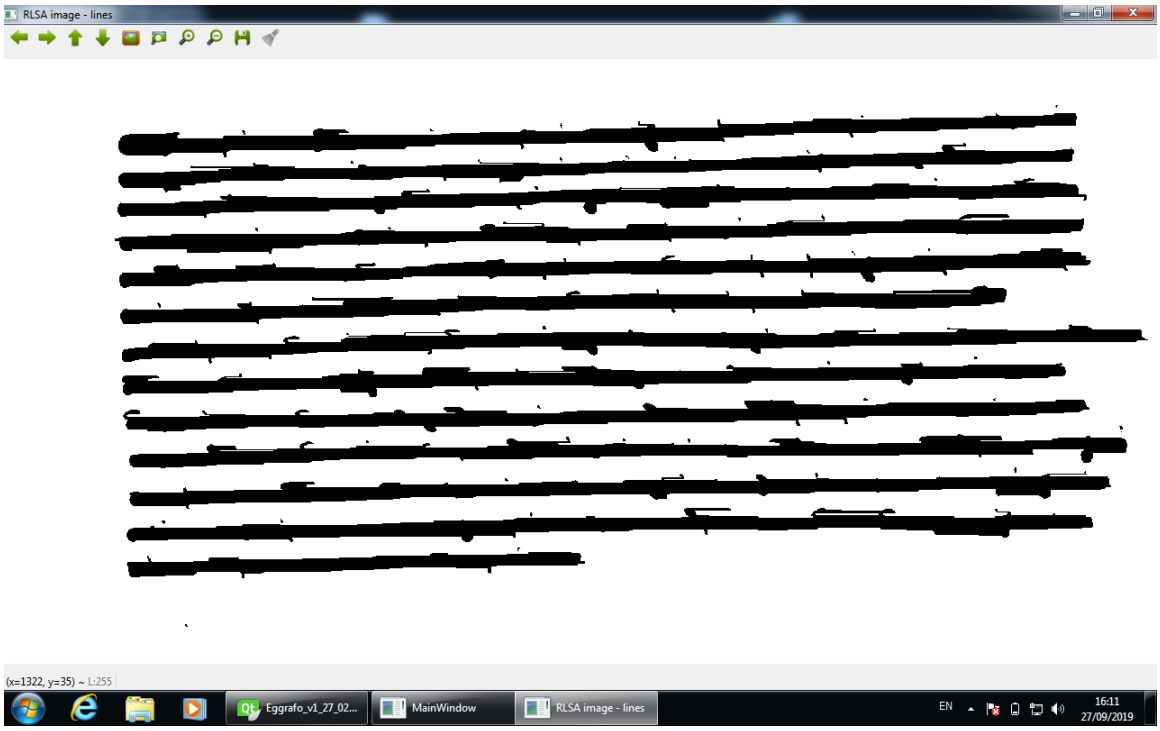
Τα αποτελέσματα αναγράφονται στον πάρα κάτω πίνακα:

1. ΔΙΑΧΩΡΙΣΜΟΣ ΓΡΑΜΜΩΝ ΜΕ RLSA (RUN LENGTH SMOOTH ALGORITHM)

Η εικόνα εξετάζεται ως προς την διεύθυνση σάρωσης που έχει οριστεί και τα διαδοχικά pixels υποβάθρου με μήκος μικρότερο από το μέγιστο μήκος διαδοχικών pixels υποβάθρου T_{max} μετατρέπονται σε σημεία εικόνας.

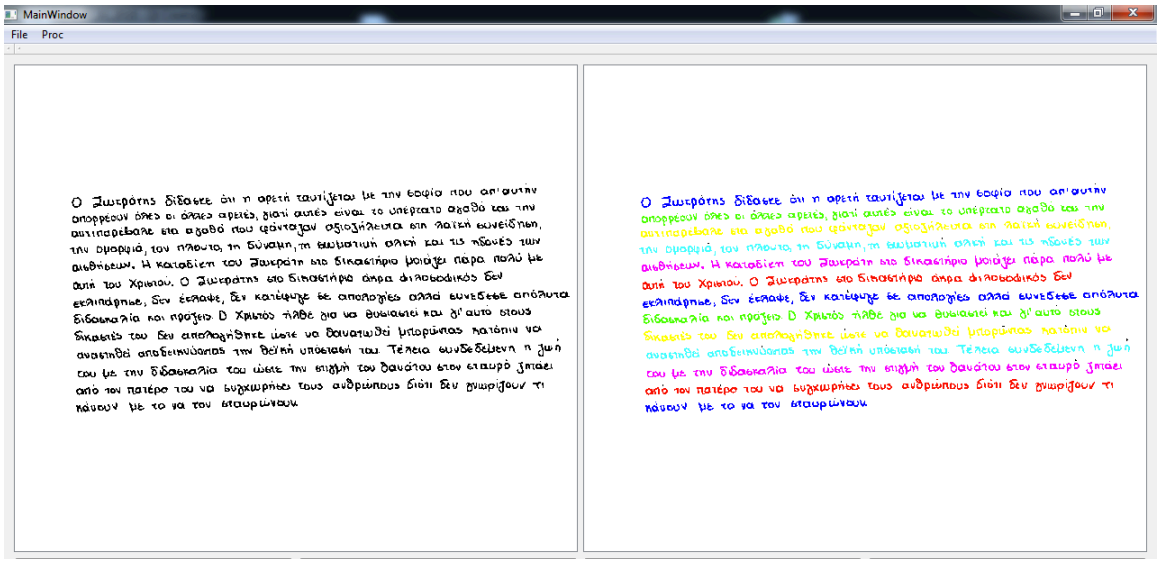
1100011110000001001 $T_{max} = 4$ **1111111110000001111**

Ένα παράδειγμα της εφαρμογής του Rlsa για να δείξουμε τα αποτελέσματα του thresholding είναι η παρακάτω εικόνα:



Η πάρα πάνω εικόνα δέχτηκε την παράμετρο 90.

Και η λέξεις που έχει βρει και η εμφάνιση του τελικού αποτελέσματος είναι στην πάρα κάτω εικόνα:

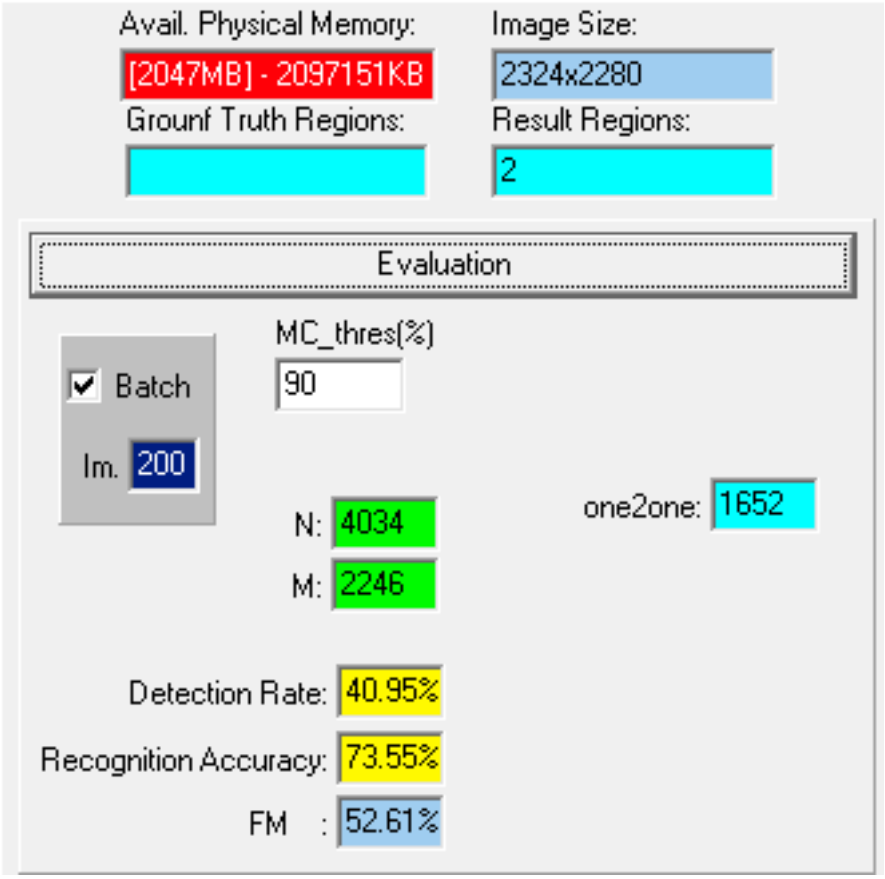


Μετά από διάφορες δοκιμές που έγιναν καταλήξαμε ότι η βέλτιστη τιμή που δόθηκε σαν παράμετρος ήταν : 132

Threshold	Detection rate	Detection Accuracy
70	25.32%	15.78%
100	40.55 %	59.49 %
132	40.95%	73.55 %

Με την συγκεκριμένη τεχνική πετύχαμε ποσοστό ακρίβειας 73.55% και εντοπίσαμε το 40.95 του συνόλου λέξεων και η παράμετρος με τα καλυτέρα αποτελέσματα ήταν =(132 pixels).παρόλο που διορθώσαμε κατά πολύ τον εντοπισμό των γραμμών στο συγκεκριμένο παράδειγμα.

Αυτός ο αλγόριθμος έτρεχε για 200 διαφορετικές εικόνες με διαφορετικά χειρόγραφα κείμενα.



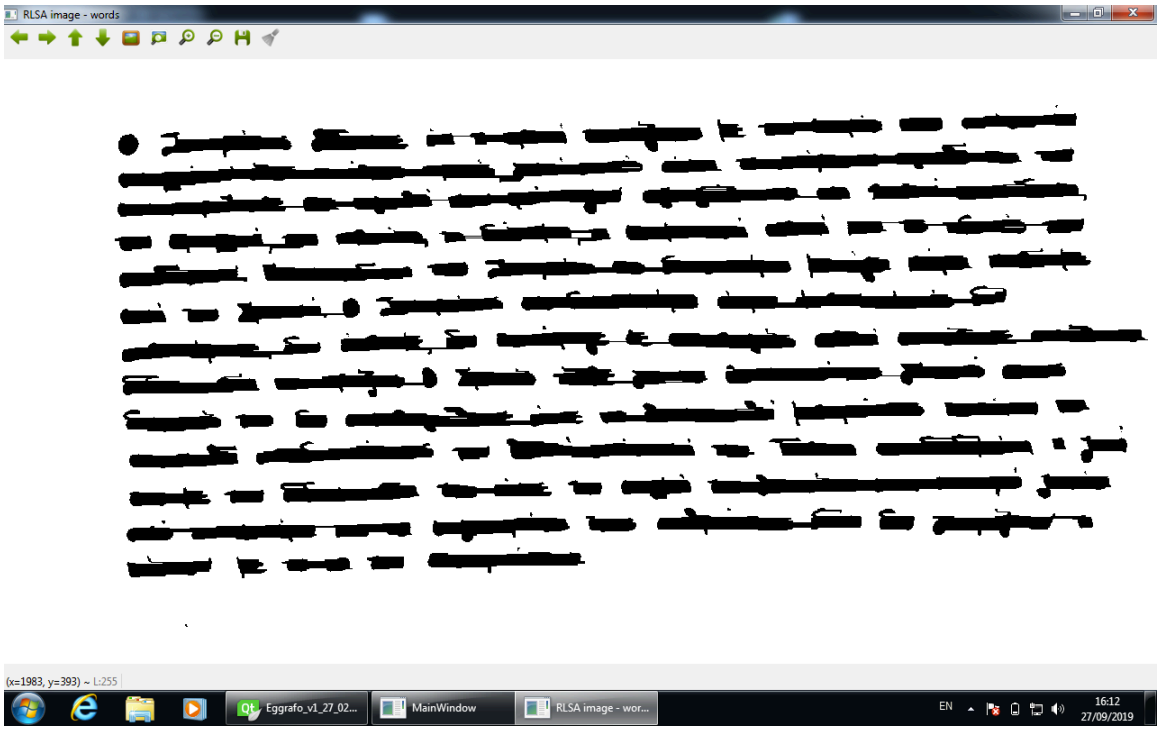
Τα αποτελέσματα αναγράφονται στον πάρα κάτω πίνακα:

Διαχωρισμος Λεξεων με RLSA (Run Length smooth algorithm)

Χρησιμοποιούμε την ιδιά τεχνική με τον διαχωρισμό γραμμών (RLSA)

Απλά με διαφορετικό (πιο μικρό) κατώφλι για να μπορέσει ο αλγόριθμος να αναγνωρίσει πιο μικρές μάζες (δηλαδή λέξεις)

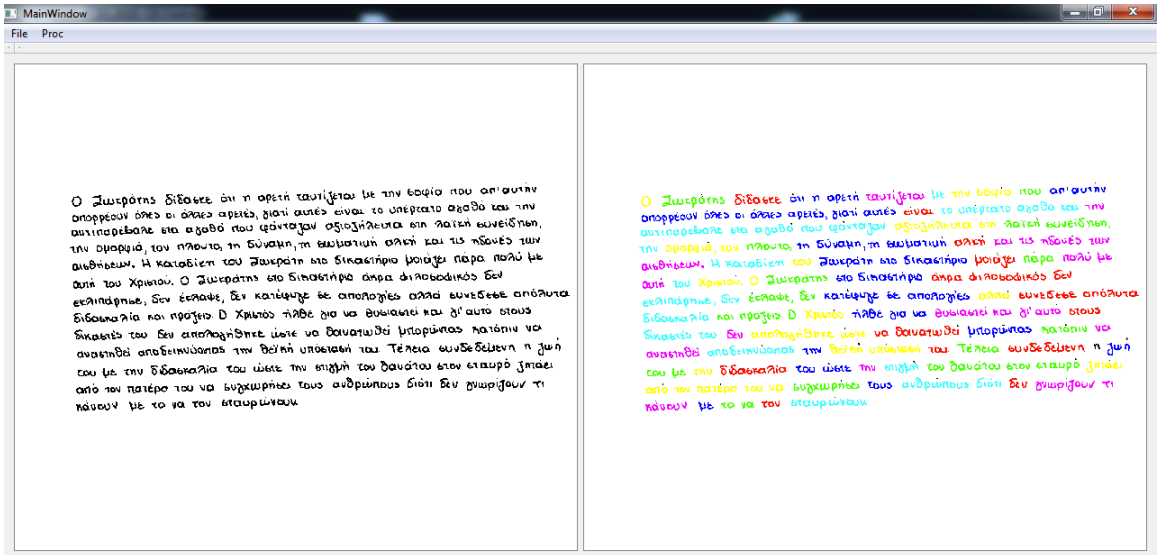
Ένα δείγμα από την εφαρμογή του αλγορίθμου είναι στην πάρα κάτω εικόνα:



Η παράμετρος που δόθηκε ως κατώφλι ήταν = 30

Και το τελικό αποτέλεσμα με τον εντοπισμό και διαχωρισμό των λέξεων είναι στην παρακάτω εικόνα:

Και το τελικό αποτέλεσμα με τον εντοπισμό και διαχωρισμό των λέξεων είναι στην παρακάτω εικόνα:

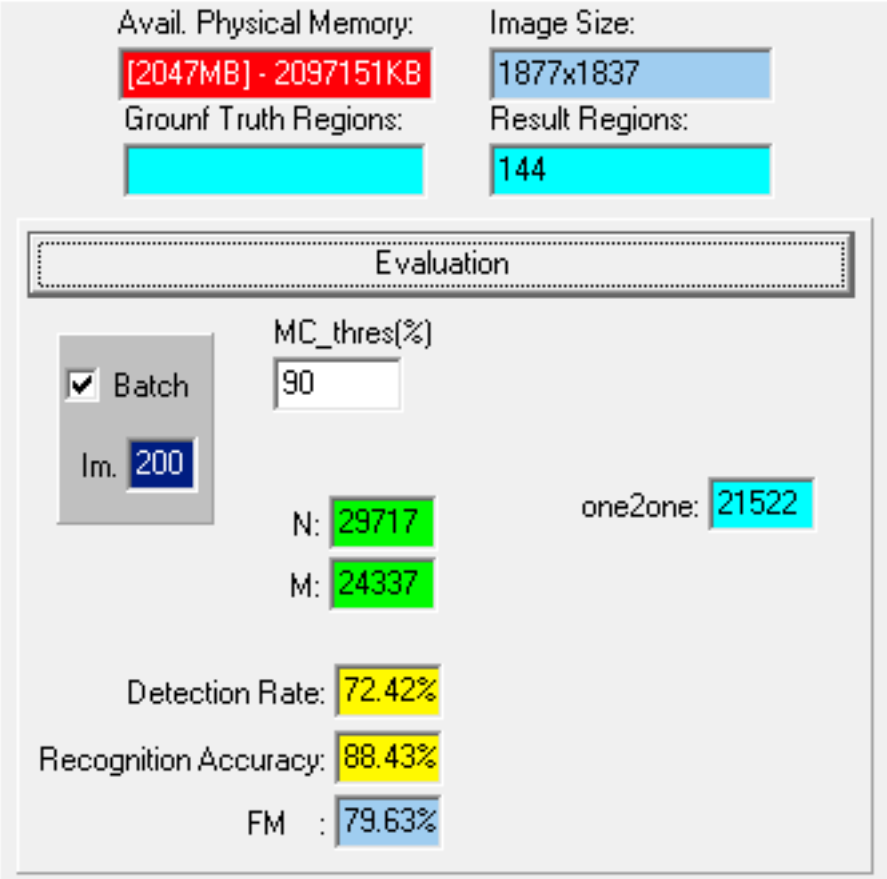


Με την συγκεκριμένη τεχνική πετύχαμε ποσοστό ακρίβειας 88.43 % από τις λέξεις που βρήκαμε και από όλες τις λέξεις που αναζητήσαμε να βρούμε , βρέθηκε το το 72.42%. Η μεταβλητή που δόθηκε με την βέλτιστη απόδοση ήταν = 30.

Μετά από διάφορες δοκιμές που έγιναν καταλήξαμε ότι η βέλτιστη τιμή που δόθηκε σαν παράμετρος ήταν : 30

Threshold	Detection rate	Detection Accuracy
30	72.80%	88.43%
20	70.75%	77.18%
22	71.11 %	82.89 %

Αυτός ο αλγόριθμος έτρεχε για 200 διαφορετικές εικόνες με διαφορετικά χειρόγραφα κείμενα



Τα αποτελέσματα αναγράφονται στον πάρα κάτω πίνακα:

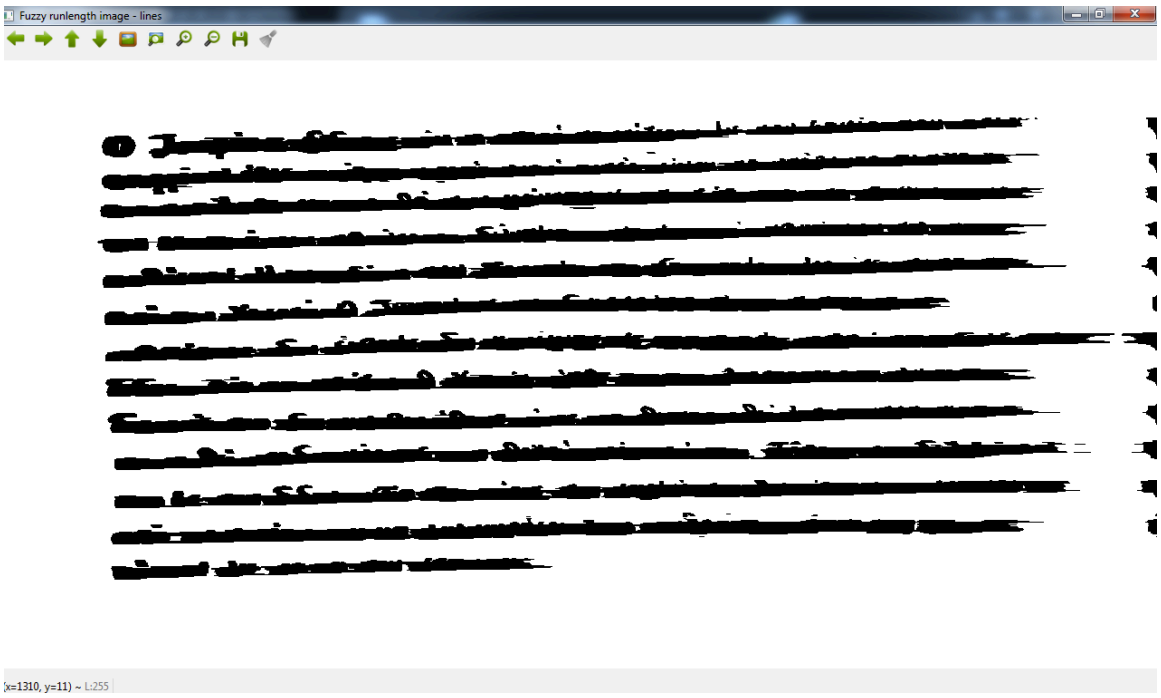
FUZZY RLSA LINES

Αρχικά ελέγχουμε αν έχει φορτωθεί στην περιοχή «A» κάποια εικόνα.
μετα μετατρέπουμε την εικόνα σε σε format rgb_8888. Εκτελείται η δομή fuzzyRlsa() που κάνει το εξής:
Ξεκινάει από αριστερά στα δεξιά και για κάθε ένα μαύρο pixel που θα συναντήσει θα το κάνει push σε μια ουρά, αν αυτά έχει η ουρά είναι περισσότερα από ένα όριο (είναι η μια παράμετρος που δέχεται ο αλγόριθμος , δηλαδή το Max block cnt) τότε βρίσκουμε την απόσταση του σημείου που βρισκόμαστε (σε pixels) πλην (-) του προηγουμένου σημείου (δηλαδή του παλαιοτέρου σημείου που υπάρχει μέσα στην ουρά) ,τότε το αφαιρούμε από την ουρά, αλλιώς αν δεν έχουμε φτάσει στο μέγιστο count ,..τότε βάζουμε στο pixel που βρήκαμε -> την τιμή του προηγουμένου pixel αυξημένη κατά ένα (+1).Μετα κάνουμε το ίδιο από τα αριστερά προς τα δεξιά.

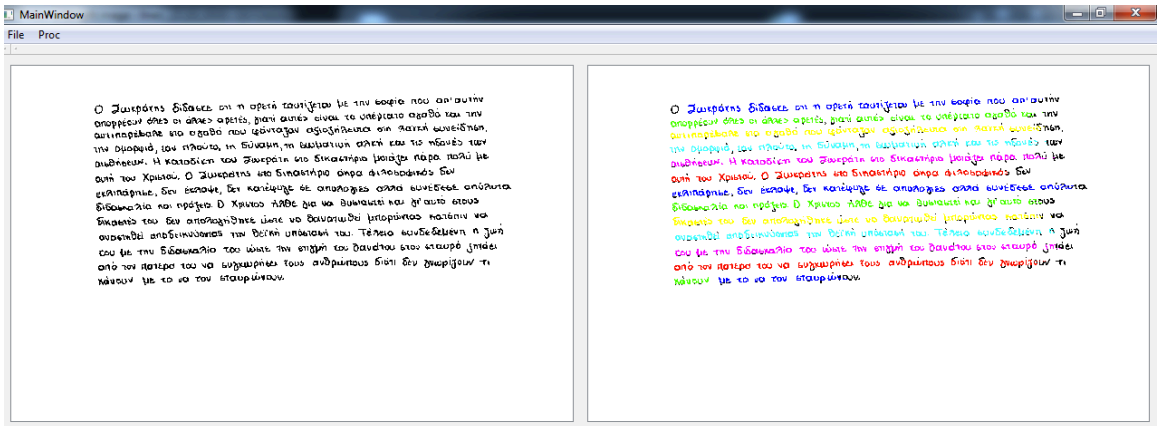
μετα κάνουμε normalizeστην εικόνα και κάνουμε threshold για να βρούμε την μάσκα που θέλουμε (είναι η δεύτερη παράμετρος που δέχεται ο αλγόριθμος), μετα κάνω όπως και στις προηγούμενες τεχνικές blobDetection για να μπορέσουμε να αναγνωρίσουμε αυτές τις μάζες σε γραμμές.

Ομοίως με τις προηγούμενες μεθόδους φτιάχνουμε και τα tags, καθένα διαφορετικό tag θα απεικονίζετε με διαφορετικό χρώμα.
Με την ιδιά διαδικασία κάνουμε την εικόνα σε Qimage για να την φορτώσουμε στο πρόγραμμα μας, για να έχουμε ωρατα αποτελέσματα.

Ένα παράδειγμα της εφαρμογής του αλγορίθμου με παραμέτρους threshold= 5000 και Max block cnt=42 είναι στην πάρα κάτω εικόνα:



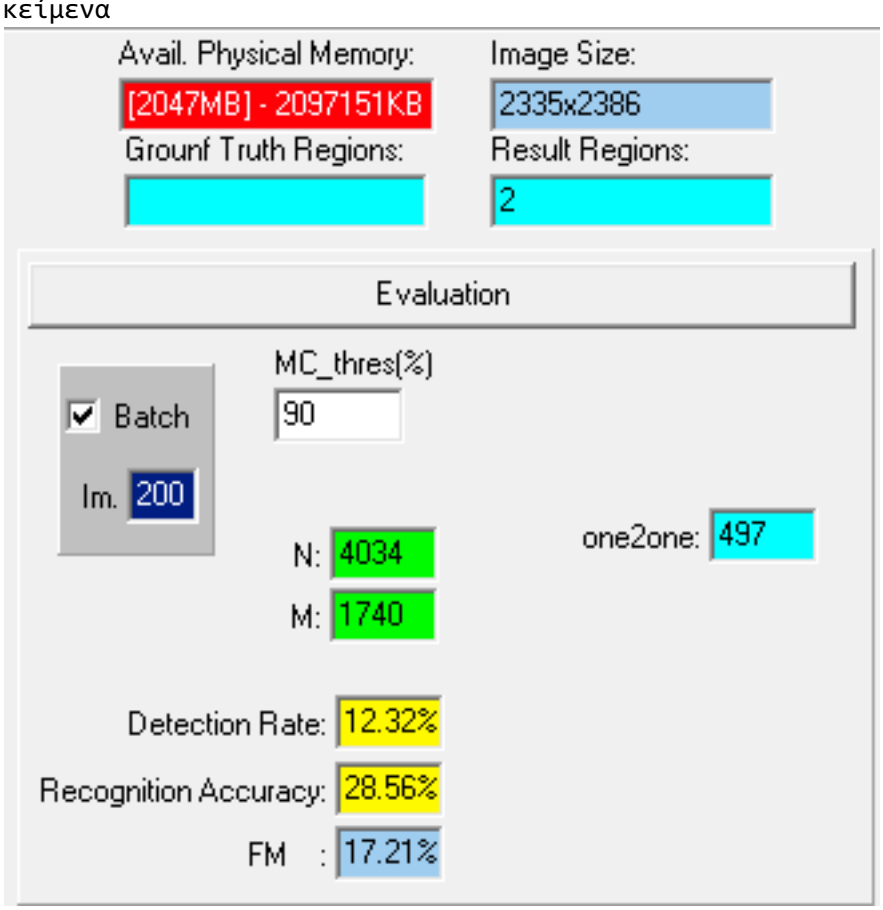
Το τελικό αποτέλεσμα με την προσπάθεια του εντοπισμού γραμμών είναι στην πάρα κάτω εικόνα:



Με την συγκεκριμένη τεχνική πετύχαμε ποσοστό ακρίβειας 28.56 % από τις γραμμές που βρήκαμε και από όλες τις γραμμές που αναζητήσαμε να βρούμε , βρέθηκε το το 12.32%.

Η μεταβλητές που δοθήκαν με την βέλτιστη απόδοση ήταν = Max block cnt =12 και threshold=3000

Αυτός ο αλγόριθμος έτρεχε για 200 διαφορετικές εικόνες με διαφορετικά χειρόγραφα
κείμενα

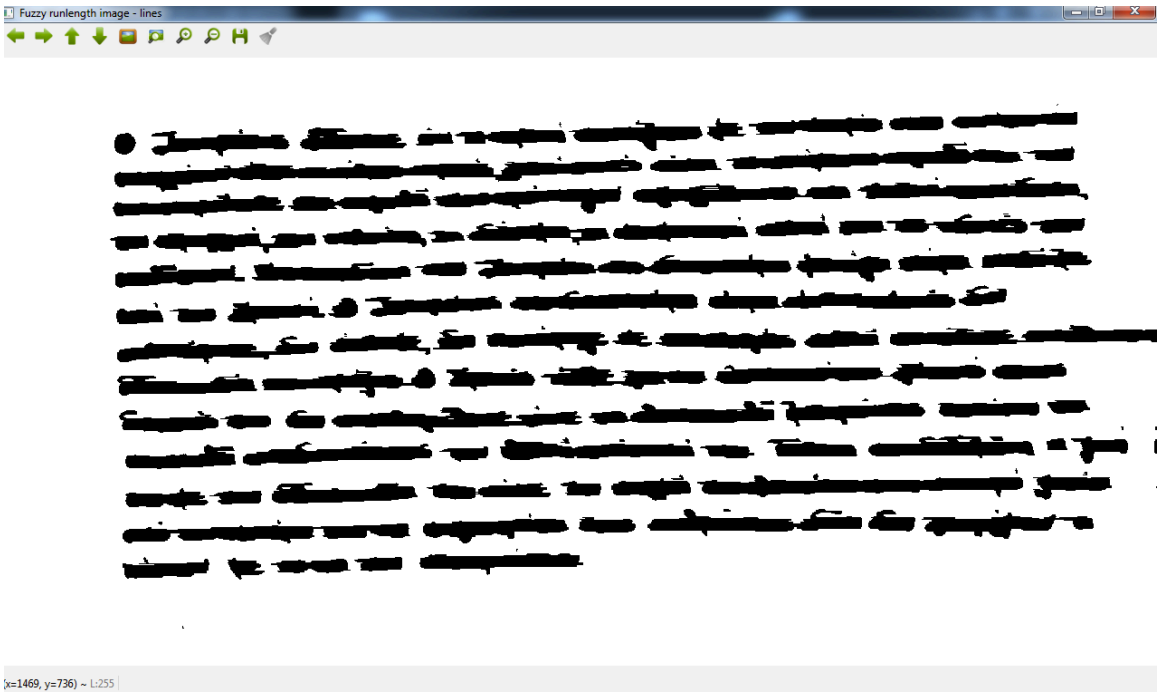


Τα αποτελέσματα αναγράφονται στον πάρα κάτω πίνακα:

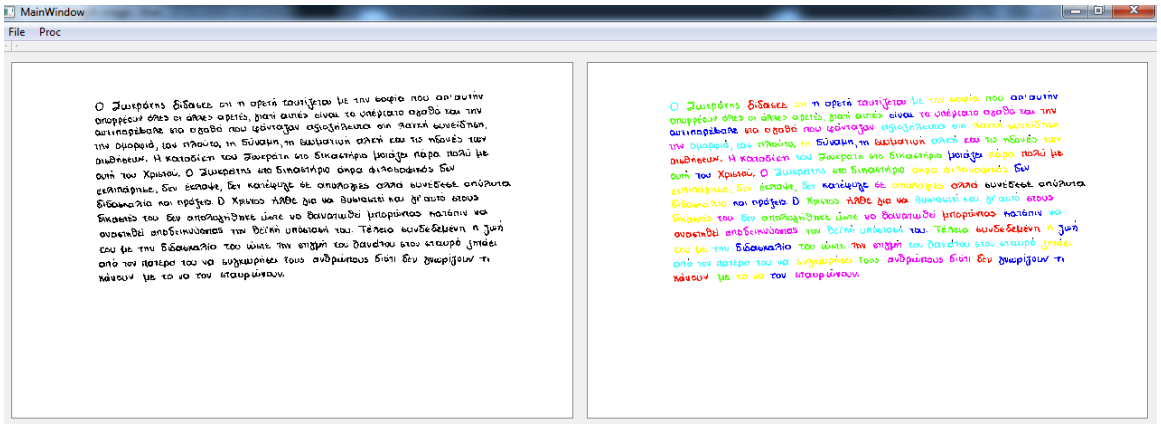
Χρησιμοποιούμε την ιδιά τεχνική με τον διαχωρισμό γραμμών (Fuzzy RLSA Lines)

Απλά με διαφορετικό (πιο μικρό) κατώφλι για να μπορέσει ο αλγόριθμος να αναγνωρίσει πιο μικρές μάζες (δηλαδή λέξεις)

Ένα δείγμα από την εφαρμογή του αλγορίθμου με threshold =500 και Max block cnt=1 είναι στην πάρα κάτω εικόνα:

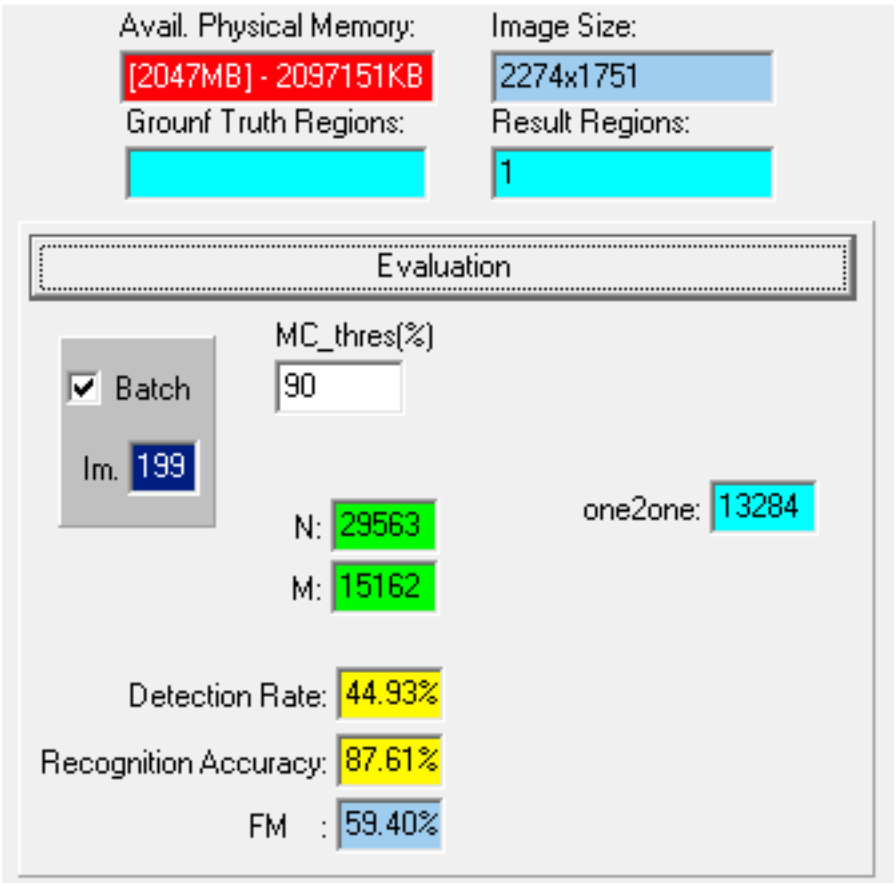


Το τελικό αποτέλεσμα με την προσπάθεια του εντοπισμού λέξεων είναι στην πάρα κάτω εικόνα:



Με την συγκεκριμένη τεχνική πετύχαμε ποσοστό ακρίβειας 87.61 % από τις λέξεις που βρήκαμε και από όλες τις λέξεις που αναζητήσαμε να βρούμε , βρέθηκε το 44.93%.

Η μεταβλητές που δοθήκαν με την βέλτιστη απόδοση ήταν = Max block cnt =12 και threshold=3000



Συμπέρασμα :

Συλλέγοντας τα στατιστικά αναγνώρισης που πέτυχαμε από τους προηγουμένους αλγορίθμους :

	Lines Detection	Words Detection
Projections	57.33%	47.98%
RLSA	40.95%	72 . 80%
Fuzzy Run Length	12.32%	44.93%

Αρα τα καλυτέρα αποτελέσματά σε εντοπισμό γραμμών τα πετύχαμε με την εφαρμογή των οριζοντίων προβολών.

Όσο για των εντοπισμό των λέξεων , το μεγαλύτερο ποσοστό το πέτυχαμε με την εφαρμογή του RLSA.