Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Ψηφιακή Επεξεργασία Εικόνας

Εργασία 2: Οπτική αναγνώριση χαρακτήρων

Ευσεβεία Νεστοροπούλου

ΑΕΜ: 9703

nestoropo@ece.auth.gr

Εαρινό Εξάμηνο 2023

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Στην παρούσα εργασία, στο μάθημα Ψηφιακή Επεξεργασία Εικόνας, μας δίνεται μία αρχικά περιστραμένη εικόνα (text1.png), στην οποία καλούμαστε να αναγνωρίσουμε την γωνία περιστροφής της και να την περιστρέψουμε [Βήμα 1ο]. Στην συνέχεια, πρέπει να γίνει κατάλληλη επεξεργασία ώστε να αναγνωρίσουμε τις γραμμές, τις λέξεις και τα γράμματα της εικόνας-κειμένου, έπειτα να βρούμε τα περιγράμματα για κάθε γράμμα και τελικά να αναγνωρίσουμε τους χαρακτήρες [Βήμα 2ο]. Τέλος, καλούμαστε να φριάξουμε ένα νευρωνικό δίκτυο τύπου knn, το οποίο θα δέχεται ως είσοδο τα γράμματα και θα αναγνωρίζει αυτόματα τους χαρακτήρες του κειμένου [Βήμα 3ο].

Κεφάλαιο 2

Προεπεξεργασία και Περιστροφή

Οι παρακάτω συναρτήσεις υλοποιήθηκαν στο αρχείο rotate.py

2.1 preprocessImage

Δέχεται ως είσοδο την αρχική εικόνα, και πραγματοποιεί ορισμένες μετρατροπές ώστε να έχουμε την θεμιτή μορφή. Πιο συγκεκριμένα, χρησιμοποιούνται οι συναρτήσεις:

1. cv2.cvtColor για μετατοπή σε Grayscale
2. cv2.morphologyEx με MORPH_GRADIENT για να τονίσουμε τα περιγράμματα των χαρακτήρων
3. cv2.threshold για να τονίσουμε περεταίρω τους χαρακτήρες

4. cv2.morphologyEx με MORPH_CLOSE για να ¨ενώσουμε¨ τις γραμμές και να διευκολύνουμε τον εντοπισμό τους.

2.2 findRotationAngle

Δέχεται ως είσοδο την επεξεργασμένη εικόνα καθώς και την αρχική εικόνα. Αρχικά, υπολογίζουμε το διακριτό μετασχηματισμό Fourier της εικόνας και μεταφέρουμε την μηδενική συνιστώσα στο κέντρο. Έπειτα, υπολογίζουμε το magnitude spectrum του DFT. Πάνω σε αυτήν την εικόνα εφαρμόζουμε τις συναρτήσεις cv2.Canny και cv2.HoughLinesP για να εντοπίσουμε τις γραμμές του spectrum. Για κάθε γραμμή που εντοπίστηκε, ελέγχουμε τις συντεταγμένες της. Αν βρίσκεται κοντά στο κέντρο, λόγω του έντονου θορύβου, την αφαιρούμε ώστε να μη παρεμβάλει στους μετέπειτα υπολογισμούς. Τέλος, υπολογίζουμε την κλίση της κάθε ευθείας και, χρησιμοποιώντας τον μέσο όρο τους, υπολογίζουμε την γωνία περιστροφής.

2.3 serialSearch

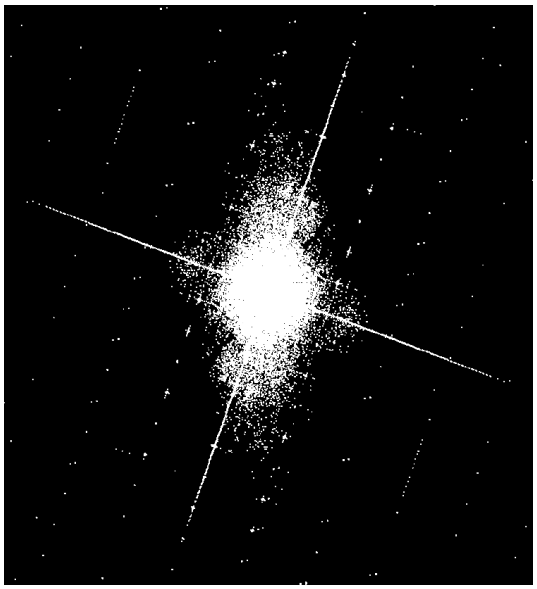
Δέχεται ως είσοδο την επεξεργασμένη εικόνα και την γωνία, που έχει υπολογιστεί από την προηγούμενη συνάρτηση. Για κάθε γωνία σε ένα εύρος ± 10 μοιρών από την δοσμένη γωνία, περιστρέφει την εικόνα, και, αφού ξαναυπολογίσουμε το magnitude spectrum του DFT, βρίσκουμε την κάθετη προβολή της φωτεινότητας. Έπειτα, συγκρίνοντας τα αποτελέσματα, βρίσκουμε την βέλτιστη γωνία περιστροφής. Τέλος, υπολογίζουμε ενα weighted average μεταξύ των δύο γωνιών και επιστρέφουμε την τελική γωνία περιστροφής.

2.4 rotateImage

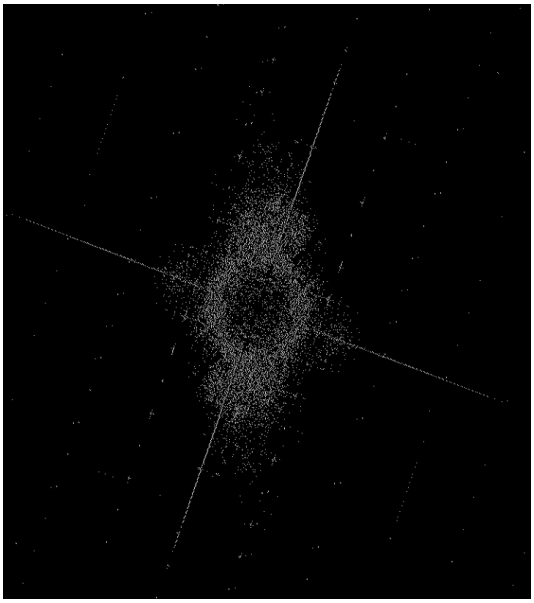
Δέχεται ως είσοδο την εικόνα και την γωνία περιστροφής. Η συγκρεκρίμενη συνάρτηση υλοποιεί την περιστροφή της δοσμένης εικόνας, χρησιμοποιώντας τις συναρτήσεις της openCV: getRotationMatrix2D και warpAffine.

2.5 Αποτελέσματα

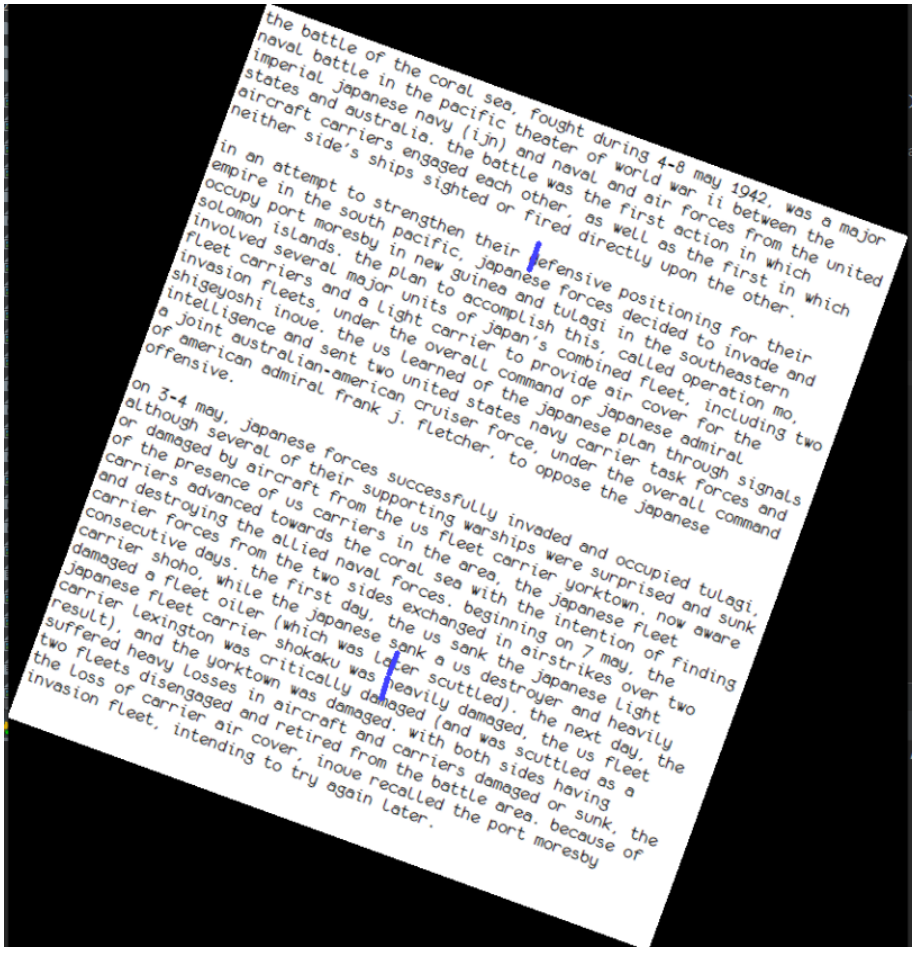
Χρησιμοποιώ την εικόνα text1.png περιστραμένη κατά 20 μοίρες προς τα δεξιά ως παράδειγμα. Η γωνία που υπολογίστηκε από την findRotationAngle είναι 19.2 μοίρες. Η γωνία που υπολογίστηκε από την serialSearch είναι 20 μοίρες. Η τελική γωνία περιστροφής είναι 19 μοίρες.



(α') Το magnitude spectrum μετά την cv2.threshold



(β') Η εικόνα μετά την cv2.Canny



Σχήμα 2.2: Οι εντοπισμένες γραμμές

Πραγματική	DFT Γωνία	Με Serial Search	Τελική Γωνία
-20	-22	-20	-20
-30	-39	-30	-31
20	19	20	19

Ενδεικτικές τιμές, όπως προέκυψαν από τον κώδικα του αρχείου rotate.py.

2.6 Παρατηρήσεις

Ο υλοποιημένος κώδικας μπορεί να εφαρμοστεί και να παράξει ορθά αποτελέσματα για τις εικόνες text1.png και text2.png, σε όποια κλίση κι αν είναι περιστραμένες. Ωστόσο, λόγω της συνεχούς αλλάγης του πρότυπου κειμένου και έλλειψης χρόνου, δεν πρόλαβα να διορθώσω τον κώδικα, ώστε να μπορεί να εφαρμοστεί στις εικόνες text1_v2.png και text1_v3.png.

Κεφάλαιο 3

Περιγραφή περιγράμματος

Οι παρακάτω συναρτήσεις έχουν υλοποιηθεί στο αρχείο text_process.py.

3.1 preprocessText

Όπως ζητείται από την εκφώνηση, πρέπει να εφαρμοστούν κάποιοι μετασχηματισμοί στην εικόνα. Χρησιμοποιούνται οι εξής συναρτήσεις:

1. cv2.cvtColor για μετατοπή σε Grayscale
2. cv2.threshold για να τονίσουμε τους χαρακτήρες
3. cv2.bitwise_not για να κάνουμε invert την εικόνα
4. cv2.dilate για να ¨παχύνουμε¨ τα γράμματα της εικόνας κα να την αφαιρέσουμε από την αρχική
5. cv2.subtract
6. cv2.bitwise_not
7. cv2.erode για να ¨λεπτύνουμε¨ τα γράμματα

3.2 getContour

Δέχεται ως είσοδο την αρχική και την επεξαργασμένη εικόνα και επιστρέφει τα περιγράμματα που εντόπισε, χρησιμοποιώντας την cv2.findContours. Έπειτα, τα κατηγοριοποεί σε εσωτερικό και εξωτερικό περίγραμμα, υπολογίζει την complex μορφή τους (που θα χρειαστεί στο επόμενο βήμα) και τα σχεδιάζει πάνω στην εικόνα που του δόθηκε.

(α΄) Η preprocessed εικόνα

(β΄) Η εικόνα με τα contours

(α΄) Η preprocessed εικόνα

(β΄) Η εικόνα με τα contours

(α΄) Η preprocessed εικόνα

(β΄) Η εικόνα με τα contours

(α΄) Η preprocessed εικόνα

(β΄) Η εικόνα με τα contours

3.3 getDFT

Δέχεται ως είσοδο ένα array και την μεταβλητή inner, που υποδηλώνει αν υπάρχει ένα ή περισσότερα περιγράμματα. Στη συνέχεια, υπολογίζουμε τον DFT της ακολουθίας και ακολουθούμε την διαδικασία που επιτάσει η εκφώνηση ώστε να καταλήξουμε στο επιθυμητό τετραγωνικό σφάλμα των περιγραφέων των περιγραμμάτων, όπως δείχνει και η σχέση:

$$d(R_1, R_2) = \sum (R_1[i] - R_2[i])^2$$

Για τον βέλτιστο εντοπισμό των χαρακτήρων, οι περιγραφείς πρέπει να έχουν το ίδιο μήκος. Τέλος, όσο μικρότερο είναι το τετραγωνικό σφάλμα, τόσο πιο ισχυρή η υπόθεση ότι οι δύο χαρακτήρες αντιπροσωπεύουν το ίδιο γράμμα.

3.4 compareDFT

Δέχεται ως είσοδο τα δύο περιγράμματα προς μελέτη και την ένδειξη inner, που υποδηλώνει αν υπάρχουν ένα ή περισσότερα περιγράμματα. Για κάθε στοιχείο του προς μελέτη περιγράμματος, εφαρμόζουμε την συνάρτηση np.allclose, για βρούμε τον χαρακτήρα με την μικρότερη απόκληση. Τέλος, επιστρέφει το γράμμα που εντόπισε ως το βέλτιστο.

3.5 Παρατηρήσεις

Ο κώδικας εφαρμόζεται μόνο στις 4 εικόνες των γραμμάτων προς μελέτη a, e, f, l. Δεν έχει ολοκληρωθεί η υλοποίηση του για αυτό και δεν είναι σε θέση να παράξει σωστά αποτελέσματα.

Κεφάλαιο 4

Αναγνώση χαρακτήρων

Οι παρακάτω συναρτήσεις υλοποιήθηκαν στο αρχείο detect.py με βάση το αρχείο text1_v2.png.

4.1 preprocessImage

Υλοποιεί τους απαραίτητους μετασχηματισμούς στην αρχική εικόνα ώστε να διευκολυνθούν οι μετέπειτα ανιχνεύσεις. Παρόμοια με την αντίστοιχη συνάρτηση στο αρχείο rotate.py.

4.2 preprocessText

Υλοποιεί τους απαραίτητους μετασχηματισμούς στην αρχική εικόνα ώστε να διευκολυνθούν οι μετέπειτα ανιχνεύσεις και να ικανοποιεί τις απαιτήσεις της εκφώνησης. Παρόμοια με την αντίστοιχη συνάρτηση στο αρχείο text_process.py.

4.3 detectLines, detectWords, detectLetters

Συναρτήσεις οι οποίες, ακολουθώντας παρόμοια διαδικασία, εντοπίζουν αντίστοιχα τις γραμμές, τις λέξεις και τα γράμματα του κειμένου. Η συνάρτηση detectLines, υπολογίζει την κάθετη προβολή της φωτεινότητας και χρησιμοποιώντας την συνάρτηση

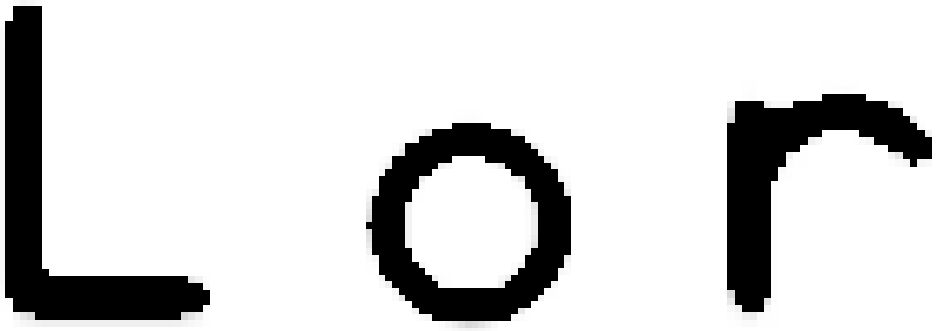
find_peaks εντοπίζει τα σημέια όπου υπάρχει έντονη μεταβολή της φωτεινότητας. Έπειτα, χωρίζει την εικόνα στις γραμμές που εντόπισε και επιστρέφει τις συντεταγμένες τους.

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam quis risus bibendum,
```

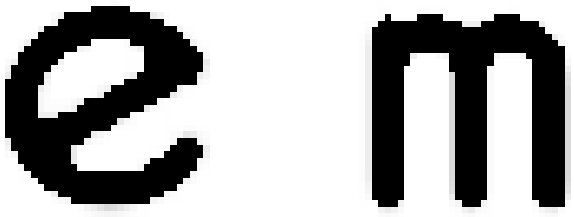
Σχήμα 4.1: Η πρώτη γραμμή του κειμένου

Αντίστοιχα, η detectLetters υπολογίζει την οριζόντια προβολή της φωτεινότητας και χρησιμοποιώντας την συνάρτηση find_peaks εντοπίζει τα σημεία όπου υπάρχει έντονη μεταβολή της φωτεινότητας. Έπειτα, χωρίζει την εικόνα στις γραμμές που εντόπισε και επιστρέφει τις συντεταγμένες τους.

Σχήμα 4.2: Η πρώτη λέξη του κειμένου



(α') Το 1ο γράμμα (β') Το 2ο γράμμα (γ') Το 3ο γράμμα



(α') Το 4ο γράμμα (β') Το 5ο γράμμα

4.4 returnCharacters

Η συγκεκριμένη συνάρτηση ανοίγει το αρχείο, στο path που έχει δοθεί, και αποθηκεύει τους χαρακτήρες του κειμένου ώστε να γίνει η σύγκριση με τα αποτελέσματα του knn. Δεν αποθηκεύει τα κενά και τις νέες γραμμές ως χαρακτήρες. Ενδεικτικά, οι αρχικοί χαρακτήρες του κειμένου:

'L', 'o', 'r', 'e', 'm', 'i', 'p', 's', 'u', 'm', 'd', 'o', 'l', 'o', 'r', 's', 'i', 't', 'a', 'm', 'e', 't', ',', 'c', 'o', 'n', 's', 'e', 'c', 't', 'e', 't', 'u', 'r', 'a', 'd', 'i', 'p', 'i', 's', 'c', 'i', 'n', 'g', 'e', 'l', 'i', 't', '.', 'E', 't', 'i', 'a', 'm', 'q', 'u', 'i', 's', 'r', 'i', 's', 'u', 's', 'b', 'i'

4.5 Παρατηρήσεις

Λόγω έλλειψης χρόνου, δεν υλοποιήθηκε το σύστημα ταξινόμησης χαρακτήρων με βάση τα εντοπισμένα περιγράμματα. Ωστόσο, έχει υλοποιηθεί knn με βάση τις εικόνες των γραμμάτων που έχουν εντοπιστεί από το κείμενο text1_v2.png. Αρχικά, έχουν αναγνωριστεί οι χαρακτήρες του κειμένου με την βοήθεια της συνάρτησης returnCharacters. Έπειτα, εχουν εντοπιστεί σωστά οι γραμμές και τα γράμματα της εικόνας text1_v2.png, έχει δημιουργηθεί dataset από αυτά, έχουν χωριστεί σε train set και test set, όπως καθορίζει η εκφώνηση, και έχει υλοποιηθεί η κατάλληλη εκπαίδευση. Η ακρίβεια του συγκεκριμένου knn είναι 0.3.

Πραγματικός Χαρακτήρας	Εντοπισμένος Χαρακτήρας
i	t
w	N
c	e
d	d