

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ**  
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**



**Ανάλυση Εικόνας**  
**Απαλλακτική Εργασία**  
*Grayscale Image Colorization*

Όνομα, Επώνυμο - Αριθμός Μητρώου	Σκάρος Γεώργιος	Π15128
Ημερομηνία παράδοσης	08/05/2020	

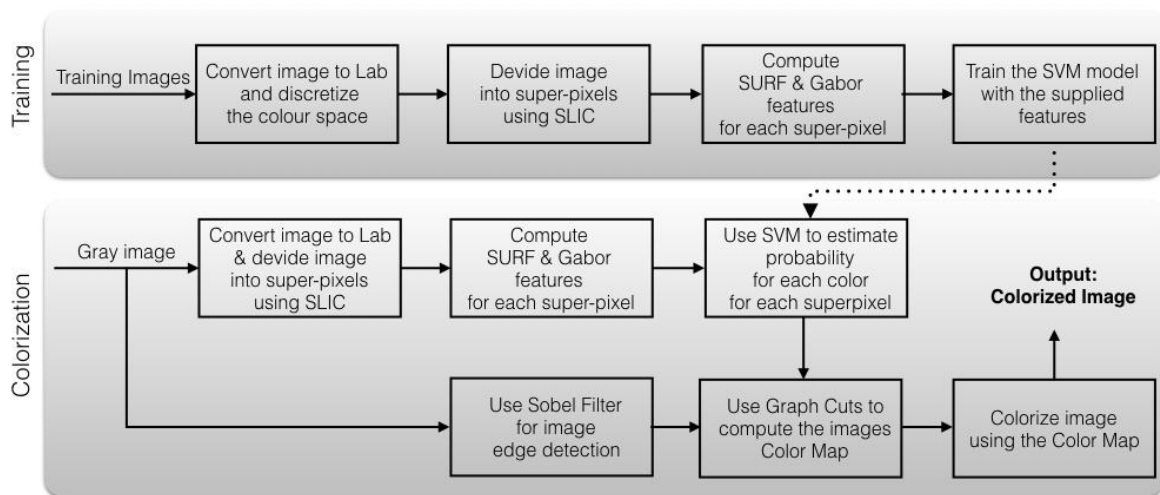
# Περιεχόμενα

Περιεχόμενα	2
Εισαγωγή	3
Απαραίτητα εργαλεία και βιβλιοθήκες	3
Project files	4
Αυτοματοποιημένος χρωματισμός εικόνας	5
RGB-LAB χρωματικοί χώροι	5
Διακρητοποίηση χρωματικού χώρου	6
Simple Linear Iterative Clustering (SLIC) algorithm	7
How does SLIC works	7
Εξαγωγή χαρακτηριστικών	8
SURF (Speeded Up Robust Features)	8
Gabor Features	9
Multilabel Classification with SVM	10
Multiclass classification to Binary Classification	11
Graph Cut Optimization	12
Sobel filter	12
Τελικά αποτελέσματα	14
Test Case 1	14
Test Case 2	15
Εκτέλεση	16
Εκτέλεση κώδικα	16
Training of classifier	16
With and without Graph Cuts	17
Βιβλιογραφία	18

# Εισαγωγή

Για την απαλλακτική εργασία του μαθήματος Ανάλυση Εικόνας καλούμαστε να δημιουργήσουμε μια σειρά αλγοριθμικών διαδικασιών με σκοπό τον αυτόματο χρωματισμό μιας ασπρόμαυρης εικόνας. Δεδομένου ότι τα βήματα για την δημιουργία του συνολικού προγράμματος περιγράφονται στην εκφώνηση της εργασίας θα προχωρήσουμε στην θεωρητική και πρακτική ανάλυση τους στις επόμενες ενότητες.

Η δομή του συνολικού προγράμματος μπορεί να αναπαρασταθεί με την χρήση του διαγράμματος που ακολουθείται:



## Απαραίτητα εργαλεία και βιβλιοθήκες

Για την υλοποίηση της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού python 3 με την χρήση των εξής βιβλιοθηκών (libraries):

Libraries	
Name	Version
sklearn (scikit-learn)	0.21.3
numpy	1.17.2
skimage (scikit-image)	0.15.0
cv2 (opencv-contrib-python)* *downgraded because of patented	3.4.2.17

algorithms (SURF & GABOR) in the current version	
matplotlib	3.1.1
joblib	0.13.2
pandas	0.25.1

## Project files

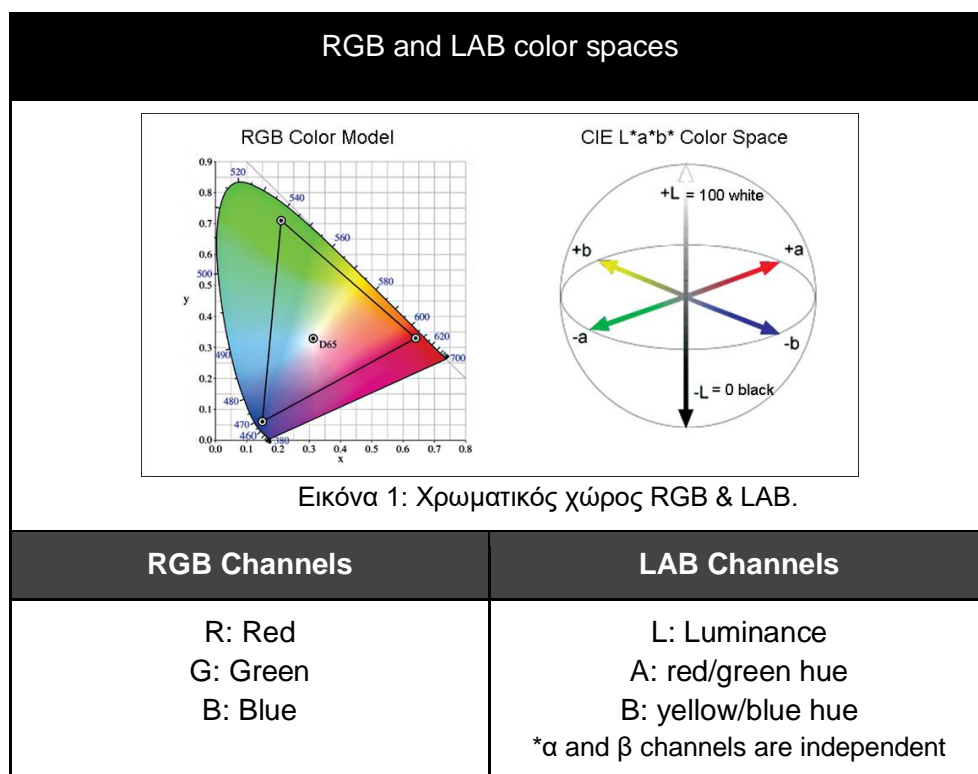
Name	Use
<b>Python files</b>	
ImageProcessor.py	Contains all the needed image processing utilities
ImageColorizer.py	Module for user-program interaction
<b>Text files</b>	
READ_ME.txt	Instructions on how to execute the program

# Αυτοματοποιημένος χρωματισμός εικόνας

Προκειμένου να χρωματίσουμε μια ασπρόμαυρη εικόνα, με βάση ένα σύνολο συναφών εικόνων, χρειαζόμαστε δεδομένα όσον αφορά τα **χαρακτηριστικά υφής** των εικόνων προκειμένου να μπορέσουμε να τα συσχετίσουμε με αυτά της ασπρόμαυρης εικόνας, ώστε να την χρωματίσουμε κατάλληλα. Η διαδικασία αυτή θα αυτοματοποιηθεί με την χρήση ταξινομητών **SVM**, με την βοήθεια των οποίων θα αναλύσουμε την πιθανότητα εμφάνισης κάθε χρώματος και στην συνέχεια θα επιλέξουμε το κατάλληλο με την **χρήση αλγορίθμων κοπής γραφημάτων** διατηρώντας την συνέχεια των χρωμάτων (**global coherence**). Προκειμένου να χρωματίσουμε την εικόνα χωρίς να μεταβάλουμε την φωτεινότητα της θα μετατρέψουμε τον χρωματικό χώρο από RGB σε **LAB**. Τέλος αξίζει να σημειωθεί ότι θα γίνει η εφαρμογή του αλγορίθμου **SLIC** για την ομαδοποίηση γειτονικών pixels σε superpixels και **διακριτοποίηση του χρωματικού χώρου** προκειμένου να επιταχύνουμε την όλη διαδικασία.

## RGB-LAB χρωματικοί χώροι

Αρχικά χρειάζεται να μετατρέψουμε τις εικόνες στον χρωματικό χώρο LAB προκειμένου να διατηρήσουμε τα δεδομένα που αφορούν τα χρώματα (color) της κάθε εικόνας με την φωτεινότητα (luminosity). Έχοντας ένα κανάλι για την φωτεινότητα (L) και 2 χρώματος (A,B) μπορούμε να επεξεργαστούμε τα χρώματα μιας εικόνας χωρίς να αλλάξουμε την φωτεινότητα του κάθε pixel. Χρησιμοποιώντας αυτόν τον χρωματικό χώρο μειώνουμε και τα δεδομένα τα οποία θα κληθεί να αναλύσει ο αλγόριθμος στην συνέχεια.



Επιπλέον το διάνυσμα (vector)  $\langle \alpha, \beta \rangle$  έχει την ιδιότητα να αναπαριστά την Ευκλείδεια απόσταση μεταξύ δύο χρωμάτων, η οποία είναι ανάλογη της αντίληψης των ανθρώπων για τις διακυμάνσεις των χρωμάτων (χρωματικές αλλαγές), που θα μας χρειαστεί στα μετέπειτα στάδια.

Προκειμένου να υλοποιήσουμε την συγκεκριμένη λειτουργία θα χρησιμοποιήσουμε την συνάρτηση **rgb2lab** της βιβλιοθήκης: **skimage**, η οποία μετατρέπει μια εικόνα απο τον χρωματικό χώρο RGB σε CIE-LAB, μέσω της εξής διαδικασίας:

Skimage rgb2lab conversion
<ol style="list-style-type: none"><li>1. RGB to XYZ</li><li>2. XYZ to LAB</li></ol> <p>*Η διαδικασία είναι αντίστοιχη της απευθείας μετατροπής, ωστόσο η βιβλιοθήκη χρησιμοποιεί την ενδιάμεση κατάσταση XYZ, για εξοικονόμηση χώρου (η μετατροπή XYZ-LAB έχει υλοποιηθεί πριν της RGB-LAB)</p>

Τέλος αξίζει να σημειωθεί ότι για την αναπαράσταση της ασπρόμαυρης εικόνας στον χρωματικό χώρο LAB θα χρησιμοποιηθεί μόνο το κανάλι L, εφόσον η εικόνα δεν έχει χρωματικό περιεχόμενο, έτσι η εξαγωγή χαρακτηριστικών για τις ασπρόμαυρες εικόνες θα περιοριστεί σε αυτό το κανάλι, και ο χρωματισμός της στα κανάλια A, B.

## Διακριτοποίηση χρωματικού χώρου

Προκειμένου να επιλέξουμε τα κατάλληλα χρώματα κατά την διαδικασία χρωματισμού της εικόνας (τελικό στάδιο), πρέπει να επιλέξουμε ένα υποσύνολο του χρωματικού χώρου, που ορίζεται απο τις εικόνες εκπαίδευσης (training set). Για να είναι το υποσύνολο που θα επιλέξουμε αποδοτικό είναι αναγκαίο να πληροί τις εξής προϋποθέσεις:

Color space discretization Requirements
<p>Πρέπει να έχει την δυνατότητα να αναπαριστά:</p> <ol style="list-style-type: none"><li>1. όλα τα αντικείμενα στο τοπίο που απεικονίζεται στις εικόνες εκπαίδευσης</li><li>2. μικρές μεταβολές ανάμεσα στα χρώματα που εμφανίζονται στις εικόνες</li></ol>

Η διακριτοποίηση του χρωματικού χώρου LAB θα γίνει με την χρήση του αλγόριθμου Mini-Batch K-Means (clustering algorithm) για τα κανάλια χρώματος (A, B), δεδομένου ότι το κανάλι L δεν περιέχει πληροφορίες σχετικές με τα χρώματα της εικόνας. Ο αλγόριθμος αποτελεί παραλλαγή του γνωστού αλγορίθμου K Means, διότι χρησιμοποιεί mini-batches προκειμένου να μειώσει την χρονική πολυπλοκότητα του αλγόριθμου. Ουσιαστικά αποτελούνται από μικρά υποσύνολα των εισαγόμενων δεδομένων, στην περίπτωση μας των εικόνων εκπαίδευσης, τα οποία επιλέγονται με ψευδοτυχαία σειρά σε κάθε επανάληψη του αλγορίθμου.

Για την υλοποίηση του αλγορίθμου χρησιμοποιήσαμε από την βιβλιοθήκη: **sklearn.cluster** την συνάρτηση **MiniBatchKMeans** με της εξής παραμέτρους

MiniBatchKMeans		
Argument	Value	Use
n_clusters	8	Αριθμός συνόλων για την ομαδοποίηση των χρωμάτων
random_state	100	Seed για την τυχειότητα ως προς την αρχικοποίηση των κεντροειδών
compute_labels	True	-

Ο κύριος λόγος που εφαρμόζουμε την διακριτοποίηση είναι προκειμένου να έχουμε ένα τετριμμένο σύνολο χρωματικών επιλογών, η οποίες θα χρησιμοποιηθούν στην συνέχεια από ταξινομητή σαν πιθανές κλάσεις. Σαν αποτέλεσμα αυτού έχουμε και την μείωση της χρονικής πολυπλοκότητας του χρωματισμού της εικόνας.

Αξίζει να αναφερθεί ότι υπάρχει η δυνατότητα να επαναφέρουμε μέρος των χρωμάτων μετά την επιλογή τους, ωστόσο δεν θα αναλυθεί περαιτέρω σε αυτήν την εργασία εφόσον δεν ζητείται.

## Simple Linear Iterative Clustering (SLIC) algorithm

Η υπολογιστική πολυπλοκότητα που απαιτείται για την ανάλυση του κάθε εικονοστοιχείου (pixel) είναι αρκετά υψηλή, ωστόσο μπορούμε να μειώσουμε την πολυπλοκότητα χωρίς να μειώσουμε την αποδοτικότητα του προγράμματος με την ομαδοποίηση pixels. Ο αλγόριθμος SLIC θα χρησιμοποιηθεί για την κατάτμηση της εικόνας σε Superpixels. Κατα αυτόν τον τρόπο μπορούμε να ομαδοποιήσουμε γειτονικά pixels με κοινά χαρακτηριστικά και στην συνέχεια να εργαστούμε σε αυτά για τον χρωματισμό της εικόνας.

### How does SLIC works

Ο αλγόριθμος εκτελεί μια τοπική ομαδοποίηση εικονοστοιχείων (pixels) σε χώρο 5-D που ορίζεται από τις τιμές L, a, b του χώρου χρώματος CIE-LAB και τις συντεταγμένες x, y των εικονοστοιχείων.

5-D Space	
(L,a,b,x,y)	
L: Luminance	
a: red/green hue	
b: yellow/blue hue	
x: x-coordinate	
y: y-coordinate	

Ο τρόπος που μετράει την απόσταση, προκειμένου να ομαδοποιήσει τα εικονοστοιχεία, επιτρέπει στον αλγόριθμο να λειτουργεί με υψηλή απόδοση ακόμα και σε ασπρόμαυρες εικόνες. Δεδομένου ότι οι ασπρόμαυρες εικόνες δεν περιέχουν δεδομένα στα κανάλια  $\alpha$  &  $\beta$  ο αλγόριθμος βασίζεται κυρίως στο κανάλι  $L$  καθώς και την θέση του κάθε εικονοστοιχείου.

Για την ομαδοποίηση των pixels ο αλγόριθμος ξεκινάει με προυπολογισμένα κέντρα συμπλέγματος (cluster centers) και στην συνέχεια εφαρμόζει μια παρόμοια τεχνική με τον αλγόριθμο K-Means προκειμένου να ομαδοποιήσει τα στοιχεία βάση των κεντροειδών. Η διαδικασία επαναλαμβάνεται έως ότου να υπάρξει σταθερότητα (να μην μεταβάλλονται οι θέσεις μετά από μία επανάληψη)

## Εξαγωγή χαρακτηριστικών

Προκειμένου να μπορέσουμε να χρωματίσουμε μια ασπρόμαυρη εικόνα χρειαζόμαστε σχετικά χαρακτηριστικά με αυτά των εικόνων εκπαίδευσης. Ωστόσο μια τέτοια εικόνα περιέχει πληροφορία μόνο στο κανάλι  $L$ , η οποία δεν αρκεί για την εύρεση του κατάλληλου χρώματος. Ως αποτέλεσμα αυτού καλούμαστε να εξάγουμε χαρακτηριστικά από όλες τις εικόνες (εκπαίδευσης και μη). Για την εξαγωγή των χαρακτηριστικών της κάθε εικόνας θα ακολουθήσουμε την εξής διαδικασία:

Feature Extraction Process
<ol style="list-style-type: none"><li>1. Load the pre-calculated color palette</li><li>2. For each train image:<ol style="list-style-type: none"><li>a. Convert the train image to LAB</li><li>b. Split the train image to superpixels using SLIC</li><li>c. For each superpixel:<ol style="list-style-type: none"><li>i. Extract the colors (A &amp; B)</li><li>ii. Find the dominant color*</li><li>iii. Extract the SURF features</li><li>iv. Extract the Gabor features</li><li>v. Associate the final feature vector {SURF, Gabor} with the corresponding color class of the dominant color in the color palette</li></ol></li></ol></li><li>3. Save the feature vectors for future use</li></ol>
*The extraction of the dominant color is part of the discretization of the color space.

Στην συνέχεια θα αναλύσουμε τα χαρακτηριστικά που θα χρησιμοποιήσουμε για την ανάλυση των εικόνων, τα οποία είναι τα SURF και Gabor.

## SURF (Speeded Up Robust Features)

Ο αλγόριθμος SURF αποτελεί έναν τοπικό ανιχνευτή χαρακτηριστικών και μπορεί να χρησιμοποιηθεί για την εύρεση αντικειμένων, καθώς και την ομαδοποίηση τους μεταξύ



άλλων, σε μια εικόνα, βοηθώντας μας κατα αυτόν τον τρόπο να διαφοροποιήσουμε αντικείμενα στην εικόνα με κοινά χαρακτηριστικά όσον αφορά των χρωματισμό τους.

Για την εξαγωγή των SURF Features θα χρησιμοποιήσουμε τις εξής παραμετροποιήσεις στον αλγόριθμο:

SURF features algorithm arguments library in use: cv2		
Argument Name	Use	Value
hessianThreshold	Threshold for hessian matrix keypoint detector used in SURF	400
nOctaves	Number of pyramid octaves the keypoint detector will use	3

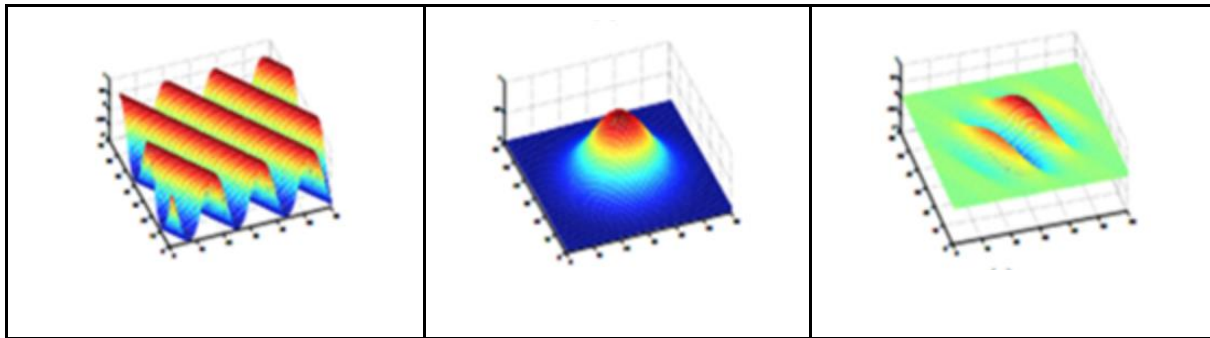
## Gabor Features

Τα Gabor φίλτρα, από τα οποία εξάγονται τα ζητούμενα χαρακτηριστικά, είναι γραμμικά φίλτρα που χρησιμεύουν στην ανάλυση εικόνων με διάφορες εφαρμογές όπως:

Gabor filter example uses
edge detection texture analysis feature extraction

Αυτά τα φίλτρα ανήκουν στην κατηγορία των bandpass φίλτρων και μπορούν να αναλυθούν ως:

Gabor filter break down		
<b>Ημιτονοειδές σήμα</b> (με συγκεκριμένη συχνότητα και προσανατολισμό) διαμορφωμένο από ένα Gaussian κύμα.		
Example		
Ημιτονοειδές σήμα (προσανατολισμός: 30 x-axis)	2D Gaussian κύμα	Παραγόμενο 2D Gabor filter



Για την εξαγωγή των Gabor Features θα χρησιμοποιήσουμε τις εξής παραμετροποιήσεις στον αλγόριθμο:

Gabor features algorithm arguments Library in use: cv2		
Argument Name	Use	Value(s)
theta_range	Orientation of the normal to the parallel stripes	$[0, \pi/6, \pi/4, \pi/3, \pi/2, 2\pi/3, 3\pi/4, 5\pi/6]$
scale_range (sigma)	Standard deviation of the gaussian function	[3, 6, 13, 28, 58]
kernel size	Size of gabor filter (n, n)	(20, 20)
lambda	Wavelength of the sinusoidal factor	7
gamma	Spatial aspect ratio	0.9
psi	Phase offset	1.5
ktype	Type and range of values that each pixel in the gabor kernel can hold	cv2.CV_32F $\rightarrow$ 0.0 - 1.0

## Multilabel Classification with SVM

Το σύστημα ταξινόμησης που θα χρησιμοποιήσουμε αποτελείται από ένα σύνολο ταξινομητών SVM. Οι ταξινομητές αυτού του είδους εκτελούν δυαδική ταξινόμηση (binary classification), για αυτόν τον λόγο είναι απαραίτητο να χρησιμοποιήσουμε παραπάνω από έναν ταξινομητή για να αντιμετωπίσουμε το πρόβλημα του Multilabel Classification, που έγκειται στην ταξινόμηση των χρωμάτων των superpixels σε ένα από τα διακριτά χρώματα που έχουμε επιλέξει παραπάνω.

## Multiclass classification to Binary Classification

Προκειμένου να μετατρέψουμε ένα multiclass πρόβλημα σε binary μπορούμε να χρησιμοποιήσουμε μία από τις δύο τεχνικές που αναφέρονται στην συνέχεια.

Multiclass classification with SVM decision function shape	
One-vs-Rest classifiers	One-vs-One classifiers
(n_samples, n_classes)	(n_samples, n_classes * (n_classes - 1) / 2)

Ο αλγόριθμος SVM εφαρμόζεται με την χρήση της βιβλιοθήκης sklearn με την παρακάτω παραμετροποίηση:

SVM algorithm arguments library in use: sklearn.svm function in use: SVC		
Argument Name	Use	Value
kernel	kernel type in use	rbf
gamma	Kernel coefficient	.25
cache_size	Size of the kernel cache	700
random_state	seed of the pseudo random number generator	110

Έχοντας ορίσει και παραμετροποιήσει καταλλίλος τον ταχηνομιτή SVM, στην συνέχεια πρέπει να προχωρήσουμε στην δημιουργία του μοντέλου OVR, βάση του οποίου θα χρησιμοποιήσουμε τον ίδιο αριθμό ταξινομητών με αυτών των κλάσεων χρωμάτων προκειμένου να βρούμε την πιθανότητα για το κάθε superpixel να είναι ένα συγκεκριμένο χρώμα. Η βιβλιοθήκη που χρησιμοποιήσαμε για την υλοποίηση του SVM μας παρέχει αντίστοιχη συνάρτηση ώστε να εφαρμόσουμε το μοντέλο που επιθυμούμε.

OVR Model library in use: sklearn.multiclass function in use: OneVsRestClassifier (The classifier uses the preconfigured SVM)		
Argument Name	Use	Value
n_jobs	Number of jobs to use for the computation	-1 (use all processors to speed up the process of training)

## Graph Cut Optimization

Μέχρι στιγμής όλες οι διαδικασίες που αναλύσαμε εφαρμόζονταν και κατά το στάδιο της εκπαίδευσης αλλά και της πρόβλεψης. Η εφαρμογή των αλγορίθμων κοπής γραφημάτων θα περιοριστεί στο τελευταίο στάδιο, δηλαδή της πρόβλεψης χρωμάτων. Η εφαρμογή του αλγόριθμου έχει ως στόχο την βελτίωση της συνοχής του χρωματικού περιεχομένου της εικόνας (global coherence). Όπως θα δούμε στην συνέχεια λαμβάνει υπόψη και τα γειτονικά superpixels κατά την επιλογή του κάθε χρώματος.

### Problems alleviated with the use of Graph Cuts

1. Multicolor selection over the same region due to use of many SVMs
2. Expand local coherence to global coherence

Προκειμένου να εκτιμήσουμε το χρωματικό περιεχόμενο της ασπρόμαυρης εικόνας θα εκτελέσουμε τις εξής διαδικασίες:

### Image Colorization process

1. Divide the image into superpixels (using SLIC algorithm)
2. Extract SURF and Gabor features for every superpixel
3. Compute the probability for every color for every super pixel using SVMs
4. Compute image edges using Sobel filter
5. Using graph cut compute the final colorization
6. Colorize the image

## Sobel filter

Η χρήση του συγκεκριμένου φίλτρου γίνεται προκειμένου να υπολογίσουμε τις γωνίες (edges) που περιέχονται στην εικόνα. Κατά αυτόν τον τρόπο μπορούμε να εκτιμήσουμε την θέση του περιεχομένου της εικόνας διαχωρίζοντας έτσι αντικείμενα ανεξάρτητα μεταξύ τους. Ως αποτέλεσμα αυτού ο αλγόριθμος Graph Cut μπορεί να εκτιμήσει με μεγαλύτερη ακρίβεια ποια γειτονικά σημεία ανήκουν στο ίδιο πεδίο με το σημείο προς εξέταση.

Εφόσον έχουμε υπολογίσει τις γωνίες (edges) της εικόνας, καθώς και τις πιθανότητες εμφάνισης του κάθε χρώματος ανά superpixel, μπορούμε να προχωρήσουμε στην δημιουργία του χρωματικού χάρτη (Colormap) της εικόνας προς χρωματισμό, με την χρήση των Graph Cuts και στην συνέχεια να την χρωματίσουμε.

**Graph Cut algorithm arguments**

[Library in use: pygco]

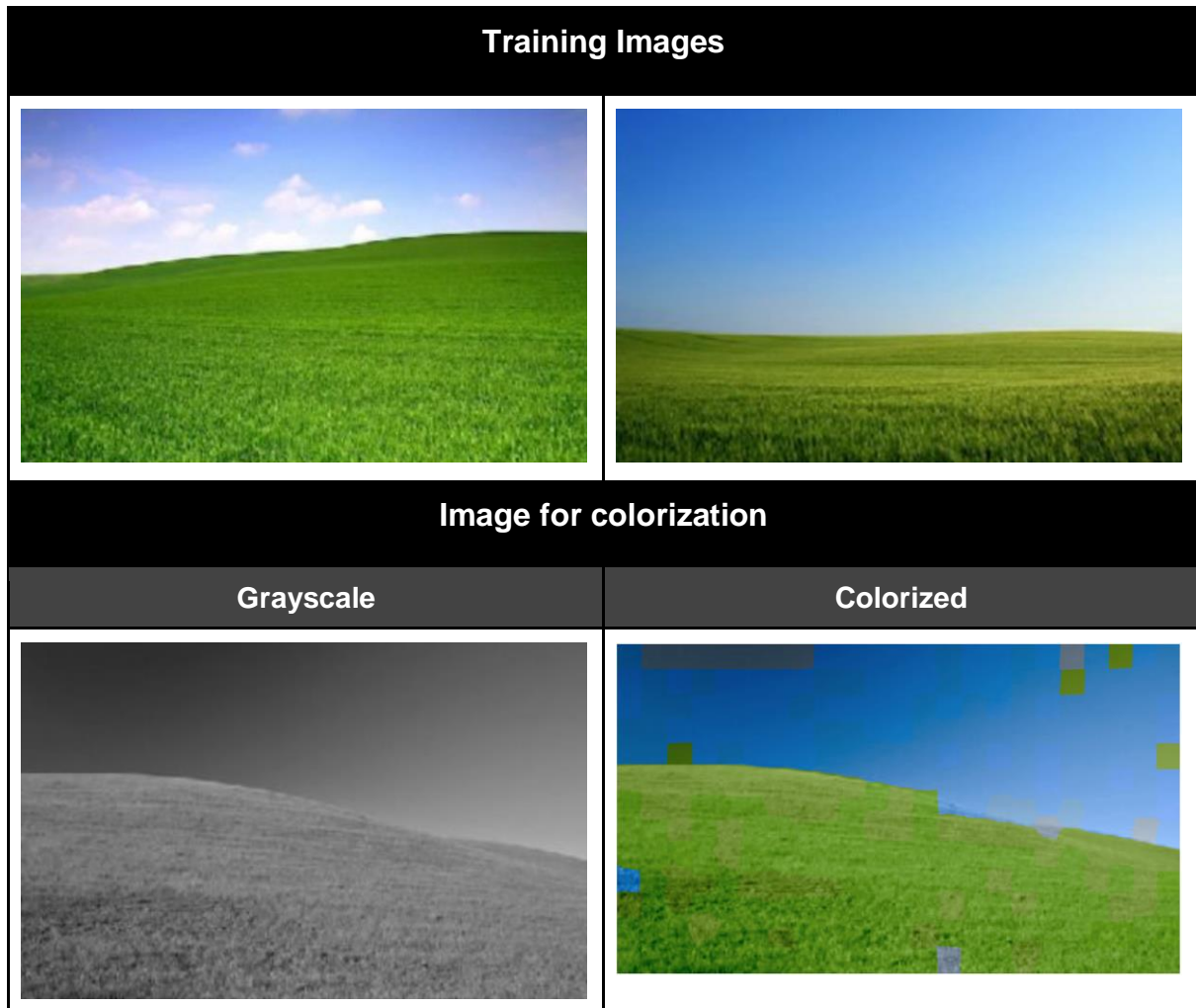
[Function in use: cut\_simple\_vh]

Argument Name	Use	Value
label_costs_int32	Precalculated cost with SVM classification	
pairwise_costs_int32	Precalculated cost for each pair of color classes	
edgesY_int32	Precalculated edges with Sobel algorithm	
edgesX_int32	Precalculated edges with Sobel algorithm	
n_iter	Number of iterations	10
algorithm	algorithm in use for energy minimization	swap

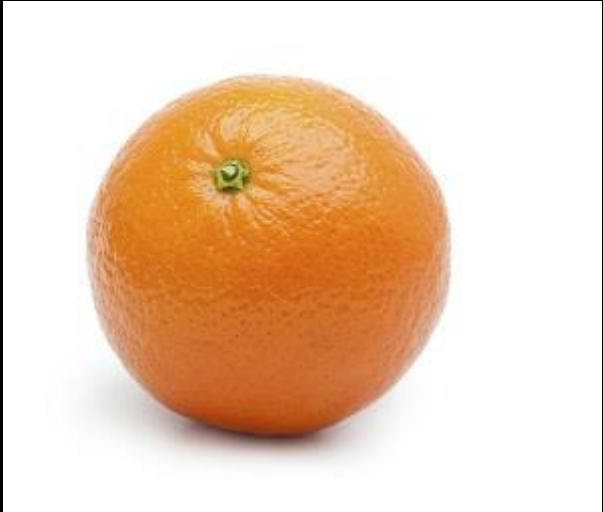
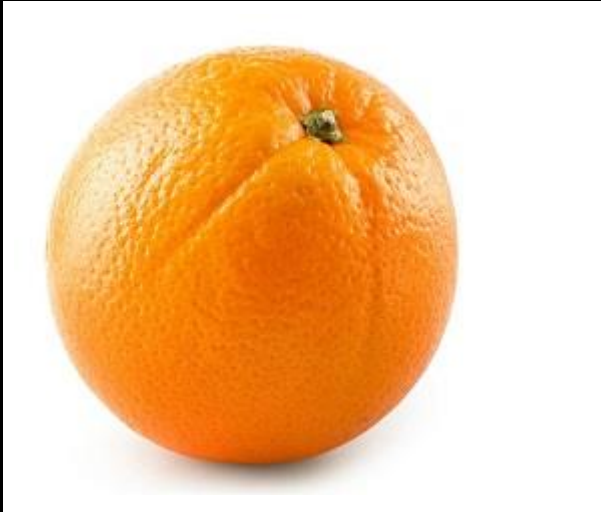
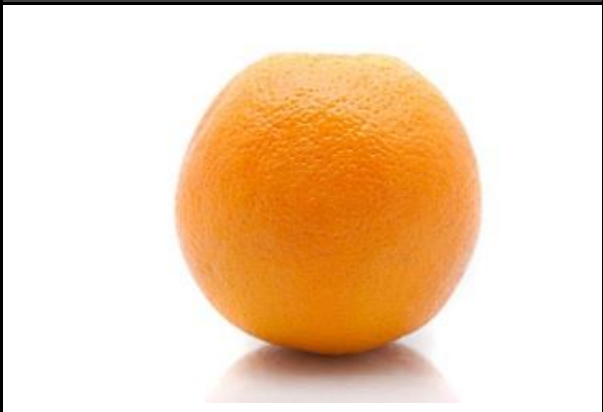

## Τελικά αποτελέσματα

Εχοντας υλοποιήσει τους επιμέρους αλγορίθμους πλέον μπορούμε να ξεκινήσουμε την αξιολόγηση του προγράμματος μας, δοκιμάζοντας να χρωματίσουμε μια ασπρόμαυρη εικόνα, βάση ενός συνόλου εικόνων εκπαίδευσης.

### Test Case 1



## Test Case 2

Training Images	
	
Image for colorization	
Grayscale	Colorized
	

Όπως βλέπουμε ο αλγόριθμος που υλοποιήσαμε έχει σχετικά υψηλά ποσοστά επιτυχίας. Ωστόσο δεν αποτελεί ένα ολοκληρωμένο σύστημα χρωματισμού ασπρόμαυρων εικόνων, αλλά μια απλή επίδειξη του ότι ο χρωματισμός εικόνων είναι εφικτός με την υλοποίηση των κατάλληλων διαδικασιών. Μερικά μέτρα που θα μπορούσαμε να λαβουμε για την βελτίωση των επιδόσεων θα ήταν η χρήση επιπρόσθετων χαρακτηριστικών όπως οι διακυμάνσεις της φωτεινότητας των superpixels καθώς και της μέσης φωτεινότητας. Επιπλέον η αύξηση του dataset των training εικόνων καθώς και ο αριθμός κύκλων εκπαίδευσης του ταξινομητή, τα οποία βέβαια αυξάνουν σε μεγάλο βαθμό τον χρόνο εκτέλεσης του προγράμματος και για αυτο τον λόγο παραλήφθηκαν από την εν λόγω εργασία.

Αξίζει να σημειωθεί ότι η παραμετροποίηση των αλγορίθμων έγινε μετά από πολυάριθμους ελέγχους για την διασφάλιση του βέλτιστου αποτελέσματος, καθώς και βάση των προτεινόμενων ρυθμίσεων απο τα Sources που αναγράφονται στην επόμενη ενότητα.

# Εκτέλεση

## Εκτέλεση κώδικα

### Training of classifier

```
gskaros@gskaros: ~/Desktop/Ανάλυση Εικόνας
(base) gskaros@gskaros:~/Desktop/Ανάλυση Εικόνας$ python3 ImageColorizer.py -t --training-set images/train/
# ===== #
# Color Palette Generation #
# ===== #

[+] Color Extraction:
[*] Extracting colors from image: images/train/orange (4).jpg
[*] Extracting colors from image: images/train/orange (5).jpg
[*] Extracting colors from image: images/train/orange (6).jpg
[*] Extracting colors from image: images/train/orange (3).jpg
[*] Extracting colors from image: images/train/orange (7).jpg
[*] Extracting colors from image: images/train/orange (8).jpg
[*] Extracting colors from image: images/train/orange (2).jpg
[*] Extracting colors from image: images/train/orange (1).jpg

[+] Saving the color palette at: objects/ColorPalette.object

# ===== #
# Feature extraction #
# ===== #

[+] Loading the pre-calculated color palette from: objects/ColorPalette.object

[+] SURF & Gabor feature extraction:
[Image: images/train/orange (4).jpg]:
[*] Converting to LAB color space
[*] Splitting the image into superpixels
[*] Superpixel processing:
Superpixel: 1/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 1/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 1/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 1/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 2/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 2/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 2/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 2/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 3/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 3/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 3/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 3/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 4/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 4/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 4/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 4/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 5/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 5/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 5/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 5/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 6/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 6/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
: Pending Superpixel: 6/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Pending Superpixel: 6/572 | Dominant Color Extraction: Done | SURF Feature Extraction: Done | Gabor Feature Extraction: Done
: Done Superpixel: 7/572 | Dominant Color Extraction: Pending | SURF Feature Extraction: Pending | Gabor Feature Extraction: Pending
```



```
gskaros@gskaros: ~/Desktop/Ανάλυση Εικόνας
Pending Superpixel: 580/580 Dominant Color Extraction: Done SURF Feature Extraction: Done Gabor Feature Extraction
Pending Superpixel: 580/580 Dominant Color Extraction: Done SURF Feature Extraction: Done Gabor Feature Extraction
Done Superpixel: 580/580 Dominant Color Extraction: Done SURF Feature Extraction: Done Gabor Feature Extraction
Done

[+] Saving the extracted features at: datasets/Features.dataset
# ===== #
# Feature Dimensionality Reduction #
# ===== #

[+] Loading the pre-calculated features from: datasets/Features.dataset
[+] Generating the PCA (Principal Component Analysis)
[+] Generating the standard scaler
[+] Performing the dimensionality reduction

[+] Saving the PCA at: objects/PCA.object
[+] Saving the standard scaler at: objects/StandardScaler.object
[+] Saving the reduced features at: datasets/ReducedFeatures.dataset

# ===== #
# OvR SVM training #
# ===== #

[+] Loading the pre-calculated reduced features from: datasets/ReducedFeatures.dataset
[+] Generating the OvR SVM classifier
[+] Generating a repeated K-Fold cross validator

[+] Training progress:
[Step: 1] OvR SVM Classifier Accuracy: 79.218 %
[Step: 2] OvR SVM Classifier Accuracy: 80.196 %
[Step: 3] OvR SVM Classifier Accuracy: 83.863 %
[Step: 4] OvR SVM Classifier Accuracy: 78.24 %
[Step: 5] OvR SVM Classifier Accuracy: 79.218 %
[Step: 6] OvR SVM Classifier Accuracy: 77.506 %
[Step: 7] OvR SVM Classifier Accuracy: 77.017 %
[Step: 8] OvR SVM Classifier Accuracy: 77.506 %
[Step: 9] OvR SVM Classifier Accuracy: 82.641 %
[Step: 10] OvR SVM Classifier Accuracy: 78.729 %
[Step: 11] OvR SVM Classifier Accuracy: 81.418 %
[Step: 12] OvR SVM Classifier Accuracy: 77.751 %
[Step: 13] OvR SVM Classifier Accuracy: 81.418 %
[Step: 14] OvR SVM Classifier Accuracy: 77.751 %
[Step: 15] OvR SVM Classifier Accuracy: 77.995 %
[Step: 16] OvR SVM Classifier Accuracy: 81.174 %
[Step: 17] OvR SVM Classifier Accuracy: 81.418 %
[Step: 18] OvR SVM Classifier Accuracy: 80.44 %
[Step: 19] OvR SVM Classifier Accuracy: 77.506 %
[Step: 20] OvR SVM Classifier Accuracy: 78.484 %

Max. OvR SVM Classifier Accuracy: 83.863 %
Min. OvR SVM Classifier Accuracy: 77.017 %
```

With and without Graph Cuts

```
gskaros@gskaros: ~/Desktop/Ανάλυση Εικόνας
(base) gskaros@gskaros:~/Desktop/Ανάλυση Εικόνας$ python3 ImageColorizer.py -c -i images/test/orange.jpg -o images/results/colorized_without_graphcuts.jpg
# ===== #
# Image Colorization #
# ===== #

[+] Loading the color palette ..... Resource: objects/ColorPalette.object
[+] Loading the PCA ..... Resource: objects/PCA.object
[+] Loading the standard scaler ... Resource: objects/StandardScaler.object
[+] Loading the OvR SVM ..... Resource: objects/OvR_SVM.object
[+] Loading the test image ..... Resource: images/test/orange.jpg

[+] Converting the test image from RGB to LAB color space
[+] Splitting the test image to superpixels
[+] Colorizing the test image:
    [*] Using graph cuts: NO
    [*] Colorizing superpixel: 600/600

[+] Converting the colorized image back to the RGB color space
[+] Display image? (y/n) y
[+] Saving the colorized image at: images/results/colorized_without_graphcuts.jpg
(base) gskaros@gskaros:~/Desktop/Ανάλυση Εικόνας$ python3 ImageColorizer.py -c -i images/test/orange.jpg -o --use-graphcuts images/results/colorized_with_graphcuts.jpg
usage: ImageColorizer.py [-h] [-t] [--training-set TRAINING SET] [-c]
                        [-i INPUT IMAGE] [-o OUTPUT IMAGE] [--use-graphcuts]
ImageColorizer.py: error: argument -o/--output image: expected one argument
(base) gskaros@gskaros:~/Desktop/Ανάλυση Εικόνας$ python3 ImageColorizer.py -c -i images/test/orange.jpg -o images/results/colorized_with_graphcuts.jpg --use-graphcuts
# ===== #
# Image Colorization #
# ===== #

[+] Loading the color palette ..... Resource: objects/ColorPalette.object
[+] Loading the PCA ..... Resource: objects/PCA.object
[+] Loading the standard scaler ... Resource: objects/StandardScaler.object
[+] Loading the OvR SVM ..... Resource: objects/OvR_SVM.object
[+] Loading the test image ..... Resource: images/test/orange.jpg

[+] Converting the test image from RGB to LAB color space
[+] Splitting the test image to superpixels
[+] Colorizing the test image:
    [*] Using graph cuts: YES
    [*] Generating the graph matrix from the SVM Margins: 100 %
    [*] Calculating the edges using the sobel filter
    [*] Calculating the pairwise costs between the color labels: 100 %
    [*] Calculating the final color labels using the graph cuts algorithm
    [*] Predicting the A and B color layers for the test image: 100 %
    [*] Merging L, A and B color layers together

[+] Converting the colorized image back to the RGB color space
[+] Display image? (y/n) y
[+] Saving the colorized image at: images/results/colorized_with_graphcuts.jpg
(base) gskaros@gskaros:~/Desktop/Ανάλυση Εικόνας$
```

## Βιβλιογραφία

Use	Source
Mini batch KMeans	<a href="https://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans">https://scikit-learn.org/stable/modules/clustering.html#mini-batch-kmeans</a>
SVM - C-Support Vector Classification	<a href="https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html">https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html</a>
rgb2lab library code analysis	<a href="https://github.com/scikit-image/scikit-image/blob/master/skimage/color/colorconv.py#L997">https://github.com/scikit-image/scikit-image/blob/master/skimage/color/colorconv.py#L997</a>
rgb2lab library	<a href="https://scikit-image.org/docs/dev/api/skimage.color.html#skimage.color.rgb2lab">https://scikit-image.org/docs/dev/api/skimage.color.html#skimage.color.rgb2lab</a>
Colorization of Grayscale Images with Machine Learning	<a href="http://www.vashishtm.net/pdf/color_report.pdf">http://www.vashishtm.net/pdf/color_report.pdf</a>
Gabor Filter	<a href="https://en.wikipedia.org/wiki/Gabor_filter">https://en.wikipedia.org/wiki/Gabor_filter</a>
Video: How SLIC (Simple Linear Iterative Clustering) algorithm works	<a href="https://www.youtube.com/watch?v=-hmUbB-Y8R0">https://www.youtube.com/watch?v=-hmUbB-Y8R0</a>
Gabor Features in Image Analysis	<a href="http://vision.cs.tut.fi/data/publications/ipta2012.pdf">http://vision.cs.tut.fi/data/publications/ipta2012.pdf</a>
Energy minimization with Graph Cuts	<a href="https://profs.etsmtl.ca/hlombaert/energy/">https://profs.etsmtl.ca/hlombaert/energy/</a>
Βιβλιο μαθήματος	-
Papers απο την εκφώνηση της εργασίας	-