Επεξεργασία Εικόνας

και

Υπολογιστική Όραση

Σύνθεση ερωτημάτων για αναζήτηση πληροφορίας σε εικόνες εγγράφων

Κατάνιας Βαγγέλης ΑΜ : 2013027

1 ΕΙΣΑΓΩΓΗ

Είναι γνωστό ότι τα παλιά έγγραφα περιέχουν χρήσιμη πληροφορία. Ο παραδοσιακός τρόπος αναζήτησης πληροφορίας σε εικόνες γίνεται με χρήση τεχνικών OCR (Optical Character Recognition). Τα συστήματα OCR δέχονται ως είσοδο μια εικόνα κειμένου και βγάζουν στην έξοδο ένα ASCII κείμενο αυτού του κειμένου. Όμως τα συστήματα OCR απαιτούν εκπαίδευση. Η εκπαίδευση γίνεται χρησιμοποιώντας εικόνες χαρακτήρων από τα έγγραφα στα οποία θέλουμε να ψάξουμε μια πληροφορία. Όμως κάποιες φορές μπορεί να είναι αδύνατον να βρούμε κατάλληλες εικόνες χαρακτήρων .

Για το λόγο αυτό έχουν εφευρεθεί νέες τεχνικές αναζήτησης πληροφορίας σε έγγραφα. Μια από αυτές είναι η τεχνική word-spotting. Στην μέθοδο αυτή ο χρήστης επιλέγει από την εικόνα εγγράφου την λέξη που τον ενδιαφέρει. Στην συνέχεια το σύστημα εμφανίζει τις τοποθεσίες που βρέθηκε η λέξη στη συλλογή των εικόνων εγγράφων. Αυτή η τεχνική όμως απαιτεί από τον χρήστη να ψάξει να βρει την λέξη που τον ενδιαφέρει στην εικόνα-έγγραφο. Αντί όμως ο χρήστης να επιλέγει την εικόνα θα μπορούσε ο χρήστης να δίνει ένα αλφαριθμητικό (string query) και το σύστημα να “μετατραπεί” αυτό το αλφαριθμητικό σε εικόνα ή σε κάποιο αφηρημένο μοντέλο και μετά αυτή η εικόνα θα εισάγεται σε ένα word-spotting σύστημα. Η τεχνική αυτή λέγεται word-retrieval**[1]**

2 ΠΡΟΗΓΟΥΜΕΝΕΣ ΜΕΛΕΤΕΣ

Στο [3] έφτιαξαν ένα σύστημα που παράγει εικόνες λέξεων που μοιάζουν με τον γραφικό χαρακτήρα του χρήστη. Ο χρήστης μέσω ενός tablet και μιας γραφίδας εισάγει αρκετά instances από μια λίστα από ν-γράμματα χαρακτήρων. Αυτά είναι τα λεγόμενα glyphs. Στην συνέχεια το πρόγραμμα συνθέτει (juxtapose) τα glyphs έτσι ώστε να σχηματίσει την λέξη που έχει δώσει ο χρήστης από το πληκτρολόγιο. Επειδή υπάρχει η δυνατότητα αντιστοίχησης περισσότερα από ένα glyphs στην ίδια λέξη η εικόνα που προκύπτει φαίνεται "φυσική" σε κάποιο μη-ειδικό περί ανάλυσης γραφικού χαρακτήρα. Επίσης το σύστημά τους προσφέρει την δυνατότητα εφαρμογής διάφορων γεωμετρικών μετασχηματισμών στα glyphs για να φανεί πιο φυσικό το αποτέλεσμα. Δεν έλαβαν όμως υπόψιν καθόλου τις διακυμάνσεις της πίεσης της γραφίδας (pen pressure). Επίσης αυτή η τεχνική δεν φαίνεται να είναι αποτελεσματική για δημιουργία cursive script.

Ο Wang [4] προσέγγισε το πρόβλημα της σύνθεσης χειρόγραφων χαρακτήρων χρησιμοποιώντας Bézier Splines. Κάθε χαρακτήρας λοιπόν είναι μια B-spline της οποίας τα σημεία (control points) προκύπτουν αν εφαρμόσουμε ένα φίλτρο Gabor πάνω στην εικόνα του χαρακτήρα . Αν πάρουμε αρκετά instances του ίδιου χαρακτήρα και εξάγουμε τα σημεία ελέγχου μπορούμε με τον αλγόριθμο Expectation-Maximization να μάθουμε την κατανομή των σημείων ελέγχου της B-spline κάθε γράμματος. Επίσης εφαρμόζουν διάφορα μοντέλα παραμόρφωσης ώστε να φαίνονται πιο φυσικοί οι χαρακτήρες.

Στο [6] έφτιαξαν ένα σύστημα που επιτρέπει την σύνθεση αγγλικών κειμένων με χρήση glyphs. Μέσω μιας "drawing tablet" ο χρήστης εισάγει όλους τους χαρακτήρες της αγγλικής γλώσσας . Για την ακρίβεια εισάγει τρια διαφορετικά instances για κάθε χαρακτήρα ένα που αναπαριστά τον χαρακτήρα όπως θα ήταν στην αρχή μιας λέξης ,έναν στο ενδιάμεσο και ένα στο τέλος. Για την αναπαράσταση των χαρακτήρων πήραν πολλά σημεία σε αντίθεση με την προσέγγιση του Wang. Οι χαρακτήρες μπορούν να αλλοιωθούν πολύ εύκολα μετακινώντας κάποια από αυτά τα σημεία. Επιπλεόν με την χρήση ενός scaling αλγορίθμου το μέγεθος των γραμμάτων παραμένει ομοιόμορφο. Η κλίση (slant) της λέξης υπολογίζεται ως ο μέσος όρος της κλίσης κάθε γράμματος . Θέωρησαν ότι κάποια γράμματα γράφονται με παραπάνω από ένα stroke (multi-stroke) όπως πχ το

x,f,t, ƶ . Επίσης ανέπτυξαν αλγορίθμους που μαθαίνουν τον τρόπο σύνδεσης μεταξύ των γραμμάτων (ligature) για κάθε πιθανό ζεύγος γραμμάτων.

Στο [1] έφτιάξαν ένα word retrieval σύστημα για οποιοδήποτε είδους γλώσσας (Λατινική,Αραβική,Κινέζικη) . Χρησιμοποίησαν εικονίτσες (glyphs) για την αναπαράσταση κάθε χαρακτήρα. Για κάθε χαρακτήρα ανιχνεύεται η γραμμή αναφοράς του καθώς και ο τύπος σύνδεσης του με άλλους χαρακτήρες. Το σύστημα που έφτιαξαν μπορεί με βάση κάποιους γραμματικούς κανόνες μπορεί να συνθέσει την λέξη με όλες τις πιθανές καταλήξεις.

Στο [8], χρησιμοποιήσαν συνθετικές λέξεις φτιαγμένες με διάφορες γραμματοσειρές και σύνθεσαν ένα Semi-Continuous Hidden Markov Model. για την αναζήτηση σε συλλογές εικόνων-λέξων προερχόμενες από πολλούς συγγραφείς . Το SC-HMM είναι ένα είδος HMM όπου οι emission συναρτήσεις πυκνότητας πιθανότητας είναι ένα μείγμα πολυδιάστατων κατανομών Gauss.
Οι συναρτήσεις πυκνότητας πιθανότητας είναι ίδιες για όλες τις κρυμμένες καταστάσεις με την διαφορά ότι σε κάθε κρυμμένη κατάσταση αλλάζει το βάρος κάθε συνάρτησης. Επίσης το SC-HMM απαιτεί πολύ λίγα παραδείγματα εκπαίδευσης σε σχέση με τα C-HMM (ακόμα και ένα αρκεί). Το πολυδιάστατο μείγμα κατανομών Gauss προκύπτει αν εφαρμόσουμε τον Expectation-Maximization πάνω στο feature set της χειρόγραφης συλλογής. Τα features εξάγονται από τις εικόνες μέσω ενός αλγορίθμου που βασίζεται σε ένα "κυλιόμενο παράθυρο" .Στην συνέχεια με τον αλγόριθμο Baum-Welch μαθαίνονται οι παράμετροι του SC-HMM (πίνακας πιθανοτήτων μετάβασης, πίνακας βαρών). Τέλος με τον αλγόριθμο forward-backward υπολογίζεται ένα σκορ ομοιότητας. Σύμφωνα με τα πειράματα των συγγραφέων η μέθοδος αυτή έχει καλύτερη απόδοση από τις κλασσικές μεθόδους αναζήτησης

3 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΣΥΝΘΕΣΗΣ ΣΕ ΤΥΠΩΜΕΝΑ ΕΓΓΡΑΦΑ

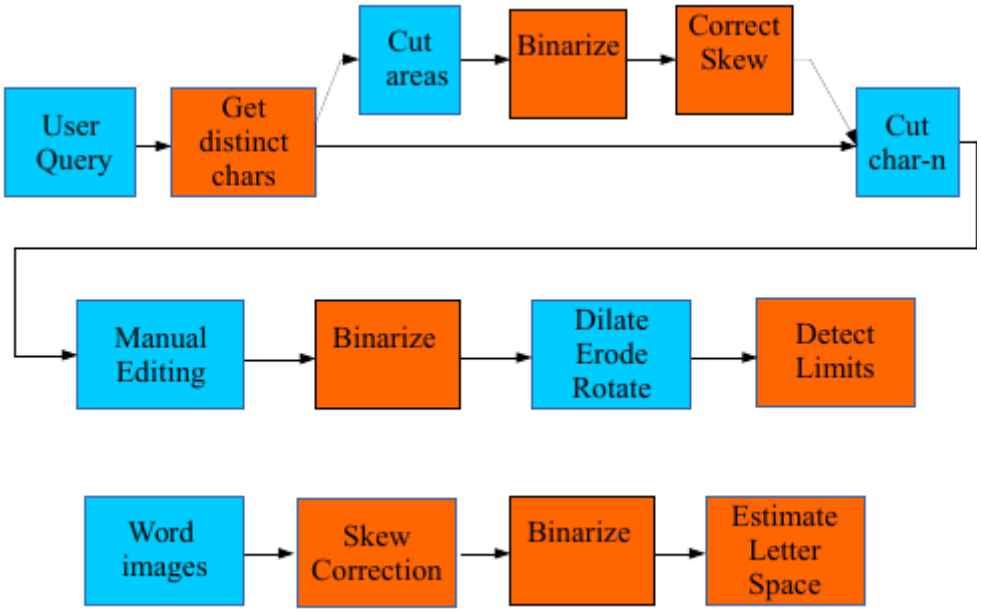
3.1 ΕΙΣΑΓΩΓΗ

Όπως είπαμε και στην προηγούμενη παράγραφο ο χρήστης θα εισάγει το query στο σύστημα. Αυτό όμως δεν είναι αρκετό. Για να φτιαχτεί η εικόνα το σύστημα πρέπει να έχει τις εικόνες που αντιστοιχούν στον κάθε χαρακτήρα από τους οποίους αποτελείται το query. Αυτό επίσης γίνεται από τον χρήστη. Επιπλέον το σύστημα πρέπει να ανιχνεύσει τις κενές στήλες ανάμεσα στα γράμματα. Για το λόγο αυτό ο χρήστης θα πρέπει να δώσει στο σύστημα μια συλλογή από εικόνες λέξεων. Συνοπτικά ένα τέτοιο σύστημα αποτελείται (θα έπρεπε να αποτελείται) από τα παρακάτω στάδια

Στο πρώτο στάδιο το σύστημα επεξεργάζεται τα δεδομένα του χρήστη ,στο δεύτερο γίνεται η σύνθεση της εικόνας και στο τελευταίο (θα έπρεπε να) γίνεται μια τελική επεξεργασία στο image query. Υλοποιήσαμε το σύστημα σε Python χρησιμοποιώντας την βιβλιοθήκη Qt για την σχεδίαση της γραφικής διεπαφής ,το opencv για την επεξεργασία εικόνας και την numpy για τον υπολογισμό της ευθείας της γραμμικής παλινδρόμησης με την μέθοδο των ελαχίστων τετραγώνων.

3.2 ΑΝΑΛΥΣΗ ΕΠΙΜΕΡΟΥΣ ΣΤΑΔΙΩΝ

Το πρώτο στάδιο (η είσοδος του χρήστη) αποτελείται από τα παρακάτω επιμέρους στάδια. Η διαδικασία που αναγράφεται στα κουτάκια που έχουν πορτοκαλί χρώμα εκτελείται από τον υπολογιστή ενώ αυτά που είναι με θαλασσί εκτελούνται από τον χρήστη.



Σχήμα 3.1: Πρώτο Στάδιο

Ακολουθεί μια αναλυτική περιγραφή των σταδίων:

i. User Query : Στο στάδιο αυτό ο χρήστης πληκτρολογεί το query.

ii. Get Distinct chars : Υποθέτουμε ότι μόνο ένα glyph μπορεί να αντιστοιχιστεί στον ίδιο χαρακτήρα. Η διαδικασία αυτή δηλαδή ανιχνεύει τους μοναδικούς χαρακτήρες που περιέχει το αλφαριθμητικό query.

iii. Cut Areas : Στον χρήστη δίνεται η επιλογή είτε να κόψει περιοχές είτε χαρακτήρες. Στην περίπτωση όπου θέλουμε να κόψουμε χαρακτήρες από μια σελίδα που έχει σκαναριστεί στραβά συνίσταται να ακολουθηθεί η πρώτη επιλογή.

Η διαδικασία επιλογής των χαρακτήρων γίνεται ως εξής :

-Επιλέγει ο χρήστης από ένα μενού εάν θέλει να κόψει περιοχές ή μεμονωμένους χαρακτήρες.

-Κάνει διπλό κλικ είτε στην άνω δεξιά ή στην άνω αριστερά γωνία.

-Ξανακάνει διπλό κλικ στην κάτω αριστερή ή δεξιά γωνία.

Εάν έχει ο χρήστης έχει επιλέξει να κόψει περιοχές τότε εφαρμόζεται στην περιοχή που έκοψε ο χρήστης μια διαδικασία διόρθωσης περιστροφής (skew correction).

iv. Binarize: Εδώ η εικόνα υφίσταται μια διαδικασία κατωφλίωσης , δηλαδή η τελική εικόνα θα περιέχει μόνο άσπρο και μαύρο. Χρησιμοποιείται η μέθοδος Otzu.

v. Correct Skew: Είναι αρκετά πιθανό κάποιες σελίδες από τις εικόνες εγγράφων να έχουν σκαναριστεί στραβά. Η διόρθωση της περιστροφής βγάζει καλύτερα αποτελέσματα όταν εφαρμοσθεί σε μια εικόνα που περιέχει αντί σε εικόνα-γράμμα. Για να γίνει η διόρθωση περιστροφής πρέπει να ανιχνεύσουμε τις συντεταγμένες των μαύρων pixel κάθε στήλης .**[2]**

   Μετά με βάση αυτά τα σημεία μέσω της διαδικασίας της απλής γραμμικής παλινδρόμησης θέλουμε να "μάθουμε" μια ευθεία που να περνά από τις συντεταγμένες αυτές.

   Η ευθεία φυσικά είναι της μορφής y=ax+b όπου a ονομάζεται slope και το b intercept.

   Για να διορθωθεί η κλίση αρκεί να περιστρέψουμε την εικόνα κατά arctan(a) . Η συνάρτηση που εκτελεί αυτήν την διαδικασία είναι η CorrectSkew

vi. Estimate Letter Space : Για να γίνει η σύνθεση χαρακτήρων πρέπει να βρούμε την απόσταση μεταξύ των χαρακτήρων στην εικόνα του εγγράφου. Ο πιο απλός τρόπος είναι να υπολογίσουμε τον μέσο όρο των αριθμών τον λευκών στηλών μεταξύ των γραμμάτων σε ένα μικρό σύνολο εικόνων που δίνει ο χρήστης. Σε αυτές τις εικόνες όμως ο χρήστης έχει αφαιρέσει τους ascenders και descenders (ουρές άνω και κάτω από τον χαρακτήρα)

Εικόνα 3.1: Εικόνες για ανίχνευση απόστασης γραμμάτων

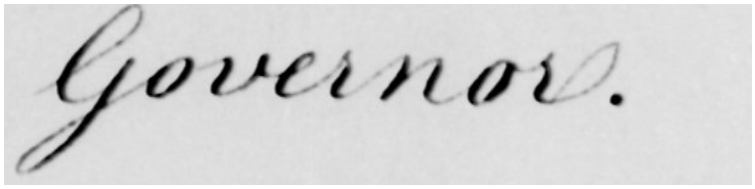
vii. Cut char-n: Δυστυχώς στην χειρόγραφη συλλογή του Washington παρατηρούμε ότι ο

συγγραφέας έχει την τάση να κολλάει τα γράμματα σε αντίθεση με την συλλογή της Σάμου όπου οι χαρακτήρες μπορούν να απομονωθούν.

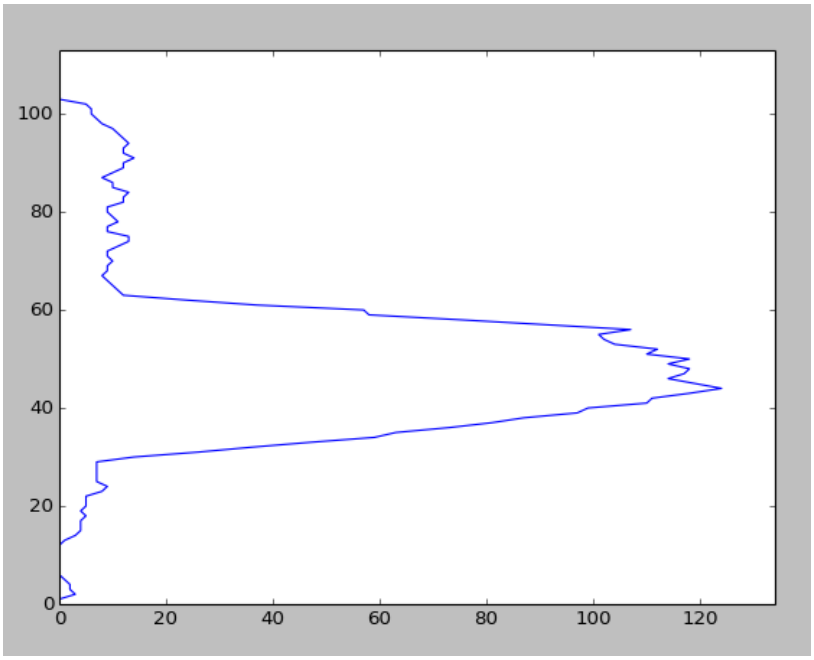
viii. Manual editing: Για να είναι το τελικό image query ομοιόμορφο καλό θα ήταν οι εικόνες χαρακτήρων από τις οποίες αποτελείται το query να έχουν το ίδιο scale. Για το λόγο αυτό το λογισμικό με το οποίο ο χρήστης κόβει εικόνες από το έγγραφο δεν επιτρέπει να γίνει zoom στην εικόνα

ix. Dilate/Erode/Dilate : Μπορεί να υπάρξουν κάποιες περιπτώσεις όπου κάποιοι χαρακτήρες να είναι πιο παχύ από τους υπόλοιπους ή έχουν κάποια περιστροφή. Σε αυτή την περίπτωση θα πρέπει ο χρήστης χειροκίνητα να εφαρμόσει μια διαδικασία λέπτυνσης (erosion) ή πάχυνσης (dilation) των γραμμάτων. Μάλιστα δίνεται στον χρήστη η επιλογή να φτιάξει τον δικό του πυρήνα.

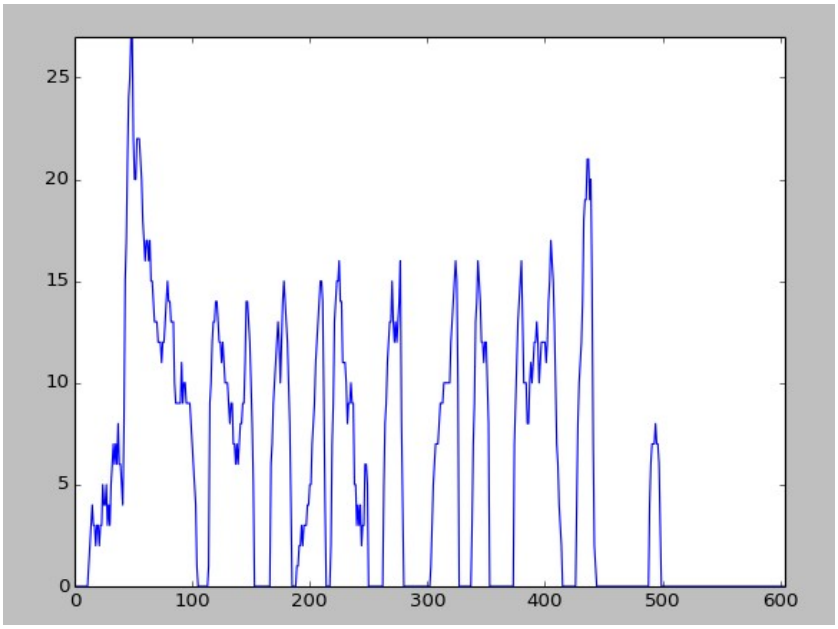
x. Detect Limits : Για να γίνει η σύνθεση χαρακτήρων πρέπει από κάθε εικόνα να βρούμε την περιοχή που περιέχει τα μαύρα pixels. Φτιάχνουμε δηλαδή ένα bounding box.**[1]** Υπολογίζουμε το bounding box με βάση την οριζόντια και κάθετη προβολή της εικόνας. Η οριζόντια προβολή είναι μια "γραφική παράσταση" όπου μας δείχνει το πλήθος των μαύρων pixels σε κάθε γραμμή. Αντίστοιχα η κάθετη προβολή μας λέει πόσα μαύρα pixels υπάρχουν σε κάθε στήλη. Για την ακρίβεια μας ενδιαφέρει η πρώτη και η τελευταία γραμμή που περιέχει έστω και ένα μαύρο pixel. Το ίδιο ισχύει για τις στήλες. Για παράδειγμα η λέξη Governor περιέχεται μεταξύ των γραμμών 2 και 116 και των στηλών 10 έως 505



Εικόνα 3.2: Εικόνα από την χειρόγραφη συλλογή



Σχήμα 3.2: Οριζόντια προβολή της λέξης Governor



Σχήμα 3.3: Κάθετη προβολή της λέξης Governor

Ιδιαίτερα σημαντική είναι η ιδιότητα BaseLine (γραμμή αναφοράς) . Πάνω σε αυτήν την γραμμή θα "κάτσουν" όλοι οι χαρακτήρες του query. Για απλότητα θεωρούμε ότι η γραμμή αναφοράς του query είναι γραμμή αναφοράς του πρώτου χαρακτήρα του query. Επίσης θεωρούμε ότι η γραμμή στην οποία βρίσκεται το χαμηλότερο μαύρο pixel είναι η γραμμή αναφοράς. Δυστυχώς υπάρχουν κάποιοι χαρακτήρες όπως γ,ζ,η,μ,ρ,θ,χ,ψ για τους αυτή η θεώρηση δεν ισχύει. Για αυτούς τους χαρακτήρες η γραμμή αναφοράς είναι λίγο πιο πάνω από την γραμμή με το χαμηλότερο pixel.

Η ανάθεση γίνεται χειροκίνητα μέσα στον κώδικα. Θεωρητικά για τυπωμένα έγγραφα η γραμμή αναφοράς θα είναι ίδια για όλα τους χαρακτήρες με descenders.

Παραθέτουμε το στάδιο δημιουργίας του query

Σχήμα 3.4: Δεύτερο στάδιο σύνθεσης

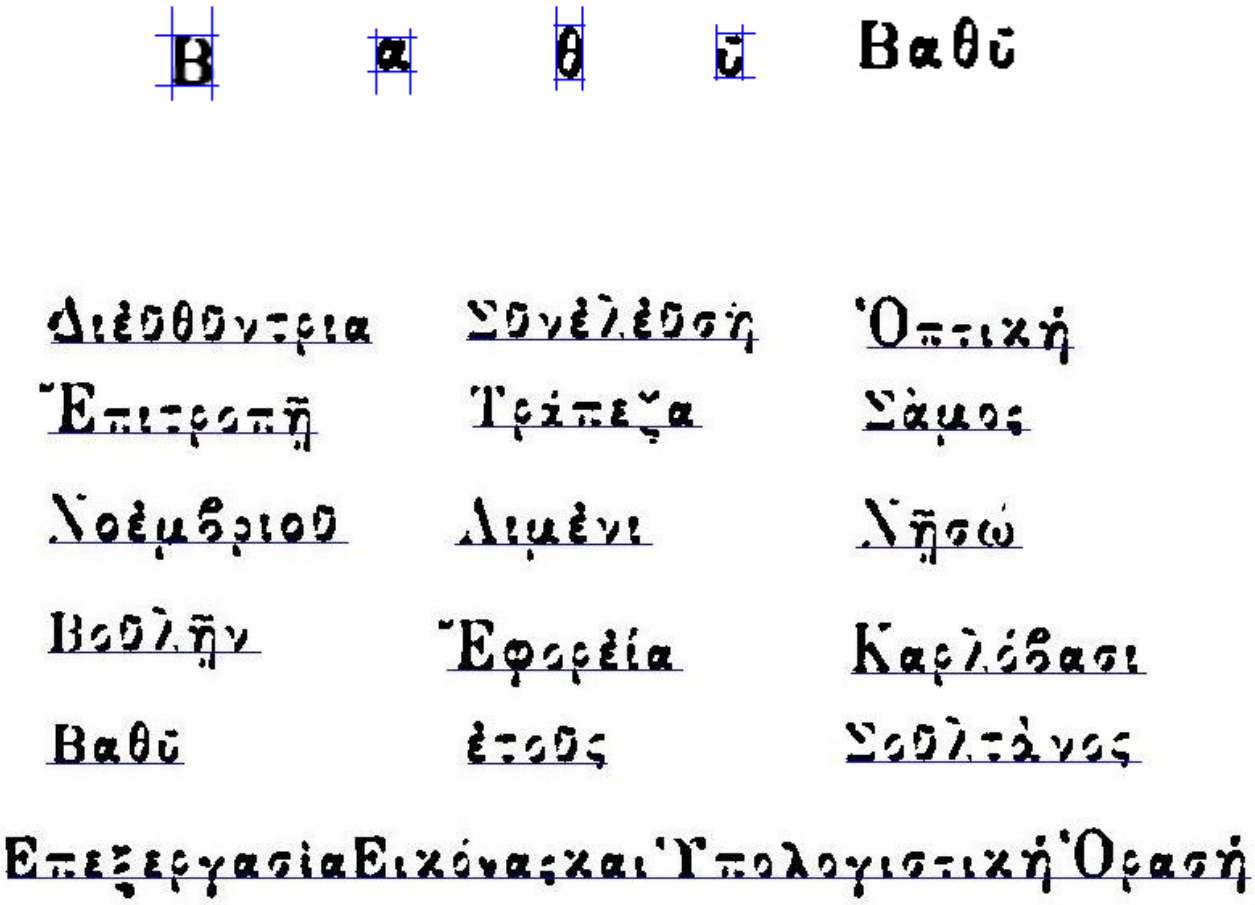
Με βάση τις εικόνες χαρακτήρων που έκοψε ο χρήστης υπολογίζουμε τις διαστάσεις της τελικής εικόνας και κολλάμε τις εικόνες την μια δίπλα στην άλλη (juxtapose). Το πλάτος της εικόνας του query δεν μπορεί να ξεπερνά το άθροισμα των πλατών των μαύρων περιοχών των επιμέρους χαρακτήρων και του letter white space. Με το ίδιο σκεπτικό το ύψος δεν μπορεί να ξεπερνά το διπλάσιο του μεγαλύτερου ύψους των επιμέρους εικόνων.

Ο αλγόριθμος αυτός λειτουργεί πολύ καλά για τυπωμένους χαρακτήρες επειδή αυτοί μπορούν να απομονωθούν πολύ εύκολα. Αντίθετα ίσως είναι αδύνατο χωρίς την μεσολάβηση του χρήση να συνθεθεί με ένα query από χειρόγραφους χαρακτήρες κάνοντας χρήση αυτού του αλγορίθμου. Αφού ανιχνευθεί η γραμμή αναφοράς παίρνουμε το περιεχόμενο του bounding box του κάθε glyph και το κολλάμε κατάλληλα στο προηγούμενο χαρακτήρα λαμβάνοντας υπόψιν και το κενό ανάμεσα στα γράμματα. Για να βρούμε το bounding box κάθε χαρακτήρα βρίσκουμε την στήλη που βρίσκεται το πρώτο και το τελευταίο μαύρο pixel και την γραμμή που βρίσκεται το πρώτο και το τελευταίο μαύρο pixel.

Αλγόριθμος 1

```
Είσοδος  : string query qS
Crate an Empty Image qI
for each distinct character c in query qS:
        assign a glyph g to c
for each character c in query qS:
        gb = contents of the bounding box of g
        Find the vertical shift of gb so that gb.BaseLine=qI.BaseLine
        Juxtapose gb to qI
```

Έστω ότι θέλουμε να συνθέσουμε το ερώτημα Βαθῦ από τις παρακάτω εικονίτσες. Σε κάθε εικονίτσα έχουμε ζωγραφίσει το bounding box. Ο αλγόριθμος παίρνει το περιεχόμενο του bounding box κάθε χαρακτήρα και το αντιγράφει στον πίνακα της εικόνας του query.

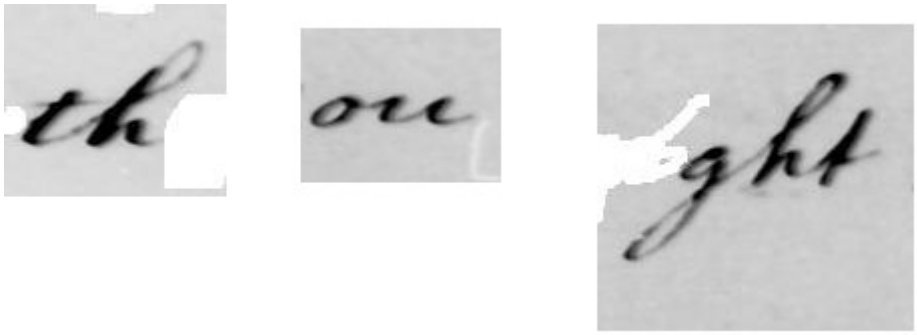


Εικόνα 3.4 : Παραδείγματα σύνθεσης

Όπως αναφέρθηκε πιο πάνω αυτή η μέθοδος λειτουργεί πολύ καλά για τυπωμένους χαρακτήρες. Στους χειρόγραφους αποτυγχάνει πλήρως διότι υπάρχει επικάλυψη μεταξύ των χαρακτήρων όπως στην παρακάτω εικόνα.

Εικόνα 3.5 : Εικόνα από την χειρόγραφη συλλογή

Έστω ότι θέλουμε να σχηματίσουμε την λέξη thought από τα εξής glyphs.



Εικόνα 3.6: Εικόνες των ν-γραμμάτων th ou ght

Έαν εφαρμόσουμε τον αλγόριθμο όπως τον περιγράψαμε πιο πάνω θα πάρουμε το παρακάτω query:

Εικόνα 3.7 : Αποτέλεσμα σύνθεσης

Για το λόγο αυτό εφαρμόζουμε τον δεύτερο τρόπο “σύνθεσης”.

4 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ΑΝΑΖΗΤΗΣΗΣ ΣΕ ΧΕΙΡΟΓΡΑΦΑ

4.1 ΕΙΣΑΓΩΓΗ

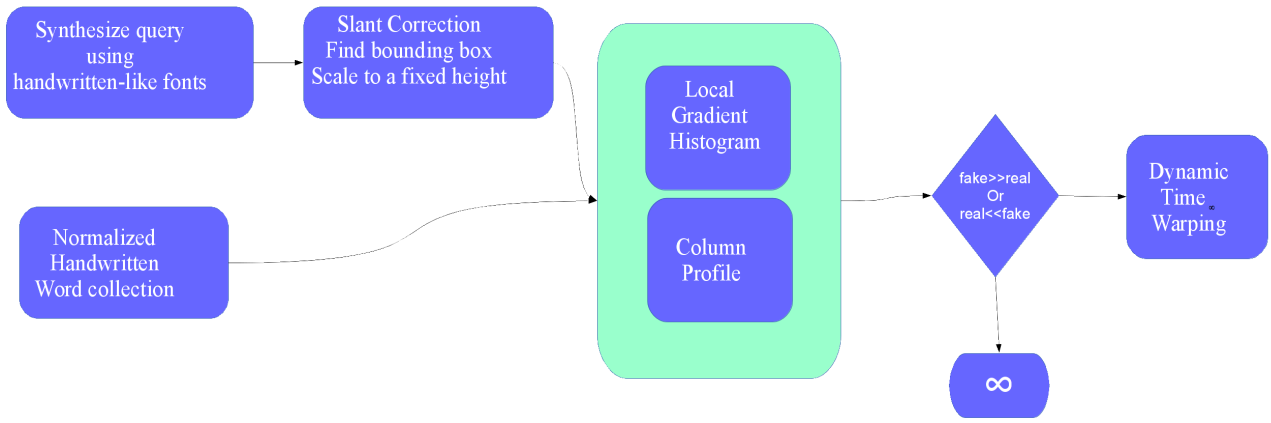
Ένας άλλος τρόπος να προσεγγίσουμε το πρόβλημα είναι οι εξής : Μπορούμε να φτιάξουμε συνθετικές γραμματοσειρές και να τις συγκρίνουμε με χειρόγραφες λέξεις χρησιμοποιώντας τον αλγόριθμο δυναμικής στρέβλωσης χρόνου δοκιμάζοντας είτε column features [9][10] είτε LGH features [7] . Χρησιμοποιήθηκε η έτοιμη βάση IAM Database χειρόγραφων λέξεων από τις επιστολές του George Washington . Οι εικόνες της χειρόγραφης συλλογής είναι κανονικοποιημένες (διόρθωση κλίσης γραμμάτων,περιστροφής λέξης, γραμμής αναφοράς) και έχουν ύψος 120 pixels.

Χρησιμοποιήθηκαν οι παρακάτω γραμματοσειρές και keywords . Τα βήματα της κανονικόποίησης εφαρμόστηκαν ώς εξής : διόρθωση κλίσης γραμμάτων (slant) [5][13] ,whitespace crop, αλλαγή της κλίμακας της εικόνας έτσι ώστε το ύψος της να είναι 120 pixels. Η χειρόγραφη συλλογή περιέί0χε 4985 εικόνες και η σύνθετη 511 εικόνες

Γραμματοσειρές	
Abecedario	Across the road
Adine Kimberg Script	aquafont
Atatürk	Breip
Delphine	Digital Handwriting
Elegante	Exmouth
Freebooter	Ginette
Halohand Letter	Harry Morgan Hand
Homy Script	Jane Austen
Jennifer Lynne	LaurenScript
Lucida Handwriting	Notera
Pecita	Simplesnails

Steve	Jaggetes
Tex Gyre Chorus	Times New Roman
URW Chancery L	Vavont
Windsong	

Keywords		
Regiment	Alexandria	Winchester
Instructions	Sergeant	Virginia
Letters	December	Orders
Parole	Captain	October
Colonel	Washington	Report
Corporal	Lieutenant	Cumberlands
Fort		



Σχήμα 4.1: Διαδικασία σύνθεσης

4.2 Ο ΑΛΓΟΡΙΘΜΟΣ ΔΥΝΑΜΙΚΗΣ ΣΤΡΕΒΛΩΣΗΣ ΧΡΟΝΟΥ (Dynamic Time Warping)

Βασισμένοι στα [11][10] φτιάξαμε ένα σύστημα που συγκρίνει τις συνθετικές λέξεις με τις χειρόγραφες χρησιμοποιώντας τον αλγόριθμο δυναμικής στρέβλωσης χρόνου όπως παρουσιάζεται

στο [10] περιορίζοντας το μονοπάτι στρέβλωσης είτε σε μια ζώνη Sakoe-Chiba [12] είτε σ'ενα παραλληλόγραμο Itakura [12] .Μπορούμε να μειώσουμε το χρόνο σύγκρισης εαν θεωρήσουμε ότι από λέξεις διαφορετικού μήκους βγαίνουν επίσης σειρές διαφορετικού μήκους.. Με άλλα λόγια σειρές που έχουν πολύ διαφορετικό μήκος είναι αρκετά απίθανο να αναπαριστούν τις ίδιες λέξεις. Βέβαια το πόσο διαφέρουν εξαρτάται από την γραμματοσειρά.

Αλγόριθμος 2

```
    def SimpleDynTimeWarp(self,s,t,Band_Width):
        if len(t)>len(s) and len(s)/float(len(t))<0.65:

            return float("inf")
        elif len(s)>len(t) and len(t)/float(len(s))<0.65:

            return float("inf")
        else:

            DTW= [[0 for y in xrange(len(t))] for x in xrange(len(s))] #s rows
t columns

            DTW[0][0]=self.SimpleCost(s[0],t[0])

            for i in range(1,len(s)):
                DTW[i][0]=DTW[i-1][0]+self.SimpleCost(s[i], t[0])

            for j in range(1,len(t)):
                DTW[0][j]=DTW[0][j-1]+self.SimpleCost(s[0], t[j])

            #Sakoe-Chiba band
            #get a fraction of the rows size
            R=int(Band_Width*len(s))

            #Calculate Cost Matrix

            for i in range(2,len(s)):
                for j in range(2,len(t)):
                    if abs(j-i)>=R:
                        DTW[i][j]=float("inf")
                    else:
                        cost=self.SimpleCost(s[i],t[j])
                        DTW[i][j]=cost+min(DTW[i-1][j],DTW[i][j-1],DTW[i-1][j-
1])

            kost=DTW[len(s)-1][len(t)-1]/float(len(s) + len(t)) #Normalize with
the sum of series len

            return kost


    def SimpleCost(self,a,b):
        return (a-b)**2.0
```

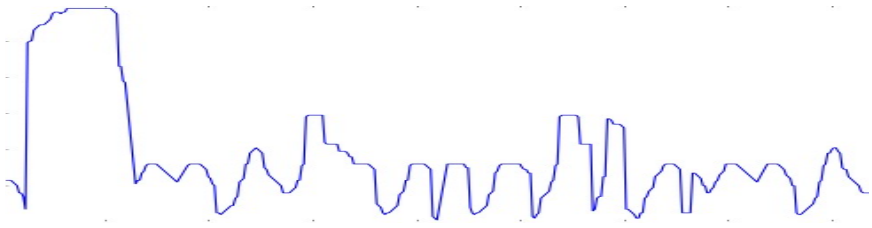
Επίσης χρησιμοποιήσαμε την βιβλιοθήκη mlpy η οποία περιέχει μια πιο γρήγορη έκδοση του αλγορίθμου. Αυτή η βιβλιοθήκη όμως υποστηρίζει σύγκριση μονοδιάστατων σειρών. Σε αυτήν την

περίπτωση χρησιμοποιήσαμε ως feature το top profile της εικόνας.

4.3 Column Features

Χρησιμοποιήσαμε ως features τα background to ink transitions, low profile, upper profile και το κάθετο ιστόγραμμα λέξης [9]. Δηλαδή για κάθε στήλη μετράμε τις εναλλαγές από μαύρο σε άσπρο, την γραμμή του τελευταίου μαύρου pixel την γραμμή του πρώτου μαύρου pixel και τον αριθμό των pixels. Χρησιμοποιήσαμε το τετράγωνο της Ευκλείδιας απόστασης ως μέτρο σύγκρισης αφού πρώτα κανονικοποιήσαμε τις δυο σειρές.

Εικόνα 4.1: συνθετική κανονικοποιημένη εικόνα



Σχήμα 4.2: Top profile

Σε κάθε query ταξινομήσαμε τα αποτελέσματα ως προς αύξουσα σειρά με βάση την απόσταση που έδωσε ο DTW. Ελέξγαμε τα τρία πρώτα αποτελέσματα. Εαν η λέξη που ψάχναμε εμφανίζονταν στην πρώτη θέση το σκορ ήταν 3 βαθμοί , αν εμφανίζονταν στην δεύτερη το σκορ ήταν δύο βαθμοί και τέλος αν εμφανίζονταν στην τρίτη το σκορ ήταν ένας βαθμός. Με άλλα λόγια το σκορ για ένα query παίρνει κάποια από τις τιμές 0,1,2,3,4,5,6

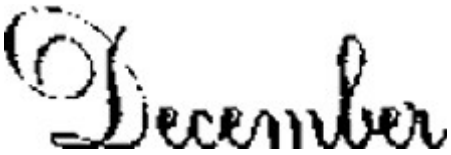
Parameters	Score
Prunning:0.8 Band_Width:10	3,51%
Prunning:0.1 Band Type SC Band_Width:80	11,70%
Prunning:0.9 Band Type SC Band_Width:120	11,34%
Prunning:0.5 Band Type SC Band_Width:200	9,52%
Prunning:0.3 Band Type SC Band_Width:8	5,98%
Prunning:0.1 Band Type SC Band_Width:60	7,86%
Prunning:0.65 Band Type SC Band Width 50	4,02%
Prunning:0.65 Band Type ITK Band Width 10	6,71%
Prunning:0.65 Band Type ITK Band Width 100	0,10%
Prunning:0.65 Band Type SC Band Width 100	3,10%

Πίνακας 4.1 : Top Profile DTW

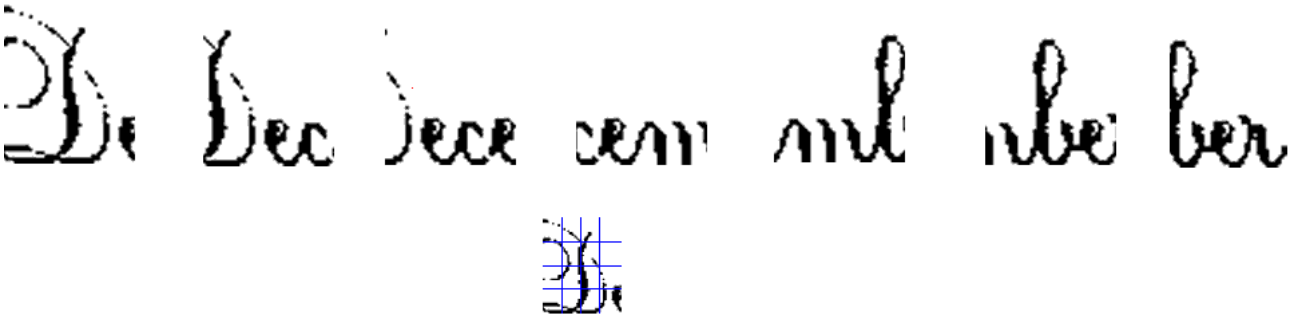
4.4 LGH Features

Με βάση το [7] κατασκευάσαμε έναν αλγόριθμο εξαγωγής χαρακτηριστικών με βάση το ιστόγραμμα κλίσεων. Η διαδικασία είναι πολύ απλή και έχει ως εξής:

Εφαρμόζουμε ένα smoothing φίλτρο στην εικόνα. Παίρνουμε ένα παράθυρο σταθερού πλάτους .Το παράθυρο ,το οποίο έχει ύψος όσο το ύψος της εικόνας και πλάτος w -μια παράμετρος που καθορίζεται από το χρήστη - κινείται πάνω στην εικόνα με κατεύθυνση από αριστερά προς τα δεξιά επειδή η συλλογή εικόνων περιέχει λέξεις στην αγγλική γλώσσα. Εαν η γλώσσα ήταν τα αραβικά τότε το παράθυρο θα ξεκινούσε από τα δεξιά από το τέλος της εικόνας δηλαδή με κατεύθυνση προς τα αριστερά . Επεκτείνουμε την εικόνα κατάλληλα προσθέτοντας στήλες με λευκό στο τέλος της εικόνας έτσι ώστε η νέα εικόνα να χωρέσει ακέραιο αριθμό παραθύρων. Με παρόμοιο τρόπο επεκτείνουμε την εικόνα προσθέτοντας γραμμές με λευκό στο κάτω μέρος της εικόνας έτσι ώστε το πλήθος το γραμμών να διαιρείται ακριβώς με το πλήθος των κελιών. Τα παράθυρα έχουν μια επικάλυψη μεταξύ τους η οποία καθορίζεται από την χρήστη και κυμαίνεται από 0 εώς και 100%.



Εικόνα 4.2: Εικόνα 3.3 : query για την λέξη December



Εικόνα 4.3: Διάσπαση σε παράθυρα ,εφαρμογή πλέγματος

Το κάθε παράθυρο χωρίζεται σε c ισοδιάστατα κελιά. Χωρίζουμε το διάστημα [-π,π] σε n bins. Σε κάθε pixel κάθε κελιού ορίζουμε ένα πλάτος (magnitude) και μια κατεύθυνση (direction) . Στην συνέχεια βρίσκουμε από ποια bins περικλείεται η κατεύθυνση και μετράμε την συνεισφορά του pixels σε αυτά τα bins. Από κάθε κελί προκύπτει ένα διάνυσμα μήκους τόσου όσου και τα bins. Τα

διανύσματα του κάθε κελιού σε κάθε παράθυρο συνενώνονται (concatenation) σε ένα διάνυσμα. Τα στοιχεία αυτού του διανύσματος κανονικοποιούνται ώς προς το άθροισμα των στοιχείων τους. Το διάνυσμα αυτό έχει μήκος τόσο όσο και το γινόμενο του αριθμού των κελιών σε κάθε παράθυρο επί τον αριθμό των bins. Κάθε εικόνα με άλλα λόγια περιγράφεται από ένα σύνολο τέτοιων διανυσμάτων. Το τελικό διάνυσμα για κάθε εικόνα προκύπτει αν ενώσουμε (concatenation) τα διανύσματα όλων των παραθύρων. Το μήκος αυτού του διανύσματος ποικίλλει ανάλογα με το πλάτος της εικόνας.

.

Paramets	**Score**
Window width=64 Window overlap=0.9 Cells=4 Bins=5 SC Band 28 queries	12.92 %
Window width=90 Window overlap=0.75 Cells=2 Bins=6 SC Band 510 queries	0.8 %
Window width=90 Window overlap=0.75 Cells=2 Bins=6 SC Band 490 queries	0.96 %
Window width=50 Window overlap=0 Cells=2 Bins=6 SC Band 490 queries	0.74 %

Πίνακας 4.2 : LGH DTW

5 ΣΥΜΠΕΡΑΣΜΑΤΑ -ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Ο αλγόριθμος που αναπτύχθηκε φαίνεται να δουλεύει πολύ καλά για τυπωμένους χαρακτήρες. Όμως θα πρέπει για κάθε φωνήεν με πολυτονικούς χαρακτήρες θα πρέπει η γραμμή αναφοράς να εισαχθεί από τον χρήστη. Ενδιαφέρον θα ήταν να μελετηθεί η απόδοση της σύνθεσης με μια τεχνική word spotting. Ένα άλλο πράγμα που δεν δοκιμάσαμε να συνθέσουμε είναι αριθμούς ή χαρακτήρες από το πολυτονικό σύστημα όπως πχ ᾷ ῷ. Ακόμη δεν υπάρχει η δυνατότητα σύνθεσης δύο ή και παραπάνω λέξεων στην ίδια εικόνα ή αντιστοίχησης παραπάνω από ένα glyph στον ίδιο χαρακρατήρα.

Η πάρα πολύ χαμηλή απόδοση του συστήματος σύνθεσης χειρόγραφων οφείλεται στην διαφορετική προέλευση (χειρόγραφα - σύνθετα) των εικόνων. Θα μπορούσαμε να εξετάζαμε διαφορετικές τεχνικές keyword image retrieval .

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Leydier, Yann, et al. "Towards an omnilingual word retrieval system for ancient manuscripts." *Pattern Recognition* 42.9 (2009): 2089-2105.
2. Τσακαλίδου Ελίνα Διπλωματική Εργασία "Κανονικοποίηση λέξεων" Πανεπιστήμιο Αιγαίου 2014
3. Guyon, Isabelle. "Handwriting Synthesis From Handwritten Glyphs." *In Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition*. 1996.
4. Wang, Jue, et al. "Learning-based cursive handwriting synthesis." *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*. IEEE, 2002
5. de Zeeuw, Frank. "Slant Correction Using Histograms." *Undergraduate Thesis. http://www. ai. rug. nl/~ axel/teaching/bachelorprojects/zeeuw_slant correction. Pdf* (2006).
6. Lin, Zhouchen, and Liang Wan. "Style-preserving English handwriting synthesis." *Pattern Recognition* 40.7 (2007): 2097-2109.
7. Rodrıguez, José A., and Florent Perronnin. "Local gradient histogram features for word spotting in unconstrained handwritten documents." *Int. Conf. on Frontiers in Handwriting Recognition*. 2008.
8. Rodriguez-Serrano, Jose A., and Florent Perronnin. "Synthesizing queries for handwritten word image retrieval." *Pattern Recognition* 45.9 (2012): 3270-3276.
9. Rath, Tony M., and Raghavan Manmatha. "Word image matching using dynamic time warping." *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 2. IEEE, 2003.
10. Rath, Tony M., and Rudrapatna Manmatha. "Word spotting for historical documents." *International Journal of Document Analysis and Recognition (IJDAR)* 9.2-4 (2007): 139-152.
11. Sakoe, Hiroaki, and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition." *Acoustics, Speech and Signal Processing, IEEE Transactions on* 26.1 (1978): 43-49.
12. Müller, Meinard. "Dynamic time warping." *Information retrieval for music and motion* (2007): 69-84.
13. Vinciarelli, Alessandro, and Juergen Luettin. "A new normalization technique for cursive

handwritten words." Pattern Recognition Letters 22.9 (2001): 1043-1050.