

Ψηφιακή Επεξεργασία Εικόνας

2^η Εργασία Μαθήματος 2022 - 2023

Αλεξόπουλος Δημήτριος ΑΕΜ 10091

aadimitri@ece.auth.gr

Περιεχόμενα

1	Εισαγωγή: Οπτική αναγνώριση χαρακτήρων	4
2	Προεπεξεργασία και Εντοπισμός Λέξεων	5
2.1	find_rotation_angle	5
2.2	rotate_image	6
3	Περιγραφή Περιγράμματος	7
3.1	get_contour	7
3.2	get_descriptor	10
4	Ανάγνωση Χαρακτήρων - Σύστημα Ταξινόμησης	11
4.1	get_dataset	11
4.2	divide_into_classes	12
4.3	form_dataset	13
4.4	train_test	13
4.5	read_text	14
5	Αποτελέσματα - Σχολιασμοί	16
	Βιβλιογραφία	19

Κατάλογος Σχημάτων

1.1	Εικόνα προς εκπαίδευση	4
1.2	Εικόνα προς ανάγνωση	4
2.1	Λογάριθμος του μέτρου του DFT	5
3.1	Δοκιμαστική εικόνα για εύρεση περιγραμμάτων	7
3.2	Original character	8
3.3	Resized and blurred	8
3.4	Subtracted the dilated	8
3.5	Thinned	8
3.6	Περιγράμματα του a	9
3.7	Περίγραμμα του f	9
3.8	Περίγραμμα του l	9
3.9	Περίγραμμα του e	9
4.1	Διαχωρισμός γραμμής του κειμένου	11
4.2	Θολωμένη γραμμή του κειμένου	12
4.3	Διαχωρισμός λέξης του κειμένου	12
5.1	Πρωτότυπο κείμενο	16
5.2	Αναγνωσμένο	16

1 Εισαγωγή: Οπτική αναγνώριση χαρακτήρων

Αντικείμενο της παρούσας εργασίας είναι η αυτόματη, οπτική αναγνώριση χαρακτήρων. Συγκεχριμένα, θα μελετήσουμε μία εικόνα η οποία περιέχει κείμενο και θα εντοπίσουμε και θα τυπώσουμε τους χαρακτήρες που το αποτελούν.

Η εικόνα αυτή στην γενικότερη περίπτωση μπορεί να μην βρίσκεται στην κατάληξη μορφής προκειμένου να γίνει η οπτική αναγνώριση χαρακτήρων. Χαρακτηριστικό παράδειγμα αποτελεί μια σκαναρισμένη εικόνα κειμένου, η οποία πιθανόν να είναι ελαφρά περιεστραμμένη ή να περιέχει διαφορετικά λέξη. Για τον λόγο αυτό πριν από την διαδικασία περιγραφής κι ανάγνωσης των χαρακτήρων απαιτείται η προεπεξεργασία της εικόνας.

Πιο αναλυτικά, τα βασικά βήματα για την οπτική αναγνώριση των χαρακτήρων της είναι:

- η προεπεξεργασία της εικόνας ώστε να είναι κατάλληλη για επεξεργασία και να εντοπίσουν λέξεις
 - η περιγραφή των χαρακτήρων με χρήση του διακριτού μετασχηματισμού Fourier
 - η ανάγνωση της φωτογραφίας, εκπαιδεύοντας και αξιολογώντας ενα σύστημα ταξινόμησης

Etiam quis risus bibendum,
venenatis nunc sed. Ametuntur ante. Volumen tanta ligula semper nibus sollicitudin
tempor et in arcu. Donec blandit ligula ipsum, vel pulvinar ligula maximus a.
Quisque et orci cursus, egestas dictum et, efficitur metus. Cras ultricies odio ex,
sed posuere. Lumen dictum et, Sed facilis, ipsum et semper sollicitudin, nulla
luctus. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea commodo tortor. Fusce facilisis fermentum augue egest tueri imperdiet.
Consequuntur enim ullamcorper, vel feugiat diam consetetur et, Fusce efficitur
consonant, senectus et, sclerótica. Sed eu natus si. Aliquam eleifend convallis viverra

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 J J K K W W Y Y Z Z J J K K W W Y Y Z Z
Nullam ac sagittis velit. Aenean velit risus. *Lacinia eu sapien sed, consequat luctus sem.*
Cras mollis imperdunt fermentum. Quisque dapibus elit libero, eget semper pulvinar. nec. Donec ac aliquam justo, ut rutrum felis. Donec condimentum finibus ipsum, nec auctor
risus sodales vel. Donec viverra tempus ultramcorper. Nunc quis suscipit elit, lobortis finibus
ipsum. Quisque sed eleifend erat. Maecenas sed augue elementum, rhoncus et vel, malesuada
sepien. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia
curae; Aenean sollicitudin. *Curit sit amet enim malesuada, et feugiat turpis erat.* Nullam
nisi tortor, imperdunt nec turpis mollis, aliquam venenatis nunc. *Nunc ut erat velit.* Etiam
nisi tortor, imperdunt nec turpis mollis, aliquam venenatis nunc. *Nunc ut erat velit.* Etiam

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0] J K K W w Y Z Z] J K K W w Y Z Z
Vestibulum eu augue eugue. Nullam vestibulum ullamcorper velit nec pharetra. Proin tincidunt
Lacus vitae magna pretium consequet. Aenean vel purus velut augue laoreet. Libertis et diam.
Donec cursus, sapien vel vehicula ornare. Leo velit viverra eros, sit amet pellentesque est augue
non velit. Praesent feugiat sem ac tincidunt tristique. Sed commodo feugiat velit, quis augue
Lem fermentum ac. Nullam mettis sclerisque quam a rhoncus. Quisque facilisis Lacus quis neque
euismod vehicula. Nullam quis eleifend libero.

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0] j K k W w Y y Z z] j K k W w Y y Z z
Fusce et molestie nullia, sutm amet gravida eros. Sed ut lorem lobortis, posuere ante sit amet, mattis arcu. Etiam rutrum molestie faucibus. Nunc et vel utm velit, auctor rhoncus velit. Fusce venenatis enim, a molestie metus facilisis id. Fusce suscipit est ex, Luctus dictum nibh mollis at. Donec imperdiet massa consetetur. Donec et lorem quae tincidunt vulputate. Duis sit amet Lacinia nisi. Proin in magna at eros vestibulum placerat. Phaselus non condimentum purus, eget tristique diam. Etiam vel convallis purus.

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0] j K k W w Y y Z z J j K k W w Y y Z z
Non Lacinia pretium consecetur. In placerat dulit et tector tempor porta. Maecenas eu neque euismod, bibendum lorem vel, consecetur orci. Sed velut mauris, efficitur non eleifend et, neque tempor vitae nisl. Proin tincidunt vulputate luctus. Integer nec condimentum quam. Donec aliquet sagittis ultrices. Maecenas porta bibendum facilisis. Curabitur efficitur sem eu suscipit euismod

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 J j K k W w Y y Z z J j K k W w Y y Z z

the battle of the coral sea, fought during 4-8 may 1942, was a major naval battle in the pacific theater of world war ii between the imperial japanese navy (ijn) and naval and air forces from the united states and australia. the battle was the first action in which aircraft carriers engaged each other, as well as the first in which neither side's ships sighted or fired directly upon the other.

in an attempt to strengthen their defensive positioning for their empire in the south pacific, japanese forces decided to invade and occupy port moresby in new guinea and tulagi in the southeastern solomon islands. the plan to accomplish this, called operation mo, involved several major units of japan's combined fleet, including two fleet carriers and a light carrier to provide air cover for the invasion fleets, under the overall command of japanese admiral shigeyoshi inoue. the us learned of the japanese plan through signals intelligence and sent two united states navy carrier task forces and a joint australian-american cruiser force, under the overall command of american admiral frank j. fletcher, to oppose the japanese offensive.

on 3-4 may, japanese forces successfully invaded and occupied tulagi, although several of their supporting warships were surprised and sunk or damaged by aircraft from the us fleet carrier yorktown, now aware of the presence of us carriers in the area, the japanese fleet carriers advanced towards the coral sea with the intention of finding and destroying the allied naval forces. beginning on 7 may, the carrier forces from the two sides exchanged in airstrikes over two consecutive days. the first day, the us sank the japanese light carrier shoh, while the japanese sank a us destroyer and heavily damaged a fleet oiler (which was later scuttled). the next day, the japanese fleet carrier shokaku was heavily damaged, the us fleet carrier lexington was critically damaged (and was scuttled as a result), and the yorktown was damaged, with both sides having suffered heavy losses in aircraft and carriers damaged or sunk, the two fleets disengaged and retired from the battle area. because of the loss of carrier air cover, iron recallled the port moresby invasion fleet, intending to try again later.

Σχήμα 1.1: Εικόνα προς εκπαίδευση

Σχήμα 1.2: Εικόνα προς ανάγνωση

Παραπάνω βλέπουμε τις δύο φωτογραφίες με τις οποίες θα προχωρήσουμε στην ανάλυσή μας. Η πρώτη φωτογραφία αποτελεί μια εικόνα κειμένου σε υψηλή ανάλυση με μεγάλο και αντιπροσωπευτικό δείγμα χαρακτήρων. Θα χρησιμοποιηθεί για την εκπαίδευση του συστήματος ταξινόμησής μας, το οποίο στη συνέχεια θα αξιολογήσουμε στην ανάγνωση της δεύτερης εικόνας. Αυτή είναι ελαφρώς περιεστραμμένη και χαμηλότερης ανάλυσης, ώστε να προσομοιώσει μια σκαναρισμένη εικόνα κειμένου.

2 Προεπεξεργασία και Εντοπισμός Λέξεων

Το πρώτο βήμα, όπως είπαμε, είναι η προεπεξεργασία της εικόνας ώστε να έρθει σε μορφή κατάλληλη για την αναγνώριση του καθενός χαρακτήρα χωριστά. Η εικόνα ενδεχομένως να έχει υποστεί κάποια περιστροφή, γι' αυτό και πρώτο μας μέλημα είναι να διασφαλίσουμε πως οι γραμμές του κειμένου θα είναι οριζόντιες. Έτσι, θα είναι δυνατός στην συνέχεια ο διαχωρισμός του κειμένου σε γραμμές, λέξεις και τελικά χαρακτήρες.

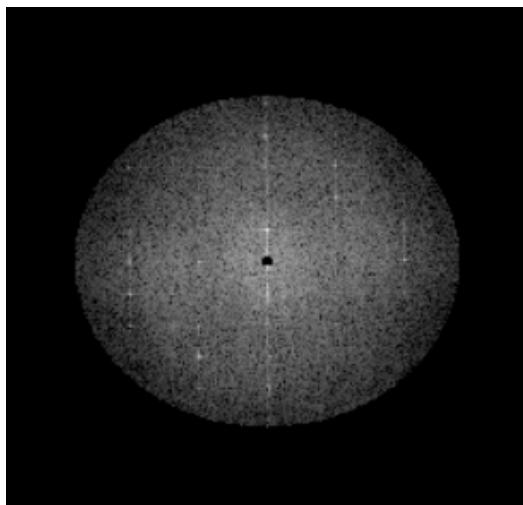
Παρακάτω θα δούμε αναλυτικά τις μεθόδους που εκτελέσαμε για την προεπεξεργασία της εικόνας μας.

2.1 `find_rotation_angle`

Αρχικά, υλοποιούμε την συνάρτηση $angle = find_rotation_angle(x)$, η οποία εντοπίζει την γωνία με την οποία ενδεχομένως έχει περιστραφεί η εικόνα και περιλαμβάνει τα εξής ορίσματα κι εξόδους:

- x : η εικόνα της οποίας θέλουμε να εντοπίσουμε την γωνία περιστροφής
- $angle$: η γωνία που εντοπίσαμε

Για την αναίρεση της περιστροφής που έχει υποστεί μια εικόνα κειμένου χρησιμοποιούμε πρώτα τον διδιάστατο διακριτό μετασχηματισμό Fourier (DFT), αφού καταρχάς μετατρέψουμε την εικόνα σε grayscale και την θολώσουμε κατάλληλα, χρησιμοποιώντας Gaussing Blur, ώστε να συνενωθούν τα γράμματα κάθε γραμμής. Το μέτρο, λοιπόν, του μετασχηματισμού Fourier (DFT) για την εικόνα προς ανάγνωση έχει την παρακάτω μορφή:



Σχήμα 2.1: Λογάριθμος του μέτρου του *DFT*

Έχουμε αποκόψει, μάλιστα, το κατάλληλο τμήμα του λογαρίθμου του μέτρου του DFT, αφαιρώντας την DC συνιστώσα και έναν δίσκο μικρής ακτίνας γύρω από αυτήν. Αφαιρούμε,

ακόμη, τις υψηλές συχνότητες, καθώς εισάγουν θόρυβο και μας εμποδίζουν από τον σωστό εντοπισμό της γωνίας. Έτσι, εντοπίζουμε την μέγιστη συχνότητα μεταβολής που αντιστοιχεί στην μεταβολή της φωτεινότητας από γραμμή (κειμένου) σε γραμμή και για εκείνο το πίξελ σχεδιάζουμε την ευθεία που διέρχεται από αυτό και από το κέντρο της εικόνας. Η κλίση αυτής της ευθείας στο κατάλληλο τεταρτημόριο, μας δίνει μια πρώτη εκτίμηση της γωνίας περιστροφής. Ουστόσο, η μέθοδος αυτή δεν είναι απόλυτα ακριβής, κυρίως λόγω της περιορισμένης ανάλυσης του DFT.

Για να εντοπίσουμε, λοιπόν, ακριβώς την γωνία, κάνουμε μία σειριακή αναζήτηση γύρω από την πρώτη εκτίμηση της γωνίας. Για κάθε υποψήφια γωνία, αναφούμε την περιστροφή, βρίσκουμε την προβολή της φωτεινότητας στον κατακόρυφο άξονα και υπολογίζουμε το άθροιστικό *gradient*. Κριτήριο επιλογής, έτσι, της βέλτιστης προσέγγισης της γωνίας, είναι αυτή με το μεγαλύτερο άθροισμα από μεταβολές της φωτεινότητας στην προβολή.

2.2 **rotate_image**

Τηλοποιούμε, έπειτα, την συνάρτηση $y = \text{rotate_image}(x, angle)$, η οποία περιστρέφει μία εικόνα κατά μια οποιαδήποτε γωνία κι έχει τα παρακάτω ορίσματα κι εξόδους:

- x : η εικόνα που θέλουμε να περιστρέψουμε
- $angle$: η γωνία κατά την οποία θέλουμε να περιστρέψουμε την εικόνα (θετική ή αρνητική)
- y : η επιστρεφόμενη περιεστραμμένη εικόνα κατάλληλα γεμισμένη

Η συνάρτηση αυτή περιστρέφει την εικόνα εφαρμόζοντας σε αυτήν έναν κατάλληλο πίνακα περιστροφής και γεμίζοντας τα κενά που προκύπτουν περιφερειακά από την περιστροφή με κατάλληλο *padding* με την μέθοδο της γραμμικής παρεμβολής.

3 Περιγραφή Περιγράμματος

Έχουμε πλέον την εικόνα μας κατάλληλα προεπεξεργασμένη για την διαδικασία αναγνώρισης των χαρακτήρων της. Κανέως, όμως, η αναγνώριση θα γίνει για καθέναν από τους χαρακτήρες χωριστά, θα προχωρήσουμε πρώτα στην μελέτη μιας κατάτυησης της εικόνας. Συγκεκριμένα, θα επιχειρήσουμε να περιγράψουμε το περίγραμμα του κάθε γράμματος της παρακάτω δοκιμαστικής εικόνας:

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

Σχήμα 3.1: Δοκιμαστική εικόνα για εύρεση περιγραμμάτων

Η υψηλή ανάλυση και ευχρίνεια των γραμμάτων σε αυτήν την εικόνα θα μας διευκολύνει τόσο στον σχεδιασμό ενός ικανοποιητικού αλγορίθμου εύρεσης των περιγραμμάτων, όσο και στην εύρεση του κατάλληλου μετρικού για την αναπαράστασή τους.

3.1 `get_contour`

Τηλοποιούμε την συνάρτηση $c = get_contour(x)$, η οποία εφαρμόζει τον αλγόριθμο εύρεσης περιγραμμάτων μετά από κατάλληλη επεξεργασία της εικόνας και περιλαμβάνει τα εξής ορίσματα κι εξόδους:

- x : εικόνα που περιέχει έναν μόνο χαρακτήρα του οποίου το περίγραμμα επιθυμούμε να περιγράψουμε
- c : μια λίστα με τα περιγράμματα του χαρακτήρα ως σύνολο από συντεταγμένες του περιγράμματος πάνω στο πλέγμα της εικόνας

Αρχικά, θα απομονώσουμε το περίγραμμα του γράμματος χρησιμοποιώντας τους κατάλληλους μορφολογικούς μετασχηματισμούς από την ψηφιακή επεξεργασία εικόνας. Συγκεκριμένα, για να υπάρχει ομοιομορφία μεταξύ των εικόνων που περιέχουν τους δεδομένους χαρακτήρες, πραγματοποιούμε *resize* και *normalize* των πίζελς της εικόνας, ενώ η εικόνα μετατρέπεται σε

3.1. get_contour

grayscale. Προς διευκόλυνση, ακόμη, της εύρεσης του περιγράμματος θολώνουμε ελαφρώς την εικόνα με την χρήση Gaussian Blur.



Σχήμα 3.2: *Original character*

Σχήμα 3.3: *Resized and blurred*



Σχήμα 3.4: *Subtracted the dilated*

Σχήμα 3.5: *Thinned*

Στη συνέχεια, υπολογίζουμε την *dilated* εκδοχή της εικόνας και αφαιρούμε από αυτή την αρχική, με αποτέλεσμα την εμφάνιση ενός, ενδεχομένως άτσαλου, περιγράμματος. Για την εξομάλυνσή του, μετατρέπουμε την εικόνα σε *binary* βάσει ενός κατοφλίου, ενώ έπειτα εφαρμόζουμε και *thinning* ώστε να μειώσουμε το πάχος των ακμών του περιγράμματος στο ένα πιξελ. Η διαδικασία που περιγράφηκε φαίνεται εποπτικά στα παραπάνω σχήματα.

Έπειτα σχεδιάζουμε έναν αλγόριθμο εύρεσης των συντεταγμένων ενός περιγράμματος κατά την ωρολογιακή φορά. Ο αλγόριθμός μας απαιτείται να αναγνωρίζει όλα τα περιγράμματα ενός χαρακτήρα (ένα, δύο ή τρια στην περίπτωσή μας) και να τα ιεραρχεί, ώστε να μπορεί να αποφασιστεί ποιό είναι το πιο εξωτερικό/εσωτερικό περίγραμμα. Παρακάτω παραθέτουμε τον ψευδοκώδικα του αλγορίθμου εύρεσης περιγραμμάτων που σχεδιάσαμε:

Find the coordinates of all black pixels

Start at top left corner

while True:

 Define search area around current pixel

 Find all neighboring black pixels in search area

 if not neighbors:

 Append current contour to the "c" list

```

for pixel in black pixels:

    if pixel not in "c" list:

        Begin new contour search starting from this
        black pixel

        break

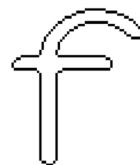
    if there is new contour:
        continue
    else:
        break
else:
    Define the next pixel as the one located to the nearest
    clockwise position and append it to the current contour

```

Ο παραπάνω αλγόριθμος πετυχαίνει να εντοπίσει όσα περιγράμματα περιέχει ένας χαρακτήρας και τα τοποθετεί στην λίστα *c* από τα πιο εξωτερικά στα πιο εσωτερικά. Ανιχνεύει τις συντεταγμένες του περιγράμματος πάνω στο πλέγμα κατά την ωρολογιακή φορά με μια επαναληπτική διαδικασία, η οποία τερματίζεται μόλις φτάσουμε στο σημείου εκκίνησης του περιγράμματος.



Σχήμα 3.6: Περιγράμματα του *a*



Σχήμα 3.7: Περιγράμματα του *f*



Σχήμα 3.8: Περιγράμματα του *l*



Σχήμα 3.9: Περιγράμματα του *e*

Παραπάνω βλέπουμε ενδεικτικά τα περιγράμματα τεσσάρων χαρακτήρων της εικόνας 3.1. Σε περίπτωση που ένας χαρακτήρας έχει παραπάνω από ένα περιγράμματα, αυτά τονίζονται με διαφορετική απόχρωση για λόγους διακριτήτας.

3.2 get_descriptor

Προκειμένου να γίνει η αναπαράσταση και η σύγκριση των περιγραμμάτων που θα οδηγήσει στην αναγνώριση των χαρακτήρων του κειμένου είναι απαραίτητο να κατασκευάσουμε έναν περιγραφέα περιγράμματος.

Ορίζουμε, έτσι, την συνάρτηση $descriptor = get_descriptor(contour)$, η οποία υπολογίζει τον περιγραφέα ενός περιγράμματος κι έχει τα εξής ορίσματα και εξόδους:

- *contour*: λίστα με τις συντεταγμένες των σημείων που αποτελούν το περίγραμμα
- *descriptor*: ο περιγραφέας του περιγράμματος

Έστω, λοιπόν, ένα περίγραμμα (από όσα μπορεί να έχει ένας χαρακτήρας) του οποίου τα σημεία είναι τα $x[i]$, $y[i]$, $i = 0, 1, \dots, n - 1$, όπου n είναι ο αριθμός των σημείων τα οποία περιγράφουν το περίγραμμα. Ορίζουμε, αρχικά, την μιγαδική ακολουθία:

$$r[i] = x[i] + jy[i] \quad (3.1)$$

όπου $j = \sqrt{-1}$. Στη συνέχεια, υπολογίζουμε τον DFT της ακολουθίας, έστω $R[i]$. Ο περιγραφέας του περιγράμματος είναι το μέτρο της $R[i]$, δηλαδή $|R[i]|$, εξαιρώντας από αυτή τον πρώτο όρο, δηλαδή για $i = 1, 2, \dots, n - 1$. Ο λόγος που αφαιρούμε την DC αυτή συνιστώσα είναι ώστε να μην επηρεάζει τον περιγραφέα η θέση του περιγράμματος στο πλέγμα της εικόνας, παρά μόνο η σχετική μορφή του περιγράμματος.

Έχουμε δημιουργήσει, λοιπόν, ένα κατάλληλο μετρικό για την σύγκριση δύο περιγραμμάτων. Αυτή μπορεί να γίνει με χριτήριο απόστασης για δύο τροχιές r_1 και r_2 το τετραγωνικό σφάλμα των περιγραφέων τους $|R_1|$ και $|R_2|$, δηλαδή:

$$d(|R_1|, |R_2|) = \sum_{i=1}^{n-1} (|R_1[i]| - |R_2[i]|)^2 \quad (3.2)$$

Όσο μικρότερο είναι αυτό, τόσο περισσότερο μοιάζουν δύο περιγραφέις, κι επομένως δύο περιγράμματα. Πρέπει να παρατηρήσουμε εδώ ότι οι περιγραφείς $|R_1|$ και $|R_2|$ πρέπει να έχουν το ίδιο μήκος. Το χριτήριο αυτό θα το χρησιμοποιήσουμε στην συνέχεια μέσω του αλγορίθμου *KNN* (*K – Nearest – Neighbors*) για την αναγνώριση των χαρακτήρων του κειμένου μέσω των περιγραφέων τους.

4 Ανάγνωση Χαρακτήρων - Σύστημα Ταξινόμησης

Έχουμε σχεδιάσει πλέον μια μέθοδο εύρεσης και περιγραφής ενός περιγράμματος. Αυτή θα αξιοποιηθεί σαν εργαλείο για το επόμενο βήμα, το οποίο είναι να εκπαιδεύσουμε ένα σύστημα ταξινόμησης χαρακτήρων με βάση το κείμενο αναφοράς της εικόνας 1.1.

Προκειμένου να εκπαιδευτεί το σύστημα ταξινόμησης απαιτείται να διαμορφώσουμε πρώτα ένα *dataset*, το οποίο θα αποτελείται από τους περιγραφείς των περιγραμμάτων των χαρακτήρων της προς εκπαίδευση εικόνας, καθώς κι από ένα *label* για τον καθένα που θα αντιστοιχίζει τους περιγραφείς με τους χαρακτήρες *ascii*. Παρακάτω βλέπουμε πιο αναλυτικά την υλοποίηση του *dataset*.

4.1 get_dataset

Τηλοποιούμε, αρχικά, την συνάρτηση *img_dataset, ascii_dataset = get_dataset(img, text)*, η οποία κατασκευάζει μια πρωταρχική μορφή του *dataset* τόσο για τα περιγράμματα, όσο και για τους χαρακτήρες *ascii*. Το *dataset* στην συνέχεια θα υποστεί επεξεργασία ώστε να έρθει σε κατάλληλη μορφή για την χρήση του για την εκπαίδευση του συστήματος ταξινόμησης. Η συνάρτηση *get_dataset()* έχει τα εξής ορίσματα κι εξόδους:

- *img*: η εικόνα κειμένου από την οποία επιθυμούμε να εξάγουμε το *dataset* των χαρακτήρων
- *text*: το *path* του εγγράφου που περιέχει την αντιστοιχία των χαρακτήρων σε *ascii*
- *img_dataset*: μια λίστα που περιέχει τα περιγράμματα των χαρακτήρων της εικόνας κειμένου ταξινομημένα κατά γραμμή και κατά λέξη
- *ascii_dataset*: μια λίστα που περιέχει τους χαρακτήρες *ascii* του ίδιου κειμένου, επίσης ταξινομημένους κατά γραμμή και κατά λέξη

Πρώτα αναφρούμε την οποιαδήποτε περιστροφή της εικόνας χρησιμοποιώντας τις συναρτήσεις που κατασκευάσαμε στην προηγούμενη ενότητα της προεπεξεργασίας. Ακόμη, διώχνουμε από την εικόνα τυχόν έγχρωμα 'σκουπίδια', την μετατρέπουμε σε *grayscale* και την κάνουμε *resize* στις διαστάσεις για τις οποίες είναι πιο αποδοτικός ο αλγόριθμος αναζήτησης περιγραμμάτων. Αυτές επιλέγονται βάσει της λογικής *trial and error*.

Για την δημιουργία του *dataset*, κατάλληλα ταξινομημένου κατά γραμμή και κατά λέξη, πρώτη μας μέριμνα είναι ο διαχωρισμός των γραμμών του κειμένου. Χρησιμοποιούμε την καταχώρυφη προβολή της φωτεινότητας για να εντοπίσουμε τα σημεία στα οποία το κείμενο μεταβαίνει από κείμενο σε κενό *background* και ξεχωρίζουμε, έτσι, τις γραμμές κειμένου μεταξύ τους. Παρακάτω βλέπουμε ένα παράδειγμα αυτού του διαχωρισμού στην εικόνα προς εκπαίδευση:

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

Σχήμα 4.1: Διαχωρισμός γραμμής του κειμένου

Στη συνέχεια, για κάθε γραμμή κειμένου, χρησιμοποιούμε με παρόμοιο τρόπο την προβολή της φωτεινότητας στον οριζόντιο άξονα ώστε να ξεχωρίσουμε τις λέξεις. Πριν εντοπίσουμε τα

σημεία στα οποία η γραμμή του κειμένου μεταβαίνει από λέξη σε ενδιάμεσο κενό μεταξύ λέξεων, θα πρέπει να θολώσουμε κατάλληλα την γραμμή (Gaussing Blur), τόσο ώστε να ενωθούν τα γράμματα της κάθε λέξης μεταξύ τους, αλλά να μην ενωθούν οι λέξεις μεταξύ τους:

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

Σχήμα 4.2: Θολωμένη γραμμή του κειμένου

Μια εναλλακτική λύση που θα μπορούσαμε να ακολουθήσουμε για να αποφύγουμε την διαδικασία του trial and error στον υπολογισμό της κατάλληλης παραμέτρου για το θόλωμα, θα ήταν να ξεχωρίσουμε τις λέξεις μεταξύ τους στα σημεία όπου το κενό είναι μεγαλύτερο από το μέσο κενό μεταξύ των χαρακτήρων. Και με τις δύο μεθόδους, πάντως, καταφέρνουμε σε αυτό το στάδιο να αποκόψουμε τις λέξεις κι όχι τον κάθε έναν χαρακτήρα χωριστά, όπως φαίνεται στο παρακάτω ενδεικτικό παράδειγμα:

Lorem

Σχήμα 4.3: Διαχωρισμός λέξης του κειμένου

Τέλος, ξεχωρίζουμε τα γράμματα της κάθε λέξης με μία απλή επαναληπτική διαδικασία, η οποία ακολουθεί ακριβώς την ίδια λογική με τον διαχωρισμό των λέξεων μεταξύ τους. Υπολογίζουμε, δηλαδή, την προβολή της φωτεινότητας στον οριζόντιο άξονα και χωρίζουμε την λέξη στα σημεία όπου μεταβαίνει από χαρακτήρα σε κενό. Έχει ολοκληρωθεί, με αυτόν τον τρόπο, η κατασκευή του *img_dataset*.

Προκειμένου, σε επόμενο βήμα, να αντιστοιχίσουμε το περιγραμμα (ή τα περιγράμματα) του κάθε χαρακτήρα με ένα *string* που θα υποδηλώνει τον χαρακτήρα αυτόν σε κώδικα *ascii* (*label*), μας διευκολύνει να χωρίσουμε με παρόμοιο τρόπο και τους χαρακτήρες *ascii* του κειμένου *text*. Κατασκευάζουμε, λοιπόν, μια λίστα *ascii_dataset* που περιέχει τους χαρακτήρες χωρισμένους ανά γραμμή και ανά λέξη.

4.2 divide_into_classes

Ένα ενδιάμεσο βήμα πριν την τελική διαμόρφωση του *dataset* που θα περάσει στον αλγόριθμο *KNN* είναι ο χωρισμός του σε τρεις τάξης ανάλογα με τον αριθμό των περιγραμμάτων που έχει κάθε χαρακτήρας. Υλοποιούμε, έτσι, την συνάστηση *class1, class2, class3 = divide_into_classes(img_dataset, ascii_dataset)*, η οποία έχει τα παρακάτω οφίσματα κι εξόδους:

- *img_dataset*: η λίστα που περιέχει τα περιγράμματα των χαρακτήρων της εικόνας κειμένου ταξινομημένα κατά γραμμή και κατά λέξη
- *ascii_dataset*: η λίστα που περιέχει τους χαρακτήρες *ascii* του ίδιου κειμένου, επίσης ταξινομημένους κατά γραμμή και κατά λέξη
- *class1, class2, class3*: λίστες με δύο στοιχεία, από τα οποία το πρώτο είναι μια λίστα με όλα τα περιγράμματα χαρακτήρων που ανήκουν στην εκάστοτε κλάση, και το δεύτερο είναι μια λίστα με όλους τους χαρακτήρες *ascii* που ανήκουν στην κλάση που βρίσκεται σε αντιστοιχία 1 - 1 με τα περιγράμματα

Οι χαρακτήρες με τους οποίους ασχολούμαστε στα πλαίσια αυτής της εργασίας μπορεί να περιέχουν μόνο ένα, δύο ή τρία περιγράμματα, όπως φαίνεται κι από την εικόνα με το κείμενο προς εκπαίδευση του σχήματος 1.1.

Εφόσον, όπως είπαμε, ο εντοπισμός των χαρακτήρων θα γίνει με βάση τους περιγραφείς των περιγραμμάτων τους είναι δυνατόν να συγχρίνουμε δίκαια μόνο χαρακτήρες με τον ίδιο αριθμό περιγραμμάτων μεταξύ τους. Για τον λόγο αυτόν μοιράζουμε τους χαρακτήρες μας σε τρεις τάξεις (κλάσεις) ανάλογα με τον αριθμό των περιγραμμάτων τους, κι ακολούθως θα διαμορφώσουμε τρία τελικά *datasets*, τα οποία θα χρησιμοποιηθούν για την εκπαίδευση τριών ανεξάρτητων μοντέλων από τον αλγόριθμο *KNN*.

Για τον επιτυχή ορισμό των αντιστοίχων *labels* των περιγραμμάτων χωρίζουμε με παρόμοιο τρόπο και τους χαρακτήρες *ascii* στις τρεις κλάσεις, έτσι ώστε κάθε *label* να βρίσκεται μέσα στην λίστα σε αντίστοιχη θέση με το περίγραμμα που περιγράφει.

4.3 form_dataset

Προκειμένου να φέρουμε τις τρεις κλάσεις στην τελική μορφή που πρέπει να έχει ένα *dataset* για να χρησιμοποιηθεί από τον αλγόριθμο *KNN*, υλοποιούμε την συνάρτηση *dataset1*, *dataset2*, *dataset3 = form_dataset(class1, class2, class3, N)* με τα εξής ορίσματα κι εξόδους:

- *class1, class2, class3*: λίστες με δύο στοιχεία, από τα οποία το πρώτο είναι μια λίστα με όλα τα περιγράμματα χαρακτήρων που ανήκουν στην εκάστοτε κλάση, και το δεύτερο είναι μια λίστα με όλους τους χαρακτήρες *ascii* που ανήκουν στην κλάση που βρίσκεται σε αντιστοιχία 1 - 1 με τα περιγράμματα
- *N*: το μέγεθος (της επιλογής μας) του διανύσματος του περιγραφέα κάθε περιγράμματος που προκύπτει μετά από γραμμική παρεμβολή
- *dataset1, dataset2, dataset3*: τα τελικά διαμορφωμένα *datasets* για κάθε μία από τις τρεις τάξεις

Σκοπός μας είναι να διαμορφώσουμε για κάθε τάξη έναν πίνακα (*pandas.DataFrame*) στον οποίο κάθε γραμμή περιλαμβάνει τις *N* συχνότητες του περιγραφέα ενός περιγράμματος της τάξης και ως τελευταίο στοιχείο το *label* του περιγράμματος σαν *string* του χαρακτήρα σε κώδικα *ascii*.

Για να το πετύχουμε αυτό, για κάθε ένα περίγραμμα της κάθε κλάσης καλούμε την συνάρτηση *get_descriptor()* που περιγράψαμε παραπάνω. Σύμφωνα, όμως, με τον τύπο (3.2) για να μπορέσουμε να υπολογίσουμε την απόσταση μεταξύ δύο περιγραμμάτων πρέπει αυτά να έχουν ίδιο μήκος, δηλαδή ίδιο αριθμό σημείων ή με άλλα λόγια ίδιο εύρος ζώνης στην συχνότητα. Αυτό το πετυχαίνουμε με το να κάνουμε γραμμική παρεμβολή στις συχνότητες των περιγραφέων, ώστε να έχουν όλοι οι περιγραφές πλήθος συχνοτήτων *N*.

4.4 train_test

Με τα *datasets* έτοιμα στην μορφή που θέλουμε, μπορούμε πλέον να υλοποιήσουμε την συνάρτηση *conf_matrix, best_accuracy, knn.best = train_test(dataset, n_neighbors)*, η οποία εκπαιδεύει τα συστήματα ανάγνωσης χαρακτήρων για κάθη τάξη χωριστά κι έχει τα παρακάτω ορίσματα κι εξόδους:

- *dataset*: ένα οποιοδήποτε *dataset* στην μορφή που ορίσαμε στην προηγούμενη συνάρτηση
- *n_neighbors*: παράμετρος του αλγορίθμου *KNN*: ο αριθμός των πλησιέστερων γειτόνων που θα χρησιμοποιηθούν για να κατηγοριοποιηθεί ή προβλεψεί μια νέα παρατήρηση
- *conf_matrix*: πίνακας σύγχυσης που αξιολογεί την ταξινομητική απόδοση του αλγορίθμου, παρουσιάζοντας τις σωστές και εσφαλμένες προβλέψεις ανά κατηγορία
- *best_accuracy*: η καλύτερη εκ των σταθμευμένων ακριβειών της εκπαίδευσης του συστήματος
- *knn_best*: το μοντέλο εκπαίδευσης με την υψηλότερη σταθμευμένη ακρίβεια

Χωρίζουμε, αρχικά, το *dataset* σε δύο πίνακες, έναν πίνακα *x* με τις συχνότητες των περιγραφέων κι έναν πίνακα *y* (διάνυσμα) με τις ετικέτες. Για την εκπαίδευση του συστήματος ταξινόμησης χωρίζουμε τους περιγραφείς και τις ετικέτες σε σύνολο εκπαίδευσης (train set) και σύνολο δοκιμής (test set) χρησιμοποιώντας διαχωρισμό 70:30.

Θέτουμε ως παράμετρο *stratify=y*, έτσι ώστε να εξασφαλίσουμε ότι ο κάθε χαρακτήρας θα εκπροσωπείται στα δύο σύνολα σε αντιστοιχία με τον διαχωρισμό αυτό. Η παράμετρος *stratify* χρησιμοποιείται, δηλαδή, για να διατηρηθεί η ισορροπία των κλάσεων στο σύνολο εκπαίδευσης και ελέγχου, εξασφαλίζοντας ένα ισορροπημένο δείγμα ανάμεσα στις διάφορες κλάσεις των δεδομένων.

Επόμενο βήμα είναι η εκπαίδευση του ταξινομητή *KNN* (K-Nearest-Neighbors), ο οποίος είναι ένας αλγόριθμος που βασίζεται στην απόσταση ανάμεσα στα δείγματα και την κλάση των κοντινότερων γειτόνων για την κατηγοριοποίηση νέων δειγμάτων. Βασίζεται επί της ουσίας στο μετρικό που ορίσαμε από τον τύπο (3.2).

Έχοντας εκπαίδευτεί, το σύστημα μπορεί πλέον να κάνει πρόβλεψη των ετικετών του συνόλου δοκιμής και να υπολογίσει βάσει αυτού, τον πίνακα σύγχυσης και την σταθμευμένη ακρίβεια που ορίστηκαν παραπάνω. Για μια πιο αντιπροσωπευτική τιμή της σταθμευμένης ακρίβειας προπονούμε το σύστημα ταξινόμησης και προβλέπουμε τις ετικέτες του συνόλου δοκιμής για πολλές διαφορετικές κατατμήσεις του *dataset* κι έπειτα υπολογίζουμε την μέση τιμή. Ωστόσο, για την χρήση, στη συνέχεια, του εκπαιδευμένου συστήματος για την ανάγνωση του κειμένου του σχήματος 1.2 θα χρησιμοποιήσουμε το εκπαίδευμένο σύστημα ταξινόμησης με την μεγαλύτερη σταυθμευμένη ακρίβεια.

Σαν σημείωση αναφέρουμε, επίσης, ότι σε περίπτωση που κάποιο δείγμα εμφανίζεται μόνο μία φορά, κι επειδή αυτό δεν αρκεί για τον χωρισμό του σε σύνολο εκπαίδευσης και δοκιμής, η λύση που προτείνουμε είναι ένας απλός διπλασιασμός του δείγματος.

4.5 **read_text**

Τελευταίο βήμα στο σύστημα ανάγνωσης κειμένου από φωτογραφία που σχεδιάζουμε είναι η αξιοποίηση των παραπάνω για την εκπαίδευση του συστήματος ταξινόμησης σε μία φωτογραφία κειμένου κι ο έλεγχός του σε μία άλλη φωτογραφία κειμένου. Για τους σκοπούς της παρουσίασης θα χρησιμοποιήσουμε, όπως είπαμε, τις δύο φωτογραφίες των σχημάτων 1.1 και 1.2.

Τλοποιούμε, έτσι, την συνάρτηση *lines = read_text(train_img, train_text, test_img, test_text)*, η οποία έχει τα εξής ορίσματα κι εξόδους:

- *train_img*: η εικόνα κειμένου που θα χρησιμοποιηθεί για την εκπαίδευση του συστήματος ταξινόμησης
- *train_text*: το κείμενο σε μορφή *ascii* της εικόνας εκπαίδευσης προκειμένου να γίνει η αντιστοιχία των ετικετών
- *test_img*: η εικόνα κειμένου που θα χρησιμοποιηθεί για την αξιολόγηση του συστήματος ταξινόμησης
- *test_text*: το κείμενο σε μορφή *ascii* της εικόνας δοκιμής προκειμένου να υπολογιστεί η ακρίβεια της ανάγνωσης
- *lines*: λίστα με τους χαρακτήρες σε *ascii* του κειμένου δοκιμής

Η συνάρτηση αυτή, αφού κάνει την προεπεξεργασία των εικόνων με τις μεθόδους που παρουσιάστηκαν σε προηγούμενες ενότητες, διαμορφώνει τα *datasets* της εικόνας *train_img*, ένα για κάθε μία τάξη χαρακτήρων. Στη συνέχεια, βρίσκει τα τρία βέλτιστα εκπαιδευμένα συστήματα ταξινόμησης και τα αξιοποιεί για την πρόβλεψη των χαρακτήρων του κειμένου της εικόνας *test_img*.

Κατασκευάζεται με αυτόν τον τρόπο η λίστα *lines* που περιέχει το κείμενο της εικόνας προς ανάγνωση σε μορφή *ascii*. Το κείμενο αυτό γράφεται και αποθηκεύεται σε αρχείο *output.txt*, ώστε να φανεί παραστατικά η επίδοση του συστήματος ανάγνωσης. Για να υπολογιστεί, τέλος, και το ποσοστό ακρίβειας (*accuracy*) της ανάγνωσης συγχρίνουμε το κείμενο που παράγεται στο αρχείο *output.txt* με το κείμενο του αρχείου *test_text*.

5 Αποτελέσματα - Σχολιασμοί

Κατά τον σχεδιασμό του συστήματος ανάγνωσης κειμένου από φωτογραφία έγινε έλεγχος για την λειτουργικότητα της κάθισ μιας συνάρτησης χωριστά. Παρακάτω θα παρουσιάσουμε το τελικό αποτέλεσμα της ανάγνωσης στο παράδειγμα των δύο εικόνων που μελετήσαμε και θα σχολιάσουμε τις επιδόσεις και τις αδυναμίες του συστήματος, καθώς και τρόπους με τους οποίους αυτό θα μπορούσε να βελτιωθεί.

Για την εκτίμηση της αποδοτικότητας του αλγορίθμου παίρνουμε ένα δείγμα 10 δοκιμών και κατασκευάζουμε τον παρακάτω στατιστικό πίνακα:

Εκτέλεση	Ακρίβεια
1 ^η	84.56 %
2 ^η	85.78 %
3 ^η	82.23 %
4 ^η	87.77 %
5 ^η	86.17 %
6 ^η	81.38 %
7 ^η	83.59 %
8 ^η	88.19 %
9 ^η	84.26 %
10 ^η	83.98 %
Μέση Τιμή	84.791 %
Τυπική Απόκλιση	2.096 %

Προς οπτικοποίηση των παραπάνω ενδείξεων ακριβείας παραθέτουμε παρακάτω την σύγκριση μεταξύ του κειμένου της φωτογραφίας δοκιμής με αυτό το οποίο παρήγαγε το σύστημα ανάγνωσής μας:

the battle of the coral sea, fought during 4-8 may 1942, was a major naval battle in the pacific theater of world war ii between the imperial japanese (ijs) and naval and air forces from the united states and australia. the battle was the first action in which aircraft carriers engaged each other, as well as the first in which neither side's ships sighted or fired directly upon the other. in an attempt to strengthen their defensive positioning for their empire in the south pacific, japanes forces decided to invade and occupy port moresby in new guinea and tulagi in the southeastern solomon islands. the plan to accomplish this, called operation mo, involved several major units of japan's combined fleet, including two fleet carriers and a light carrier to provide air cover for the invasion fleets, under the overall command of japanes admiral shigeyoshi inoue. the us learned of the japanes plan through signals intelligence and sent two united states navy carrier task forces and a joint australian-american cruiser force, under the overall command of admiral frank j. fletcher, to oppose the japanes offensive. on 3-4 may, japanes forces successfully invaded and occupied tulagi, although several of their supporting warships were surprised and sunk or damaged by aircraft from the us fleet carrier yorktown, now aware of the presence of us carriers in the area, the japanes fleet carriers advanced towards the coral sea with the intention of finding and destroying the allied naval forces. beginning on 7 may, the carrier forces from the two sides exchanged in airstrikes over two consecutive days. the first day, the us sank the japanes light carrier shoho, while the japanes sank a us destroyer and heavily damaged a fleet oiler (which was later scuttled). the next day, the japanes fleet carrier shokaku was heavily damaged, the us fleet carrier lexington was critically damaged (and was scuttled as a result), and the yorktown was damaged. with both sides having suffered heavy losses in aircraft and carriers damaged or sunk, the two fleets disengaged and retired from the battle area. because of the loss of carrier air cover, inoue recalled the port moresby invasion fleet, intending to try again later.

the battle of the coral sea, fought during 4-8 may 1942, was a major naval battle in the pacific theater of world war ii between the imperial japanese (ijs) and naval and air forces from the united states and australia. the battle was the first action in which aircraft carriers engaged each other, as well as the first in which neither side's ships sighted or fired directly upon the other. in an attempt to strengthen their defensive positioning for their empire in the south pacific, japanes forces decided to invade and occupy port moresby in cef auine and tulagi in the southeastern solomon islands. the plan to accomplish this, called operation mo, involved several major units of japan's combined fleet, including two fleet carriers and a right carrier to provide air cover for the invasion fleets, under the overall command of japanes admiral shigeyoshi inoue. the cs reenred on the japanes dlon through signals intelligence and sent two united states navy carrier task forces and a joint australian-american cruiser force, under the overall command of admiral frank j. fletcher, to oppose the japanes offensive. on 3-4 may, japanes forces successfully invaded and occupied tulagi, although several of their supporting warships were surprised and sunk or pamaged by aircraft from the cs fleet carrier yorktofn, not afere on the dresence on cs carriers in the area, the japanes fleet carriers advanced tofards the coral sea with the intention of finding and pestroying the allied celal forces. beginning on 7 mea, the carrier forces from the tfo sides exchanged in airstrikes over tfo nousecutive paf. the first paf, the cs sank the japanes right nrier shoho, while the japanes sanh a cs pestroyer and weevil pamaged a fleet oiler lshich wes roter scuttled'. the ext paf, the japanes fleet carrier shohuh wes weevil pamaged, the cs fleet carrier rexintzon wes criticaflly pamaged land wes scuttled as a result, and the yorktofn wes pamaged. with both sides wavng suffered weevf rosse in aircraft and carriers pamaged or sink, the tfo fleets disengaged and retired from the detle area. becuse of the ross on carrier air cover, inoue recalled the dort moresdf invasion fleet, intendng to trf again rater.

Σχήμα 5.1: Πρωτότυπο κείμενο

Σχήμα 5.2: Αναγνωσμένο

Σχόλια - Παρατηρήσεις:

- Βλέπουμε ότι υπάρχουν λάθη στην ανάγνωση του κειμένου, ωστόσο το αποτέλεσμα είναι αρκετά ικανοποιητικό. Βάσει του επιμέρους ελέγχου των συναρτήσεων, η αυτία των λαθών εντοπίζεται στην συνάρτηση `get_contour()`. Αυτή είναι εξαιρετικά αποτελεσματική στην εύρεση των περιγραμμάτων για δεδομένες αναλογίες και διαστάσεις του γράμματος μέσα στο πλαίσιο της εικόνας, ωστόσο όταν η εικόνα αυτή είναι ελαφρά παραμορφωμένη ή το γράμμα δεν εμφανίζεται σε ικανοποιητική ανάλυση, υπάρχει περίπτωση κάποια περιγράμματα να εμφανίζονται συνενωμένα σε σημεία που δεν θα έπρεπε και γι' αυτό τον λόγο δεν εντοπίζονται σωστά. Το πρόβλημα αυτό αντιμετωπίζεται εν μέρη με το `resize` της εικόνας και την ρύθμιση των παραμέτρων των συναρτήσεων `blur`, `normalize` ή `dilate`, ωστόσο η εύρεση των βέλτιστων παραμέτρων είναι μια περίπλοκη και χρονοβόρα διαδικασία. Θα μπορούσαν να χρησιμοποιηθούν αλγόριθμοι (π.χ. γενετικοί αλγόριθμοι) εκτίμησης παραμέτρων για την εύρεση καταλληλότερων τιμών βάσει του δείχτη ακρίβειας, ωστόσο κάτι τέτοιο ξεφεύγει από τα πλαίσια της εργασίας.
- Η παραπάνω αστοχία στην αξιόπιστη εύρεση των περιγραμμάτων έχει ως αποτέλεσμα ένα (μικρό) ποσοστό των χαρακτήρων να κατατάσσεται σε διαφορετική τάξη από αυτήν που θα έπρεπε, με συνέπεια και το τελικό σφάλμα στην ανάγνωση του κειμένου της φωτογραφίας δοκιμής. Η χρήση, για παράδειγμα, της έτοιμης συνάρτησης `findContours()` της βιβλιοθήκης `OpenCV` βελτιώνει σημαντικά την επίδοση του συστήματος, ωστόσο αποφεύχθηκε.
- Για τις υπόλοιπες συναρτήσεις που υλοποιήσαμε ως μέρος του συστήματος ανάγνωσης μπορούμε να αποφανθούμε πως δουλεύουν με ικανοποιητική ακρίβεια. Τόσο η συνάρτηση `find_rotation_angle()`, όσο και οι συναρτήσεις σχηματισμού των `datasets` και εκπαίδευσης του συστήματος ταξινόμησης λειτουργούν πολύ ικανοποιητικά και δεν έχουν καμία αρνητική επίδραση στην επίδοση του συστήματος αναγνώρισης. Χαρακτηριστικά, αναφέρουμε ότι η σταθμευμένη ακρίβεια στο σύνολο δοκιμής σε κάθε ένα από τα τρία συστήματα ανάγνωσης (ένα για κάθε τάξη) είναι κοντά στο 100%.
- Όσον αφορά την παράμετρο N του αριθμού των συχνοτήτων των περιγραφέων, παρατηρούμε ότι αυξάνοντας την τιμή (με άνω όριο το 1000) βλέπουμε σημαντική βελτίωση στην ακρίβεια της ανάγνωσης, ωστόσο και αρκετά πιο αργή εκτέλεση. Στο `demo` της εργασίας θεωρούμε $N = 1000$ για βέλτιστα αποτελέσματα, ωστόσο, καθώς η εκτέλεση τότε έχει χρόνο της τάξης των μερικών λεπτών, σε περίπτωση που είναι επιθυμητή μια πιο γρήγορη εκτέλεση του κώδικα, συνίσταται να μειωθεί η τιμή του N (ενδεικτικά $N = 100$).
- Όσον αφορά την παράμετρο `n_neighbors` του αλγορίθμου `KNN` παρατηρούμε παρόμοια επίδοση του συστήματος ταξινόμησης για τιμές 1 - 3, ωστόσο για υψηλότερες τιμές παρατηρούμε μείωση της ακρίβειας ανάγνωσης. Ένδεικτικά, η τιμή `n_neighbors = 1` σημαίνει ότι ο ταξινομητής θα βασίζεται μόνο στον κοντινότερο γείτονα για την ανάθεση ετικέτας σε κάθε ένα νέο δείγμα.
- Τέλος, σαν επιπλέον πρόταση βελτίωσης της επίδοσης του συστήματος οπτικής ανάγνωσης χαρακτήρων αναφέρουμε την προσθήκη επιπλέον μετρικού για τον χαρακτηρισμό των περιγραμμάτων. Για παράδειγμα, σε συνδυασμό με τους περιγραφείς περιγραμμάτων που

ορίσαμε παραπάνω, όταν μπορούσαμε να χρησιμοποιήσουμε, ακόμη, το εμβολό της εσωτερικής επιφάνειας των περιγραμμάτων και να το προσθέσουμε σαν στοιχείο του *dataset* με έναν συντελεστή βαρύτητας. Βοηθητική, χωρίς αμφιβολία, όταν ήταν και η ύπαρξη επιπρόσθετων τάξεων χαρακτήρων (π.χ. μέσα/έξω περιγράμματα, έξω/μέσα πάνω/ μέσα κάτω περιγράμματα), ώστε να γίνεται ακόμη πιο δίκαιη σύγχριση των χαρακτήρων.

Βιβλιογραφία

- [1] *Fourier Transform in Image Processing*, <https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm>
- [2] *Suzuki Contour Algorithm*, <https://theailearner.com/tag/suzuki-contour-algorithm-opencv/>