



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

**ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΑΝΑΛΥΣΗ ΕΙΚΟΝΑΣ

ΤΕΚΜΗΡΙΩΣΗ ΕΡΓΑΣΙΑΣ

ΟΜΑΔΑ ΑΝΑΠΤΥΞΗΣ:

- **ΝΙΚΟΛΑΣ ΠΑΤΕΡΑΣ – Π17172**
- **ΑΝΔΡΕΑΣ ΘΕΟΔΩΡΙΔΗΣ – Π17164**
- **ΒΑΣΙΛΕΙΟΣ ΖΑΡΤΗΛΑΣ ΠΑΠΑΧΑΡΑΛΑΜΠΟΥΣ – Π17168**

Πειραιάς, Φεβρουάριος 2021

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ.....	2
ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ	4
ΕΙΣΑΓΩΓΗ	5
1. ΥΠΟΕΡΩΤΗΜΑ ΠΡΩΤΟ	6
1.1. Ζητούμενα.....	6
1.2. Υλοποίηση	6
1.3. Αποτελέσματα.....	9
2. ΥΠΟΕΡΩΤΗΜΑ ΔΕΥΤΕΡΟ	10
2.1. Ζητούμενα.....	10
2.2. Υλοποίηση	10
2.3. Αποτελέσματα.....	14
3. ΥΠΟΕΡΩΤΗΜΑ ΤΡΙΤΟ	14
3.1. Ζητούμενα.....	15
3.2. Υλοποίηση	15
3.3. Αποτελέσματα.....	17
4. ΥΠΟΕΡΩΤΗΜΑ ΤΕΤΑΡΤΟ	18
4.1. Ζητούμενα.....	18
4.2. Υλοποίηση	18
4.3. Αποτελέσματα.....	22

5.	ΥΠΟΕΡΩΤΗΜΑ ΠΕΜΠΤΟ	23
5.1.	Ζητούμενα.....	23
5.2.	Υλοποίηση	23
5.3.	Αποτελέσματα.....	24
6.	ΥΠΟΕΡΩΤΗΜΑ ΕΚΤΟ	25
6.1.	Ζητούμενα.....	25
6.2.	Υλοποίηση	25
7.	ΑΠΟΤΕΛΕΣΜΑ ΚΟΝΣΟΛΑΣ	27
8.	ΒΙΒΛΙΟΓΡΑΦΙΑ.....	29
9.	DEPENDENCIES/ΕΡΓΑΛΕΙΑ.....	30
9.1.	Βιβλιοθήκες.....	30

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

Εικόνα 1.1: Αρχική εικόνα.....	9
Εικόνα 1.2: Εικόνα χρωματικού χώρου Lab	9
Εικόνα 2.1: Εικόνα κονσόλας	14
Εικόνα 3.1: Εικόνα superpixels.....	17
Εικόνα 4.1: Ασπρόμαυρη εικόνα.....	22
Εικόνα 4.2: Εικόνα κονσόλας	22
Εικόνα 5.1: Εμφάνιση αρχείων	24
Εικόνα 5.2: Χρωματισμένη εικόνα RGB.....	24
Εικόνα 6.1: Παράδειγμα αλγορίθμου	26
Εικόνα 7.1: Εικόνα κονσόλας	27

ΕΙΣΑΓΩΓΗ

Αρχικά, χρησιμοποιήσαμε μερικές εικόνες αυτοκινήτων που βρήκαμε στο Διαδίκτυο σαν δείγματα για την εκτέλεση της εργασίας. Οι εικόνες αυτές τροποποιήθηκαν στο Photoshop για να ελαχιστοποιήσουμε το μέγεθος τους και το resolution τους, αυτό έγινε διότι χρησιμοποιήσαμε σχετικά μεγάλες εικόνες με αποτέλεσμα η εκμάθηση να χρειάζεται ώρες και ώρες για να επιτευχθεί. Κατά την υλοποίηση της εργασίας χρησιμοποιήθηκαν αρκετές βιβλιοθήκες που προσφέρει η γλώσσα Python έτσι ώστε να γίνει γρηγορότερα η υλοποίηση της αλλά ταυτόχρονα έγινε προσπάθεια έτσι ώστε να καταλάβουμε πως λειτουργεί κάθε συνάρτηση κλπ.

Η γενική διαδικασία όλου του αλγορίθμου ανάλυσης μιας εικόνας είναι η εξής:

Εικόνα για εκπαίδευση

Εκπαίδευση



Διακριτοποίηση
του Χρωματικού
Χώρου.

Εξαγωγή
χαρακτηριστικών
σε N τυχαία
pixels

Ανάλυση
Κύριων
Χαρακτηρισ-
τικών (PCA)

Εκπαίδε-
υση των
SVMs (1
vs ALL)

1. ΥΠΟΕΡΩΤΗΜΑ ΠΡΩΤΟ

1.1. Ζητούμενα

Να γίνει αναπαράσταση Εικόνας στον Χρωματικό Χώρο Lab.

1.2. Υλοποίηση

Ο χρωματικός χώρος **CIELAB** αναφερόμενος και ως **L*a*b*** είναι το χρωματικό μοντέλο που είναι βασισμένο σε μη γραμμικό μετασχηματισμό του CIE XYZ.

Τα ιδιαίτερα χαρακτηριστικά του χώρου Lab:

- Η συνιστώσα L^* (Luminance) εκφράζει την ομοιόμορφη φωτεινότητα (0 για μαύρο και 100 για άσπρο). Στην ουσία αντιστοιχεί στην ένταση του γκρι χρώματος για μια ασπρόμαυρη εικόνα άρα έχουμε ένα κανάλι που δεν μας δίνει χρωματική πληροφορία αλλά μας δίνει ένταση χρώματος.
- Χρωματικά κανάλια (a^* και b^*) αναφέρονται σε χρωματικές διαφορές δηλαδή τα τέσσερα ιδιαίτερα χρώματα της ανθρώπινης όρασης.
 - $a(x,y)$: Red-Green-> Έχουμε μία ένδειξη του κόκκινου και πράσινου χρώματος.
 - $b(x,y)$: Yellow-Blue-> Έχουμε μία ένδειξη του κίτρινου και μπλε χρώματος.

Βασικό χαρακτηριστικό του χρωματικού χώρου Lab είναι ότι το κανάλι L είναι ανεξάρτητο των δύο χρωματικών καναλιών (a,b), μπορούμε να μεταβάλουμε τον χρωματικό περιεχόμενο ενός εικονοστοιχείου, δηλαδή μεταβολή των χρωματικών συνιστωσών a,b, χωρίς όμως να μεταβάλλεται η ένταση του χρώματος.

Με την μετατροπή λοιπόν από τον χρωματικό χώρο RGB με 3 χρωματικές συνιστώσες (Red, Green, Blue) σε χρωματικό χώρο Lab με 2 χρωματικές συνιστώσες ο αλγόριθμος από την πληροφορία του πίνακα $L(x,y)$ θα εκτιμήσει μόνο δύο χρωματικές συνιστώσες περιορίζοντας το πλήθος των αγνώστων που θα εκτιμούσε. Άρα απλοποιούμε το πρόβλημα μας.

Χρησιμοποιώντας την συνάρτηση `cv2.COLOR_RGB2LAB` εκτελείται ο πιο κάτω αλγόριθμος.

Στην περίπτωση μας χρησιμοποιήσαμε **8-bit** εικόνες οπότε το R, G και B μετατρέπονται στη μορφή floating-point και κλιμακώνονται ώστε να ταιριάζουν στο εύρος από 0 (μηδέν) έως 1 (ένα), δηλαδή:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow X/X_n, \text{ όπου } X_n = 0.950456$$

$$Z \leftarrow Z/Z_n, \text{ όπου } Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 \cdot Y^{(1/3)} - 16, & Y > 0.008856 \\ 903.3 \cdot Y, & Y \leq 0.008856 \end{cases}$$

$$a \leftarrow 500(f(X) - f(Y)) + \text{delta}$$

$$b \leftarrow 200(f(Y) - f(Z)) + \text{delta}$$

όπου,

$$f(t) = \begin{cases} t^{(1/3)}, & t < 0.008856 \\ 7.787t + 16/116, & t \geq 0.008856 \end{cases}$$

και

$$\text{delta} = \begin{cases} 128, & \text{για } 8\text{-bit εικόνες} \\ 0, & \text{για floating-point εικόνες} \end{cases}$$

Αυτό μας δίνει σαν αποτέλεσμα $0 \leq L \leq 100, -127 \leq a \leq 127, -127 \leq b \leq 127$. Οι τιμές στη συνέχεια μετατρέπονται στους αντίστοιχους τύπους δεδομένων:

- 8-bit: $L \leftarrow L \cdot 255/100, a \leftarrow a + 128, b \leftarrow b + 128$
- 16-bit: (Δεν υποστηρίζεται)
- 32-bit: Το L, a και b παραμένουν όπως είναι.

Στο αρχείο `utilities.py` υπάρχουν οι βασικές συναρτήσεις του προγράμματος. Ταυτόχρονα η μέθοδος `display_end_result(αρχική εικόνα, μαυρόασπρη εικόνα, χρώματα)` εμφανίζει στο τέλος τα αποτελέσματα όλου του προγράμματος και για τα άλλα υποερωτήματα μαζί, στο παρόν υποερώτημα είναι αρκετό να εμφανίσουμε την αρχική εικόνα σε LAB.

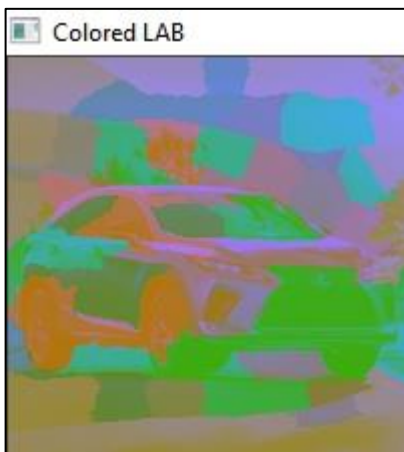
1.3. Αποτελέσματα

Αρχική εικόνα:



Εικόνα 1.1: Αρχική εικόνα

Εικόνα στον χρωματικό χώρο Lab:



Εικόνα 1.2: Εικόνα χρωματικού χώρου Lab

2. ΥΠΟΕΡΩΤΗΜΑ ΔΕΥΤΕΡΟ

2.1. Ζητούμενα

Να γίνει διακριτοποίηση του Χρωματικού Χώρου Lab με βάση ένα σύνολο συναφών εικόνων εκπαίδευσης.

2.2. Υλοποίηση

Στο παρόν ερώτημα έγινε χρήση ενός συνόλου εικόνων που σχετίζονται μεταξύ τους, αυτό εννοεί όρος συναφείς εικόνες. Πρώτα από όλα η διακριτοποίηση γίνεται στο αρχείο `image_segmentation.py` με την βοήθεια της βιβλιοθήκης **SciKit-Learn** με το package **KMeans**. Οι 10 από τις 11 εικόνες θα χρησιμοποιηθούν για να κάνουμε fit το μοντέλο KMeans.

Ο αλγόριθμός KMeans με πολύ απλά λόγια θα υπολογίζει για κάθε cluster το K-mean για να μπορούμε να πάρουμε μια χρωματική απόχρωση που είναι αντιπροσωπευτική του εκάστοτε cluster.

Σημείωση: Όσο μικρότερο είναι το πλήθος K τόσο πιο ακριβής θα είναι η ταξινόμηση αλλά τόσο χειρότερο θα είναι το τελικό αποτέλεσμα του χρωματισμού. Αυτό συμβαίνει γιατί περιορίζεται το φάσμα των χρωμάτων της αρχικής εικόνας σε ένα φάσμα που περιορίζεται σε K συνδυασμούς χρωμάτων. Από την άλλη όσο αυξάνεται το πλήθος K τόσο μειώνεται η ακρίβεια στην διαδικασία της ταξινόμησης, όπου σε αυτήν την περίπτωση θα χρειαζόμασταν και περισσότερα δείγματα.

Ο αλγόριθμος KMeans της βιβλιοθήκης SkLearn λειτουργεί ως εξής:

- 1) Πρώτα επιλέγουμε τον αριθμό των clusters που θα σχηματιστούν και των αριθμό των κεντροειδών που θα παραχθούν. Εμείς επιλέγουμε τον αριθμό 16 για να γίνει clustering σε 16 χρώματα.

- 3) Αφού γίνει αυτό θα γίνει αρχικοποίηση των κεντροειδών ανακατεύοντας πρώτα το σύνολο δεδομένων και στην συνέχεια, επιλέγοντας τυχαία σημεία δεδομένων K για τα κεντροειδή χωρίς αντικατάσταση.
- 4) Συνεχίζουμε την επανάληψη μέχρι να υπάρξει αλλαγή στα κεντροειδή, (Δεν αλλάζει η εκχώρηση των σημαδεμένων δεδομένων στο clustering).
 - Υπολογίζουμε το άθροισμα της τετραγωνικής Ευκλείδειας απόστασης μεταξύ των σημειακών δεδομένων και όλων των κεντροειδών.
 - Αντιστοιχούμε κάθε σημειακό δεδομένο στο πλησιέστερο cluster (Κεντροειδές).
 - Υπολογίζουμε τα κεντροειδή για τα clusters, λαμβάνοντας τον μέσο όρο όλων των σημειακών δεδομένων που ανήκουν σε κάθε cluster.

Η προσέγγιση KMeans που ακολουθεί έχει σκοπό την επίλυση του προβλήματος ονομάζεται «Μεγιστοποίηση Προσδοκίας». Το βήμα-Ε εκχωρεί τα σημειακά δεδομένα στο πλησιέστερο cluster. Το βήμα-Μ υπολογίζει το κεντροειδές κάθε cluster. Ακολουθεί μια ανάλυση του τρόπου με τον οποίο μπορούμε να το λύσουμε μαθηματικά.

Η αντικειμενική συνάρτηση είναι:

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \cdot ||x^i - \mu_k||^2$$

όπου $w_{ik} = 1$ για το σημείο δεδομένων x_i εάν ανήκει στο cluster k . Διαφορετικά, $w_{ik} = 0$. Επίσης, το μ_k είναι το κεντροειδές του cluster x_i .

Είναι ένα πρόβλημα ελαχιστοποίησης δύο μερών. Αρχικά ελαχιστοποιούμε το J σε σχέση με το w_{ik} και θεωρούμε το μ_k σταθερό. Στη συνέχεια ελαχιστοποιούμε το J σε σχέση με το μ_k και θεωρούμε το w_{ik} σταθερό. Από τεχνική άποψη, διαφοροποιούμε το J σε σχέση με το w_{ik} πρώτα και ενημερώνουμε τις αναθέσεις του cluster (βήμα-Ε). Τότε διαφοροποιούμε το J σε σχέση με το μ_k και υπολογίστε ξανά τα κεντροειδή μετά τις αντιστοιχίσεις του cluster από το προηγούμενο βήμα (βήμα-Μ). Επομένως, το βήμα-Ε είναι:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^m \sum_{k=1}^K \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1, & \text{εαν } k = \operatorname{argmin}_j ||x^i - \mu_j||^2 \\ 0, & \text{αλλιώς} \end{cases}$$

Με άλλα λόγια, αντιστοιχούμε το σημείο δεδομένων x_i στο πλησιέστερο cluster που κρίνεται από το άθροισμα της τετραγωνικής Ευκλείδειας απόστασης από το κεντροειδές του cluster.

Και το βήμα-M είναι:

$$\frac{\partial J}{\partial \mu_k} = 2 \cdot \sum_{i=1}^m w_{ik} \cdot (x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^m w_{ik} \cdot x^i}{\sum_{i=1}^m w_{ik}}$$

Αυτό μεταφράζεται σε υπολογισμό του κεντροειδούς κάθε cluster για να αντικατοπτρίζει τις νέες εργασίες.

Πρέπει να σημειωθεί ότι:

- Δεδομένου ότι οι αλγόριθμοι ομαδοποίησης, συμπεριλαμβανομένων των KMeans, χρησιμοποιούν μετρήσεις βάσει απόστασης για να προσδιορίσουν την ομοιότητα μεταξύ των σημειακών δεδομένων, συνιστάται η τυποποίηση των δεδομένων με μέση τιμή μηδέν και τυπική απόκλιση ενός δεδομένου ότι σχεδόν πάντα οι λειτουργίες σε οποιοδήποτε σύνολο δεδομένων θα έχουν διαφορετικές μονάδες μετρήσεων
- Δεδομένου της επαναληπτικής φύσης του KMeans και της τυχαίας αρχικοποίησης των κεντροειδών στην αρχή του αλγορίθμου, διαφορετικές αρχικοποιήσεις μπορεί να οδηγήσουν σε διαφορετικά clusters, καθώς ο αλγόριθμος KMeans μπορεί να κολλήσει σε ένα τοπικό βέλτιστο και μπορεί να μην συγκλίνει στο βέλτιστο σε γενικό επίπεδο. Επομένως, ο αλγόριθμος εκτελείται χρησιμοποιώντας διαφορετικές αρχικοποιήσεις κεντροειδών και γίνεται επιλογή από τα αποτελέσματα της εκτέλεσης που απέδωσαν το χαμηλότερο άθροισμα της τετραγωνικής Ευκλείδειας απόστασης.

Η αντιστοίχιση παραδειγμάτων δεν αλλάζει, είναι το ίδιο πράγμα με καμία αλλαγή στην παραλλαγή εντός cluster:

$$\frac{1}{m_k} \sum_{k=1}^{m_k} ||x^i - \mu_{c^k}||^2$$

Τέλος, για να επιτευχθεί ο στόχος μας γρηγορότερα γίνεται αποθήκευση ή φόρτωση του εκπαιδευμένου μοντέλου με τη χρήση της βιβλιοθήκης Pickle.

2.3. Αποτελέσματα

Στιγμιότυπο από το αποτέλεσμα στην κονσόλα:

```
TRAINING STARTED  
[!] Fitting KMeans [!]  
Fitting completed!  
KMeans saved!
```

Εικόνα 2.1: Εικόνα κονσόλας

3. ΥΠΟΕΡΩΤΗΜΑ ΤΡΙΤΟ

3.1. Ζητούμενα

Να γίνει κατάτμηση Εικόνας σε Superpixels σύμφωνα με τον αλγόριθμο SLIC.

3.2. Υλοποίηση

Για την υλοποίηση του αλγορίθμου **Simple Linear Iterative Clustering (SLIC)** χρησιμοποιήθηκε η βιβλιοθήκη **Skimage**, συγκεκριμένα το πακέτο **Segmentation**. Ο αλγόριθμος αυτός βασίζεται στον KMeans για να πραγματοποιήσει την κατάτμηση εικόνας. Αναλυτικότερα, εκτελείται KMeans σε πέντε διαστάσεις (**5D**) στο χώρο των πληροφοριών του χρώματος και της θέσης της εικόνας και συνεπώς σχετίζεται στενά με τη γρήγορη μετατόπιση (Quick-shift). Καθώς η μέθοδος ομαδοποίησης είναι απλούστερη, είναι πολύ αποτελεσματική. Είναι σημαντικό για αυτόν τον αλγόριθμο να εργαστεί στον χρωματικό χώρο LAB για να επιτύχει καλά αποτελέσματα. Η παράμετρος «**compactness**» ανταλλάσσει την ομοιότητα και την εγγύτητα χρώματος, όπως στην περίπτωση του Quick-shift, ενώ η «**n_segments**» επιλέγει τον αριθμό των κέντρων για KMeans.

Superpixels:

- Τα superpixels οργανώνουν το σύνολο των εικονοστοιχείων μιας εικόνας, λαμβάνοντας υπόψιν την έγχρωμη εκδοχή κάθε εικόνας, σε ένα σύνολο υποπεριοχών. Είναι λοιπόν μια άλλη διαδικασία clustering η οποία δεν εκτελείται στο σύνολο όλων των εικονοστοιχείων αλλά στο σύνολο των εικονοστοιχείων μιας εικόνας.
- Μπορούμε να προσδιορίσουμε το πλήθος των εικονοστοιχείων που περιλαμβάνει κάθε superpixel στον αλγόριθμο.

- Το κριτήριο για να ομαδοποιηθούν μαζί τα εικονοστοιχεία σε ένα superpixel είναι η ευκλείδεια απόσταση να είναι μικρότερη από κάποιο κατώφλι που εμείς ορίζουμε. Επίσης λαμβάνεται υπόψιν και η θέση του εικονοστοιχείου, δηλαδή στο ίδιο superpixel θα ανατεθούν εικονοστοιχεία παρεμφερή χρώματος αλλά θα βρίσκονται και σε γειτονικά σημεία της εικόνας.
- Θέλουμε τα όρια κάθε superpixel να μην διασχίζουν τα όρια 2 διαφορετικών αντικειμένων. Άρα το χρώμα του κεντρικού pixel του κάθε superpixel θα είναι και το χρώμα που θα πάρει και το σύνολο των εικονοστοιχείων εντός του κάθε superpixel.

Σημείωση: Δεν μπορούμε στην πραγματικότητα ακόμα και σε ένα αντικείμενο που οι ιδιότητες της επιφάνειάς του να του αποδίδουν ένα συγκεκριμένο χρώμα αλλά ο τρόπος που θα απεικονιστεί αυτό το χρώμα από τις συνθήκες φωτισμού αλλάζει. Άρα θέλουμε μέσα στα όρια του κάθε superpixel να ομαδοποιηθούν όλα εκείνα τα εικονοστοιχεία που αντανακλούν το φως με τον ίδιο τρόπο.

Η κατάτμηση της κάθε εικόνας σε Superpixels γίνεται με τις μεθόδους του αρχείου `slic_superpixels.py`. Στο αρχείο αυτό υπάρχει η συνάρτηση `calc_slic_superpixels(εικόνα,)`. Εμφανίζουμε το αποτέλεσμα με τη βοήθεια της συνάρτησης `mark_boundaries()`. Ο αλγόριθμος SLIC ομαδοποιεί μεμονωμένα pixels βάσει χρωματικής ομοιότητας και απόστασης στον χρωματικό χώρο LAB. Στο αρχείο `main.py` παίρνουμε την μέθοδο `get_slic_superpixels()` από το αρχείο `slic_superpixels` για να λάβουμε τα segments. Έπειτα τα κάνουμε «fit» στην βάση με την βοήθεια του SVM.

3.3. Αποτελέσματα

Εικόνα που αναπαριστά τα Superpixels:



Εικόνα 3.1: Εικόνα superpixels

4. ΥΠΟΕΡΩΤΗΜΑ ΤΕΤΑΡΤΟ

4.1. Ζητούμενα

Να γίνει εξαγωγή χαρακτηριστικών Υφής (SURF Features & Gabor Features) ανά Super Pixel.

4.2. Υλοποίηση

Κατά συνέπεια για την εξαγωγή των χαρακτηριστικών είναι ανάγκη να γίνει μετατροπή της εικόνας σε Grayscale μέσω του αρχείου `utilities.py`. Η μέθοδος `rgb2gray`(χρώματα RGB) μετατρέπει την αρχική μας εικόνα σε ασπρόμαυρη. Αυτό γίνεται καλώντας την εντολή `cv2.cvtColor` βάζοντας σαν ορίσματα τα χρώματα `rgb` που δόθηκαν και την ενσωματωμένη εντολή `cv2.COLOR_RGB2GRAY`, ο αλγόριθμος της εντολής λειτουργεί όπως αναφέρεται πιο κάτω.

Οι μετασχηματισμοί εντός χώρου RGB όπως προσθήκη/αφαίρεση του καναλιού άλφα, αντιστροφή της σειράς καναλιών, μετατροπή σε/από χρώμα RGB 16-bit (R5: G6: B5 ή R5: G5: B5), καθώς και μετατροπή σε/από κλίμακα του γκρι χρησιμοποιώντας:

$$\text{RGB}[A] \text{ σε Γκρί: } Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

και

$$\text{Γκρί σε RGB}[A]: R \leftarrow Y, G \leftarrow Y, B \leftarrow Y, A \leftarrow \max(\text{Εύρος Καναλιού})$$

για μετατροπή πίσω σε RGB.

Στην συνέχεια, η εξαγωγή των χαρακτηριστικών **SURF** (Speeded Up Robust Features) γίνεται από τις μεθόδους του αρχείου `surf_and_gabor_features.py`. Χρησιμοποιούμε την υλοποίηση από την βιβλιοθήκη `mahotas`.

Ο αλγόριθμος **SURF** λειτουργεί ως εξής:

Υπάρχουν δύο βασικά βήματα.

1. Ανίχνευση σημείων ενδιαφέροντος.
2. Περιγραφή των σημείων ενδιαφέροντος.

Η συνάρτηση `mahotas.features.surf.surf` συνδυάζει τα δύο βήματα. Λαμβάνοντας υπόψη τα αποτελέσματα, μπορούμε να πραγματοποιήσουμε ένα απλό clustering, δηλαδή να εξάγουμε τα χαρακτηριστικά **SURF** για κάθε superpixel και στην συνέχεια να γίνεται έλεγχος εάν το superpixel είναι Grayscale, και αν δεν είναι τότε το μετατρέπουμε με την χρήση της προηγούμενης συνάρτησης. Επίσης, η συνάρτηση `surf_and_gabor_features.extract_surf(εικόνα)` επιστρέφει ένα ndarray δεκαδικών αριθμών (double) με τα σημεία ενδιαφέροντος ολόκληρης της εικόνας. Αναλυτικότερα, ο αλγόριθμος SURF βασίζεται στις ίδιες αρχές και ακολουθεί τα ίδια βήματα με τον SIFT, αλλά οι λεπτομέρειες σε κάθε βήμα είναι διαφορετικές. Ο αλγόριθμος έχει τρία κύρια μέρη: ανίχνευση σημείων ενδιαφέροντος, περιγραφή τοπικής γειτονιάς και αντιστοίχιση.

1. Ανίχνευση

Το SURF χρησιμοποιεί φίλτρα τετράγωνου σχήματος ως προσέγγιση της εξομάλυνσης Gauss. Το φιλτράρισμα της εικόνας με ένα τετράγωνο είναι πολύ πιο γρήγορο εάν χρησιμοποιείται η ολοκληρωμένη εικόνα:

$$S(x, y) = \sum_{i=0}^x \sum_{j=0}^y I(i, j)$$

Το άθροισμα της αρχικής εικόνας σε ένα ορθογώνιο μπορεί να αξιολογηθεί γρήγορα χρησιμοποιώντας την ολοκληρωμένη εικόνα, απαιτώντας αξιολογήσεις στις τέσσερις γωνίες του ορθογωνίου.

Το SURF χρησιμοποιεί έναν ανιχνευτή blob με βάση τον πίνακα **Hessian** για να βρει σημεία ενδιαφέροντος. Ο καθοριστής της μήτρας Hessian χρησιμοποιείται ως μέτρο της τοπικής αλλαγής γύρω από το σημείο και επιλέγονται σημεία όπου αυτός ο καθοριστής είναι μέγιστος. Το SURF χρησιμοποιεί επίσης τον καθοριστικό παράγοντα του Hessian για την επιλογή της κλίμακας, δεδομένου του σημείου $p = (x, y)$ σε μια εικόνα I , ο πίνακας Hessian $H(p, \sigma)$ στο σημείο p και κλίμακα σ , είναι:

$$H(p, \sigma) = \begin{bmatrix} L_{xx}(p, \sigma) & L_{xy}(p, \sigma) \\ L_{yx}(p, \sigma) & L_{yy}(p, \sigma) \end{bmatrix}$$

Όπου $L_{xx}(p, \sigma)$ κλπ. είναι η συνέλιξη του παραγώγου δεύτερης τάξης του gaussian με την εικόνα $I(x, y)$ στο σημείο p .

Αναπαράσταση κλίμακας-χώρου και τοποθεσία των σημείων ενδιαφέροντος

Τα σημεία ενδιαφέροντος μπορούν να βρεθούν σε διαφορετικές κλίμακες, εν μέρει επειδή η αναζήτηση αντιστοιχιών απαιτεί συχνά εικόνες σύγκρισης όπου εμφανίζονται σε διαφορετικές κλίμακες. Οι εικόνες εξομαλύνονται επανειλημμένα με ένα φίλτρο **Gauss**, και στη συνέχεια υποβάλλονται σε δείγμα για να πάρουν το επόμενο υψηλότερο επίπεδο της πυραμίδας. Επομένως, υπολογίζονται αρκετά δάπεδα ή σκάλες με διάφορα μέτρα μάσκας:

$$\sigma = \text{το παρον μέγεθος του φίλτρου} \times \left(\frac{\text{βασική κλίμακα φίλτρου}}{\text{βασικό μέγεθος φίλτρου}} \right)$$

Ο χώρος κλίμακας χωρίζεται σε έναν αριθμό οκτάβων, όπου μια οκτάβα αναφέρεται σε μια σειρά χαρτών απόκρισης που καλύπτουν διπλασιασμό της κλίμακας. Στο SURF, το χαμηλότερο επίπεδο του χώρου κλίμακας λαμβάνεται από την έξοδο των φίλτρων 9×9 .

Ως εκ τούτου, οι χώροι κλίμακας στο **SURF** εφαρμόζονται εφαρμόζοντας φίλτρα κουτιού διαφορετικών μεγεθών. Κατά συνέπεια, ο χώρος κλίμακας αναλύεται με την αναβάθμιση του μεγέθους του φίλτρου αντί να μειώσει επαναληπτικά το μέγεθος της εικόνας. Η έξοδος του παραπάνω φίλτρου 9×9 θεωρείται ως το αρχικό επίπεδο κλίμακας στην κλίμακα $s = 1.2$ (αντιστοιχεί σε παραγώγους **Gauss** με $\sigma = 1.2$). Τα ακόλουθα επίπεδα επιτυγχάνονται με φιλτράρισμα της εικόνας με σταδιακά μεγαλύτερες μάσκες, λαμβάνοντας υπόψη τη διακριτή φύση των ολοκληρωμένων εικόνων και τη συγκεκριμένη δομή φίλτρου. Αυτό έχει ως αποτέλεσμα φίλτρα μεγέθους 9×9 , 15×15 , 21×21 , 27×27 , Η μη μέγιστη καταστολή σε μια γειτονιά $3 \times 3 \times 3$ εφαρμόζεται για τον εντοπισμό σημείων ενδιαφέροντος στην εικόνα και πάνω από τις κλίμακες. Στη συνέχεια, τα μέγιστα του καθοριστικού παράγοντα της μήτρας της Hessian παρεμβάλλονται σε κλίμακα και χώρο.

2. Περιγραφέας

Ο στόχος ενός περιγραφέα είναι να παρέχει μια μοναδική και ισχυρή περιγραφή ενός χαρακτηριστικού εικόνας, π.χ., περιγράφοντας την κατανομή έντασης των pixel εντός της γειτονιάς του σημείου ενδιαφέροντος. Οι περισσότεροι περιγραφείς υπολογίζονται έτσι με τοπικό τρόπο, επομένως λαμβάνεται μια περιγραφή για κάθε σημείο ενδιαφέροντος που προσδιορίστηκε προηγουμένως.

Το πρώτο βήμα συνίσταται στον καθορισμό ενός αναπαραγωγίμου προσανατολισμού που βασίζεται σε πληροφορίες από μια κυκλική περιοχή γύρω από το σημείο ενδιαφέροντος. Στη συνέχεια, κατασκευάζουμε μια τετραγωνική περιοχή ευθυγραμμισμένη με τον επιλεγμένο προσανατολισμό και εξάγουμε την περιγραφή **SURF** από αυτήν.

3. Αντιστοίχιση

Συγκρίνοντας τους περιγραφείς που λαμβάνονται από διαφορετικές εικόνες, μπορούν να βρεθούν ζευγάρια που ταιριάζουν.

Περίπου με τον ίδιο τρόπο θα γίνει και η εξαγωγή σε χαρακτηριστικά **Gabor** μέσω της βιβλιοθήκης `skimage.filters`. Η συνάρτηση `extract_gabor(εικόνα, συχνότητα=0,6)` επιστρέφει το «πραγματικό» φίλτρο **Gabor**. Η παράμετρος συχνότητα είναι η χωρική συχνότητα της αρμονικής λειτουργίας και καθορίζεται σε `pixel`, όσο πιο μεγάλη η συχνότητα τόσο πιο ακριβής η λεπτομέρεια. Τα «πραγματικά» και «φανταστικά» μέρη του πυρήνα φίλτρου **Gabor** εφαρμόζονται στην εικόνα και η απόκριση επιστρέφεται ως ζεύγος συστοιχιών. Η συνάρτηση `extract_gabor_for_each_superpixel(Λίστα με τα Superpixels, απρόμαυρο=False)` θα ελέγξει εάν κάθε `superpixel` είναι ασπρόμαυρο και εάν δεν είναι τότε θα το μετατρέψουμε, επιστρέφει το **Gabor** των `superpixels`.

4.3. Αποτελέσματα



Εικόνα 4.1: Ασπρόμαυρη εικόνα

```
[!] Extracting SURF features [!]  
SURF extraction finished!  
[!] Extracting Gabor features [!]  
Gabor extraction finished!
```

Εικόνα 4.2: Εικόνα κονσόλας

5. ΥΠΟΕΡΩΤΗΜΑ ΠΕΜΠΤΟ

5.1. Ζητούμενα

Να γίνει Εκμάθηση Τοπικών Μοντέλων Πρόγνωσης Χρώματος με Χρήση Ταξινομητών SVM.

5.2. Υλοποίηση

Το SVM (Support Vector Machine) προσφέρει πολύ υψηλή ακρίβεια σε σύγκριση με άλλους ταξινομητές, όπως η λογιστική παλινδρόμηση και τα δέντρα αποφάσεων. Αυτό το υποερώτημα υλοποιείται στο αρχείο `svm.py`. Το SVM που χρησιμοποιούμε είναι το SVC (Support Vector Classifier) με την χρήση «balanced» βαρών για την κάθε κλάση. Ο στόχος ενός Linear SVC είναι να ταιριάξει τα δεδομένα που του παρέχονται, επιστρέφοντας ένα "καλύτερα προσαρμοσμένο" υπερπλάνο που διαιρεί ή κατηγοριοποιεί τα δεδομένα. Από εκεί, αφού λάβουμε το υπερπλάνο, μπορούμε στη συνέχεια να τροφοδοτήσουμε ορισμένες δυνατότητες στον ταξινομητή μας για να δούμε ποια είναι η «προβλεπόμενη» τάξη. Αυτό καθιστά τον συγκεκριμένο αλγόριθμο αρκετά κατάλληλο. Για να κάνουμε fit το SVM, χρησιμοποιούμε το σύνολο των εικόνων που χρησιμοποιήσαμε και για τα KMeans.

Ως χαρακτηριστικά (features) χρησιμοποιήσαμε τα χαρακτηριστικά SURF και Gabor του κάθε superpixel της κάθε εικόνας, που υπολογίζονται ως ο μέσος όρος των αντίστοιχων χαρακτηριστικών των επιμέρους pixels. Στην συνέχεια, υπολογίζουμε το κυρίαρχο χρώμα του superpixel (κάνοντας predict το κάθε pixel με το εκπαιδευμένο KMeans μοντέλο μας και χρησιμοποιούμε το χρώμα που επιστρέφεται περισσότερες φορές) την ετικέτα του οποίου (από το εκπαιδευμένο KMeans) την χρησιμοποιούμε ως ετικέτα των χαρακτηριστικών για το συγκεκριμένο superpixel. Για την εξοικονόμηση χρόνου αποθηκεύουμε το εκπαιδευμένο μοντέλο μας με την βοήθεια της βιβλιοθήκης pickle. Η βιβλιοθήκη pickle είναι ικανή να μετατρέψει ένα python αντικείμενο σε σειρά από χαρακτήρες. Μετέπειτα όποτε χρειαζόμαστε το μοντέλο μας γίνεται απευθείας φόρτωση.

5.3. Αποτελέσματα

EXTERNAL DATA (G:) > Downloads (HDD) > University > 7th Semester > IMAGE RECOGNITION > ASSIGNMENTS > Assignment >				
Name	Date modified	Type	Size	
.git	25-Feb-21 11:49 PM	File folder		
.idea	26-Feb-21 12:24 AM	File folder		
__pycache__	25-Feb-21 11:39 PM	File folder		
assets	13-Feb-21 5:41 PM	File folder		
venv	06-Feb-21 5:39 PM	File folder		
.gitignore	05-Feb-21 7:46 PM	GITIGNORE File	4 KB	
fitted_kmeans.sav	25-Feb-21 11:39 PM	SAV File	1,537 KB	
image_segmentation.py	13-Feb-21 8:54 PM	PY File	1 KB	
main.py	25-Feb-21 2:52 PM	PY File	3 KB	
slic_superpixels.py	25-Feb-21 2:45 PM	PY File	2 KB	
surf_and_gabor_features.py	25-Feb-21 6:33 PM	PY File	2 KB	
SVC.sav	25-Feb-21 11:45 PM	SAV File	103 KB	
svm.py	25-Feb-21 6:58 PM	PY File	3 KB	
utilities.py	13-Feb-21 4:32 PM	PY File	2 KB	

Εικόνα 5.1: Εμφάνιση αρχείων



Εικόνα 5.2: Χρωματισμένη εικόνα RGB

6. ΥΠΟΕΡΩΤΗΜΑ ΕΚΤΟ

6.1. Ζητούμενα

Να γίνει Εκτίμηση Χρωματικού Περιεχομένου Ασπρόμαυρης Εικόνας με Χρήση Αλγορίθμων Κοπής Γραφημάτων.

6.2. Υλοποίηση

Ο αλγόριθμος κοπής γραφημάτων δημιουργεί ένα γράφο μεταξύ των superpixels όπου τα βάρη των συγκεκριμένων ακμών είναι ανάλογα της ευκλείδειας απόστασης του διανύσματος χαρακτηριστικών.

Στην ουσία ο αλγόριθμος προσπαθεί να μεταβάλει τις ετικέτες που έχει αποδώσει ο μηχανισμός ταξινόμησης που έχουμε χρησιμοποιήσει έτσι ώστε γειτονικά superpixel στο γράφο μας να έχουν μια ομαλότερη ανάθεση χρώματος.

- Τα βάρη των ακμών προκύπτουν στο πόσο κοντινά είναι τα διανύσματα του εκάστοτε κόμβου.

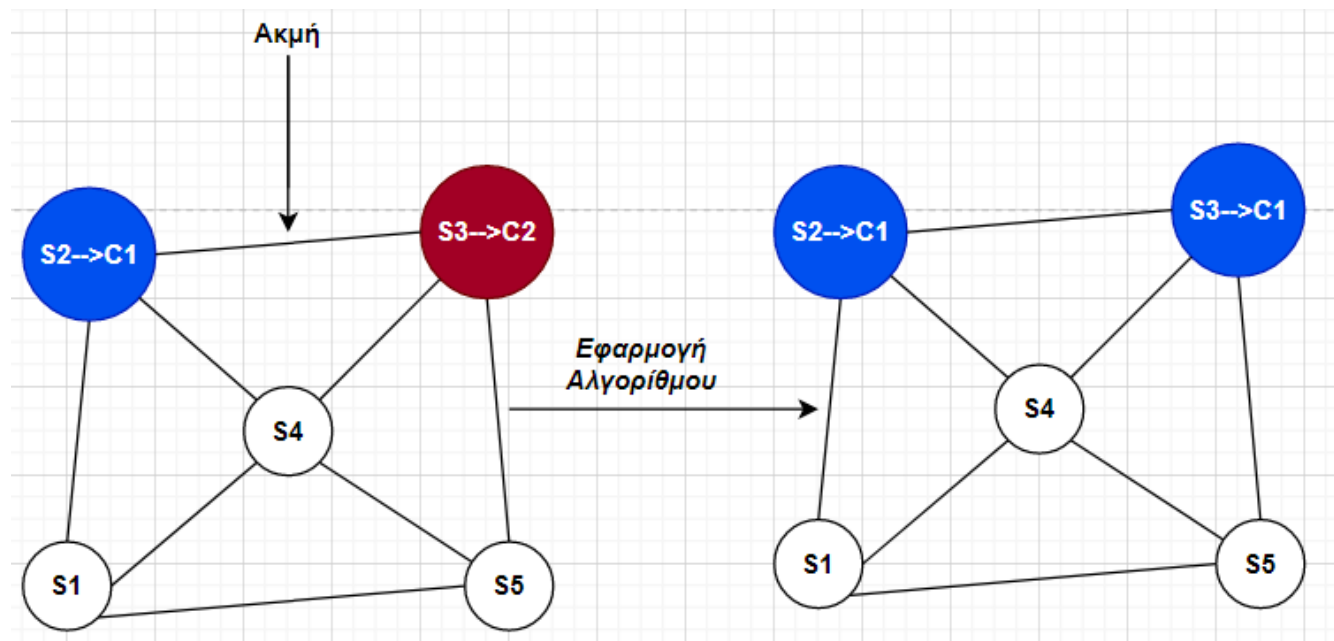
Το κόστος σύνδεσης (W_{ij}) είναι ανάλογο (\propto) της ευκλείδειας απόστασης του διανύσματος των χαρακτηριστικών για το superpixel i ($\vec{\varphi}(S_i)$) και j ($\vec{\varphi}(S_j)$).

$$W_{ij} \propto ||\vec{\varphi}(S_i) - \vec{\varphi}(S_j)||$$

Παράδειγμα αλγορίθμου:

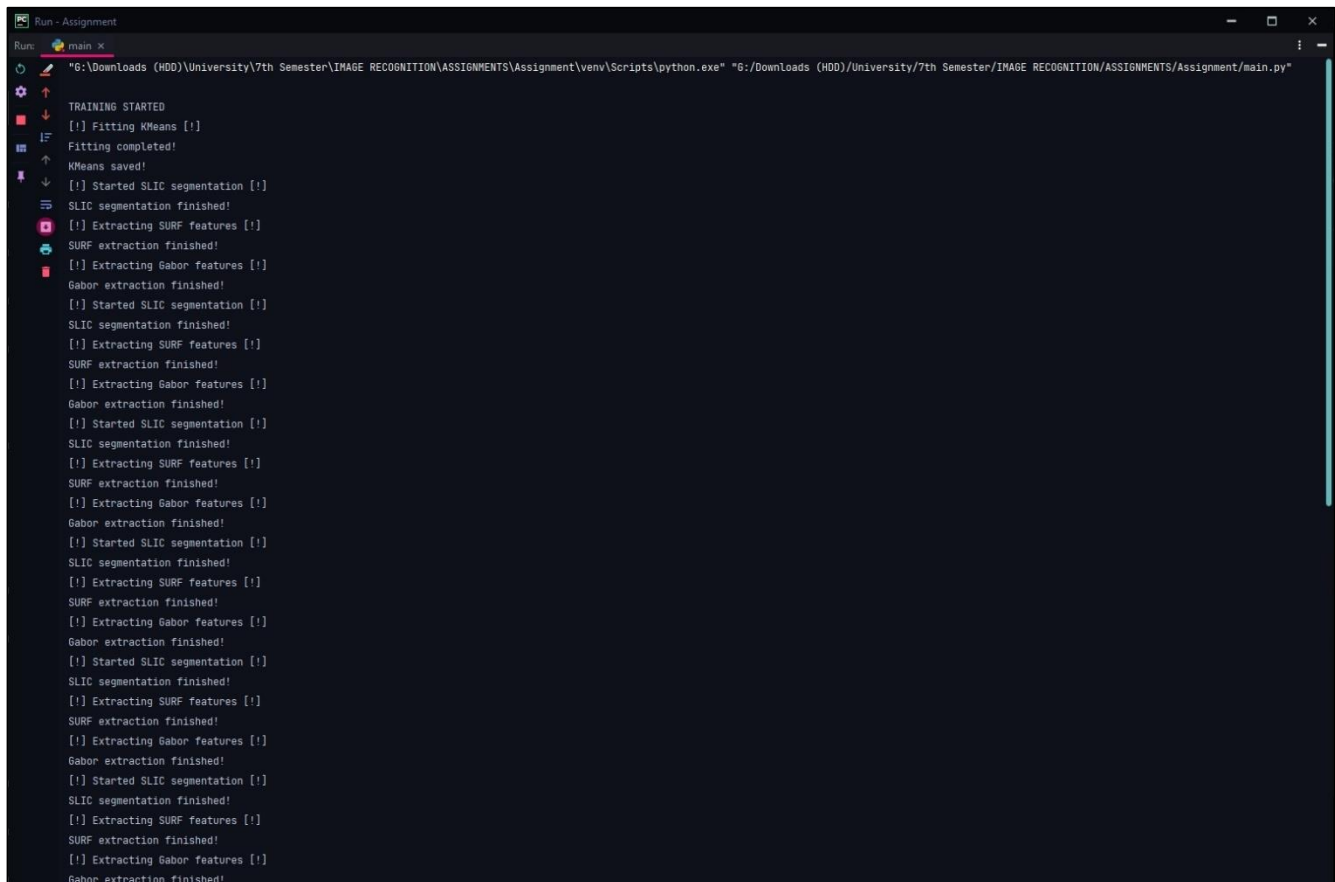
Αν ο αλγόριθμος έχει αποδώσει το χρώμα στο S_2 να είναι το C_1 και στο S_3 να είναι το C_2 και η απόσταση που έχει η ακμή που συνδέει τα superpixels είναι μικρή θα κάνει μία μικρή διορθωτική κίνηση και θα αποδώσει το ίδιο χρώμα σε αυτά τα δύο γειτονικά (γειτονικά στον γράφο) superpixels.

Σημείωση: Το πόσο μικρή θα πρέπει να είναι η απόσταση μεταξύ τους είναι δική μας σχεδιαστική απόφαση.



Εικόνα 6.1: Παράδειγμα αλγορίθμου

7. ΑΠΟΤΕΛΕΣΜΑ ΚΟΝΣΟΛΑΣ



```
Run - Assignment
main x
"G:\Downloads (HDD)\University\7th Semester\IMAGE RECOGNITION\ASSIGNMENTS\Assignment\env\Scripts\python.exe" "G:/Downloads (HDD)/University/7th Semester/IMAGE RECOGNITION/ASSIGNMENTS/Assignment/main.py"

TRAINING STARTED
[!] Fitting KMeans [!]
Fitting completed!
KMeans saved!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
```

Εικόνα 7.1: Εικόνα κονσόλας

[illegible]

```
TESTING STARTED
[!] Started SLIC segmentation [!]
SLIC segmentation finished!
[!] Extracting SURF features [!]
SURF extraction finished!
[!] Extracting Gabor features [!]
Gabor extraction finished!
[!] Preparing testing samples [!]
Testing samples have been prepared!
Colors have been predicted successfully!
```

8. ΒΙΒΛΙΟΓΡΑΦΙΑ

- Π1. Σημειώσεις μαθήματος.
- Π2. https://repository.kallipos.gr/bitstream/11419/3491/1/02_chapters_02.pdf
- Π3. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html
- Π4. <https://stackoverflow.com/questions/52767317/how-to-convert-rgb-image-pixels-to-lab>
- Π5. <https://www.xrite.com/blog/lab-color-space>
- Π6. https://web.archive.org/web/20120502065620/http://cookbooks.adobe.com/post_Useful_color_equations_RGB_to_LAB_converter-14227.html
- Π7. <https://kb.osu.edu/handle/1811/76395>
- Π8. http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/attachments/Colorization_main_6334%5b0%5d.pdf
- Π9. <https://cs.uwaterloo.ca/~zfrenett/CS886-Project.pdf>
- Π10. <https://www.vision.ee.ethz.ch/~surf/eccv06.pdf>
- Π11. <https://people.cs.clemson.edu/~jzwang/ustc13/mm2012/p369-gupta.pdf>
- Π12. <https://scikit-image.org/docs/dev/api/skimimage.filters.html#skimimage.filters.gabor>
- Π13. https://www.researchgate.net/figure/Gabor-Filter-Algorithm_fig3_284148891
- Π14. https://en.wikipedia.org/wiki/Speeded_up_robust_features
- Π15. <https://core.ac.uk/download/pdf/82820155.pdf>
- Π16. https://www.researchgate.net/figure/Main-stages-of-the-SURF-algorithm_fig1_261421604
- Π17. ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ (2012) – S. Theodoridis, K. Koutroumbas.

9. DEPENDENCIES/ΕΡΓΑΛΕΙΑ

- Python v3.9
- PyCharm Professional v.2020.3.3
- Photoshop CC

9.1. Βιβλιοθήκες

- OpenCV
- Numpy
- SciKit-Learn.svm (SVC)
- SciKit-Learn.cluster (KMeans)
- SkImage.filters (Gabor)
- SkImage.segmentation (SLIC, Mark Boundaries)
- Mahotas.features (Surf)
- OS.Path
- Pickle
- Glob