Απαλλακτική Εργασία του μαθήματος «Ανάλυση Εικόνας» για το ακαδημαϊκό έτος 2020-2021

Θέμα:

Αυτόματος Χρωματισμός Ασπρόμαυρης Εικόνας με Χρήση Τεχνικών Μηχανικής Μάθησης.

Επιμελούμενοι φοιτητές:

Παναγιώτης Δημητρέλλος – Π17026

Χαράλαμπος Φωτογιάννης – Π17156

Εκφώνηση:

Να υλοποιήσετε τις απαραίτητες αλγοριθμικές διαδικασίες για τον αυτόματο χρωματισμό μιας ασπρόμαυρης εικόνας. Συγκεκριμένα, θα πρέπει να αναπτύξετε κώδικα σε περιβάλλον Matlab ή Python για την διενέργεια των παρακάτω υπολογιστικών δραστηριοτήτων:

i. Αναπαράσταση Εικόνας στον Χρωματικό Χώρο Lab .
ii. Διακριτοποίηση του Χρωματικού Χώρου Lab με βάση ένα σύνολο συναφών εικόνων εκπαίδευσης.
iii. Κατάτμηση Εικόνας σε Superpixels σύμφωνα με τον αλγόριθμο SLIC.
iv. Εξαγωγή Χαρακτηριστικών Υφής (SURF Features & Gabor Features) ανά Super Pixel.
v. Εκμάθηση Τοπικών Μοντέλων Πρόγνωσης Χρώματος με Χρήση Ταξινομητών SVM .
vi. Εκτίμηση Χρωματικού Περιεχομένου Ασπρόμαυρης Εικόνας με Χρήση Αλγορίθμων Κοπής Γραφημάτων.

Επίλυση Θέματος:

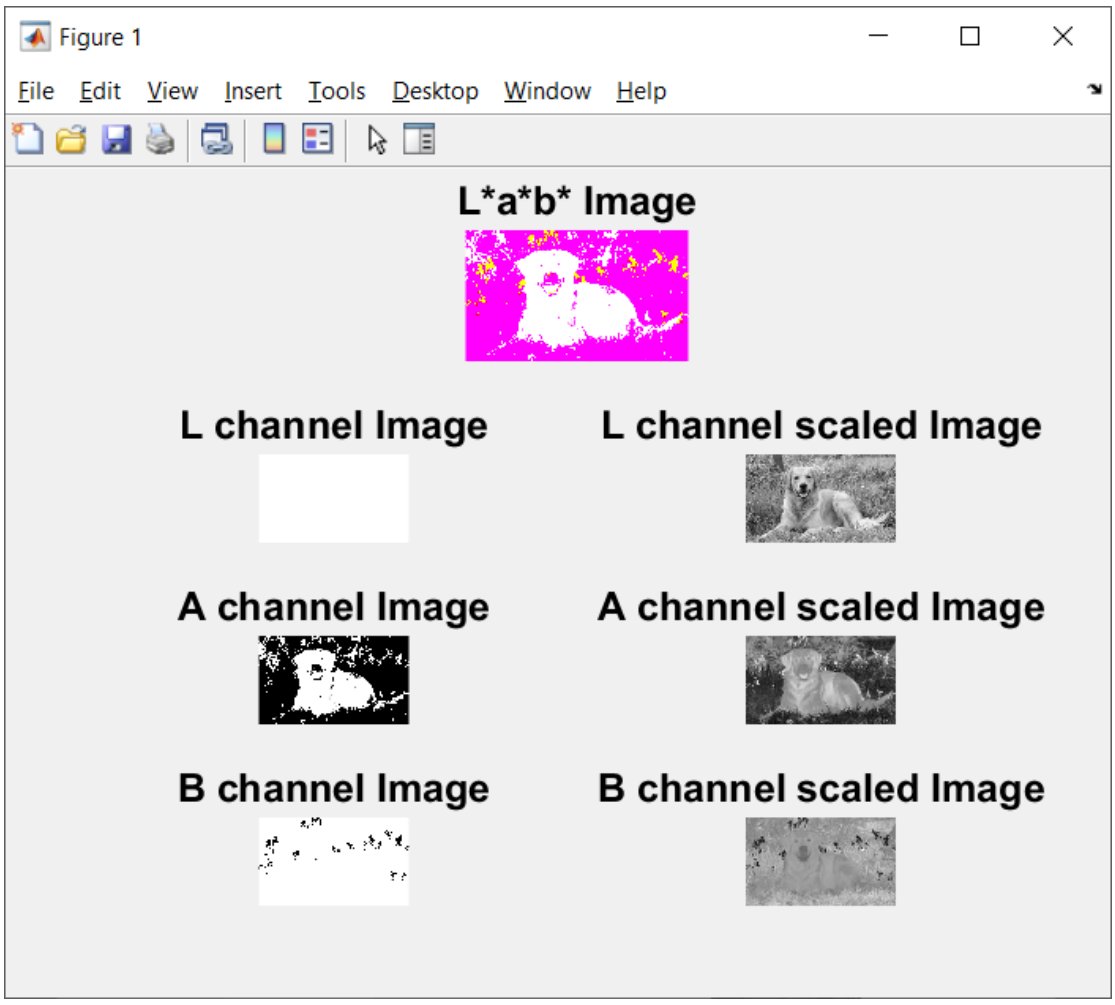
1) Αναπαράσταση Εικόνας στον Χρωματικό Χώρο Lab .

> Τεκμηρίωση πηγαίου κώδικα αρχείου «Color_Space_Lab_Representation.m»:

```
% Αποκτάμε την εικόνα
source = imread("dog.jpg");

% Μετατρέπουμε την πηγαία RGB εικόνα σε εικόνα L*a*b*
% χρησιμοποιώντας rgb2lab
labImage = rgb2lab(source);

% Παίρνουμε το κάθε κανάλι 'L*', 'a*' και 'b*' για την
L*a*b* εικόνα
LImage = labImage(:, :, 1);
AImage = labImage(:, :, 2);
BImage = labImage(:, :, 3);

% Εμφανίζουμε την εικόνα L*a*b*
subplot(4, 2, 1.5);
imshow(labImage);
title('L*a*b* Image', 'FontSize', 15);


% Εμφανίζουμε καθένα από τα κανάλια ατομικά
% και ρυθμίζοντας την απεικόνιση βασισμένη στο εύρος των
% τιμών των pixels
subplot(4, 2, 3);
imshow(LImage);
title('L channel Image', 'FontSize', 15);
```

```
subplot(4, 2, 4);
imshow(LImage, []);
title('L channel scaled Image', 'FontSize', 15);
subplot(4, 2, 5);
imshow(AImage);
title('A channel Image', 'FontSize', 15);
subplot(4, 2, 6);
imshow(AImage, []);
title('A channel scaled Image', 'FontSize', 15);
subplot(4, 2, 7);
imshow(BImage);
title('B channel Image', 'FontSize', 15);
subplot(4, 2, 8);
imshow(BImage, []);
title('B channel scaled Image', 'FontSize', 15);
```

> Παράδειγμα Εκτέλεσης:



2) Διακριτοποίηση του Χρωματικού Χώρου Lab με βάση ένα σύνολο συναφών εικόνων εκπαίδευσης.

Πηγή:

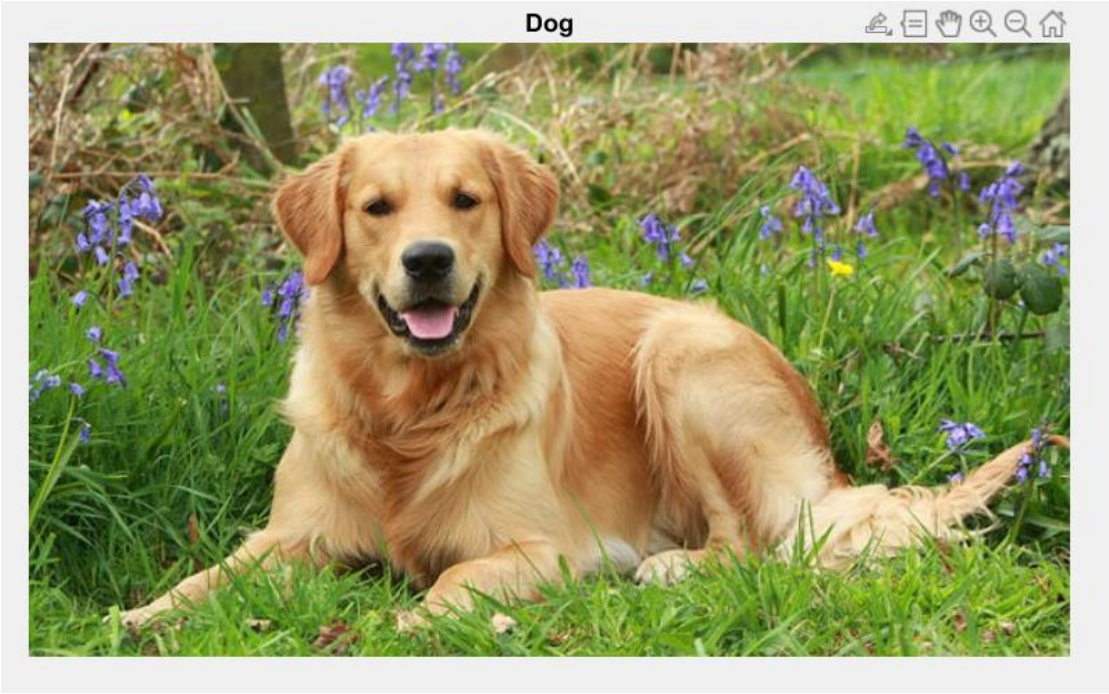
https://www.mathworks.com/help/images/color-based-segmentation-using-the-l-a-b-color-space.html#LabColorSegmentationExample-1

Σε αυτό το αρχείο δοκιμάσαμε να κάνουμε διακριτοποίηση του Χρωματικού Χώρου Lab με βάση μιας μόνο συναφής εικόνας εκπαίδευσης ως ένα αρχείο για testing της μεθοδολογίας.

> Τεκμηρίωση πηγαίου κώδικα αρχείου «Color_Based_Segmentation_single_test.m»:

Βήμα 1ο : Αποκτάμε την εικόνα.

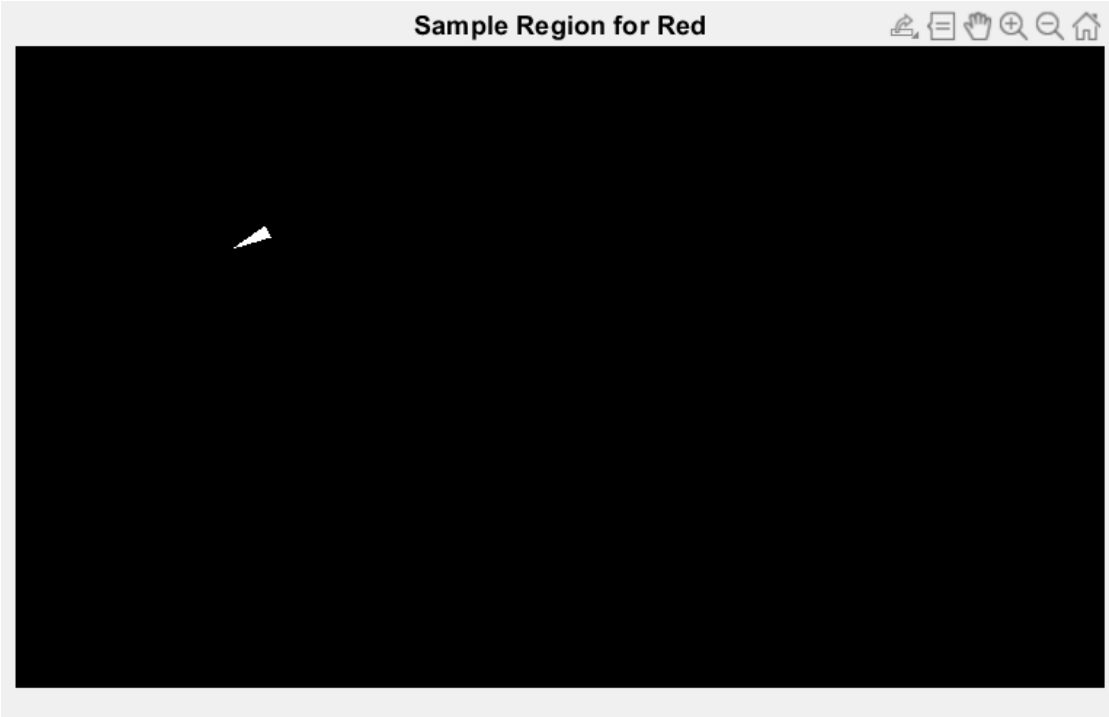
```
%Αποκτάμε την εικόνα.
source = imread("dog.jpg");
imshow(source);
title("Dog")
```



<u>Βήμα 2ο</u> : Υπολογίζουμε τα δείγματα χρώματος στον χρωματικό χώρο L*a*b για κάθε περιοχή.

```
%{
Μπορούμε να δούμε έξι βασικά χρώματα στην εικόνα: το
χρώμα φόντου, κόκκινο,
πράσινο, μοβ, κίτρινο και ματζέντα.

Ο χρωματικός χώρος L * a * b * (επίσης γνωστός ως CIELAB
ή CIE L * a * b *)
σας επιτρέπει να ποσοτικοποιήσετε αυτές τις οπτικές
διαφορές.Ο χώρος χρωμάτων L * a * b *
προέρχεται από τις τιμές τριχίσματος CIE XYZ. Ο χώρος L *
a * b * αποτελείται
από μια φωτεινότητα «L *» ή στρώμα φωτεινότητας, στρώμα
χρωματικότητας «a *»
που υποδεικνύει πού πέφτει το χρώμα κατά μήκος του
κόκκινου-πράσινου άξονα και στρώμα
χρωματικότητας «b *» υποδεικνύοντας πού πέφτει το χρώμα
κατά μήκος του μπλε-κίτρινου άξονα.

Η προσέγγισή μας είναι να επιλέξουμε μια μικρή περιοχή
δείγματος για κάθε
χρώμα και να υπολογίσετε το μέσο χρώμα κάθε περιοχής
δείγματος στο χώρο
'a * b *'.Θα χρησιμοποιήσουμε αυτούς τους χρωματικούς
δείκτες για να ταξινομήσετε
κάθε pixel.
%}
load regioncoordinates;

nColors = 6;
sample_regions = false([size(source,1) size(source,2)
nColors]);

for count = 1:nColors
  sample_regions(:,:,count) =
roipoly(source,region_coordinates(:,1,count), ...

region_coordinates(:,2,count));
end

imshow(sample_regions(:,:,2))
title('Sample Region for Red')


% Μετατρέπουμε την εικόνα RGB προέλευσης σε εικόνα L * a
* b * χρησιμοποιώντας rgb2lab
labImage = rgb2lab(source);
```

```
% Υπολογίζουμε τη μέση τιμή 'a *' και 'b *' για κάθε
περιοχή που εξάγουμε με roipoly.
% Αυτές οι τιμές χρησιμεύουν ως οι χρωματικοί δείκτες μας
στο διάστημα 'a * b *
AImage = labImage(:, :, 2);
BImage = labImage(:, :, 3);

color_markers = zeros([nColors, 2]);

for count = 1:nColors
  color_markers(count,1) =
mean2(AImage(sample_regions(:,:,count)));
  color_markers(count,2) =
mean2(BImage(sample_regions(:,:,count)));
end

% Παράδειγμα το μέσο χρώμα της κόκκινης περιοχής
δείγματος στο διάστημα 'a * b *' είναι:
fprintf('[%0.3f,%0.3f]\n',color_markers(2,1),color_marker
s(2,2));
```



<u>Βήμα 3ο :</u> Ταξινόμηση κάθε εικονοστοιχείου χρησιμοποιώντας τον κανόνα του πλησιέστερου γείτονα

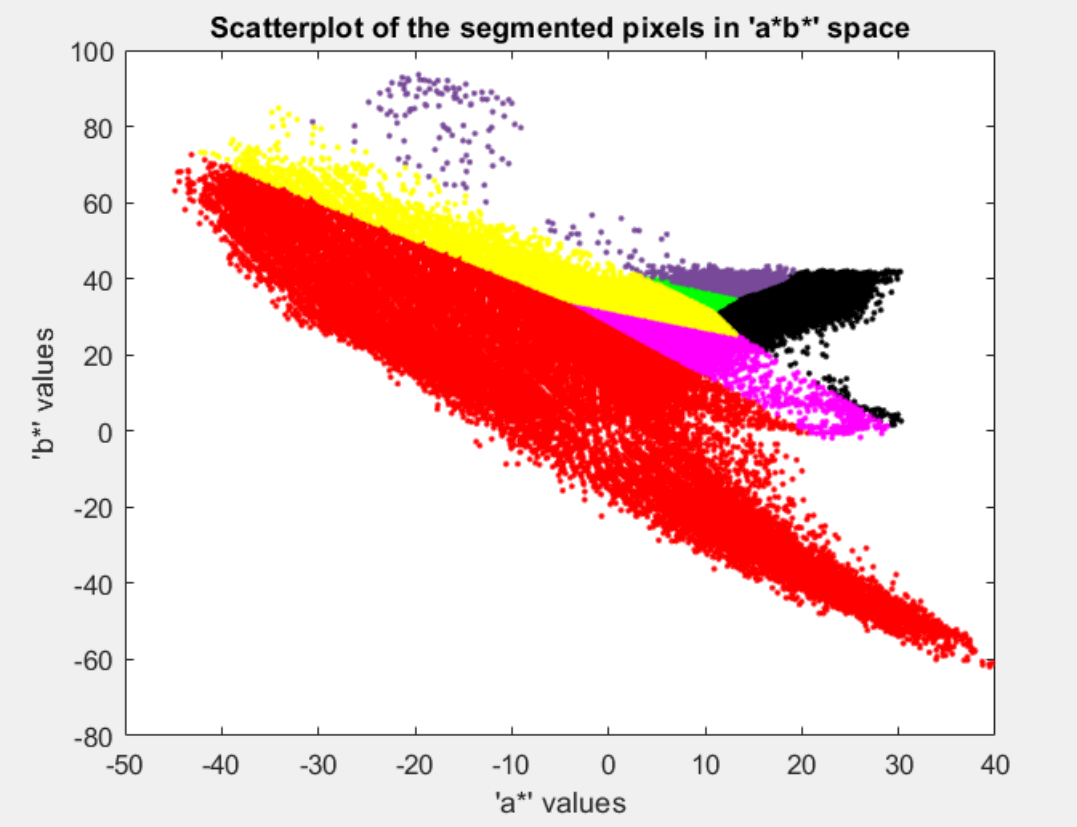
```
%{
Κάθε δείκτης χρώματος έχει τώρα τιμή «a *» και «b *».
Μπορούμε να ταξινομήσουμε κάθε εικονοστοιχείο στην εικόνα
lab_fabric υπολογίζοντας την Ευκλείδεια απόσταση
μεταξύ αυτού του pixel και κάθε χρώματος. Η μικρότερη
απόσταση θα μας πει ότι το pixel ταιριάζει περισσότερο με
αυτόν τον χρωματικό δείκτη.
Για παράδειγμα, εάν η απόσταση μεταξύ ενός
εικονοστοιχείου και του δείκτη κόκκινου χρώματος είναι η
μικρότερη, τότε το εικονοστοιχείο θα επισημαίνεται ως
κόκκινο εικονοστοιχείο.
%}

%Δημιουργούμε έναν πίνακα που περιέχει τις ετικέτες
χρωμάτων μας, δηλαδή, 0 = φόντο, 1 = κόκκινο, 2 =
πράσινο, 3 = μοβ, 4 = ματζέντα και 5 = κίτρινο.
color_labels = 0:nColors-1;

% Αρχικοποιούμε πίνακες που θα χρησιμοποιηθούν στην
ταξινόμηση του πλησιέστερου γείτονα AImage =
double(AImage);
BImage = double(BImage);
distance = zeros([size(AImage), nColors]);

% Εκτελούμε ταξινόμηση
for count = 1:nColors
  distance(:,:,count) = ( (AImage -
color_markers(count,1)).^2 + ...
                      (BImage -
color_markers(count,2)).^2 ).^0.5;
end

[~,label] = min(distance,[],3);
label = color_labels(label);
clear distance;
```

Βήμα 4ο : Αποτελέσματα εμφάνισης της ταξινόμησης πλησιέστερου γείτονα

```
%{
Ο πίνακας ετικετών περιέχει μια έγχρωμη ετικέτα για κάθε
pixel στην εικόνα προέλευσης Χρησιμοποιούμε τη μήτρα
ετικέτας για να διαχωρίσουμε αντικείμενα στο πρωτότυπο
εικόνα πηγής ανά χρώμα.
%}
rgb_label = repmat(label,[1 1 3]);
segmented_images = zeros([size(source),
nColors],'uint8');

for count = 1:nColors
  color = source;
  color(rgb_label ~= color_labels(count)) = 0;
  segmented_images(:,:,:,count) = color;
end

% Εμφανίζουμε τα πέντε τμηματοποιημένα χρώματα ως μοντάζ.
Εμφανίζουμε επίσης τα εικονοστοιχεία φόντου στην εικόνα
που δεν είναι ταξινομείται ως χρώμα.
montage({segmented_images(:,:,:,2),segmented_images(:,:,:
,3) ...
segmented_images(:,:,:,4),segmented_images(:,:,:,5) ...
segmented_images(:,:,:,6),segmented_images(:,:,:,1)});
title("Montage of Red, Green, Purple, Magenta, and Yellow
Objects, and Background")
```

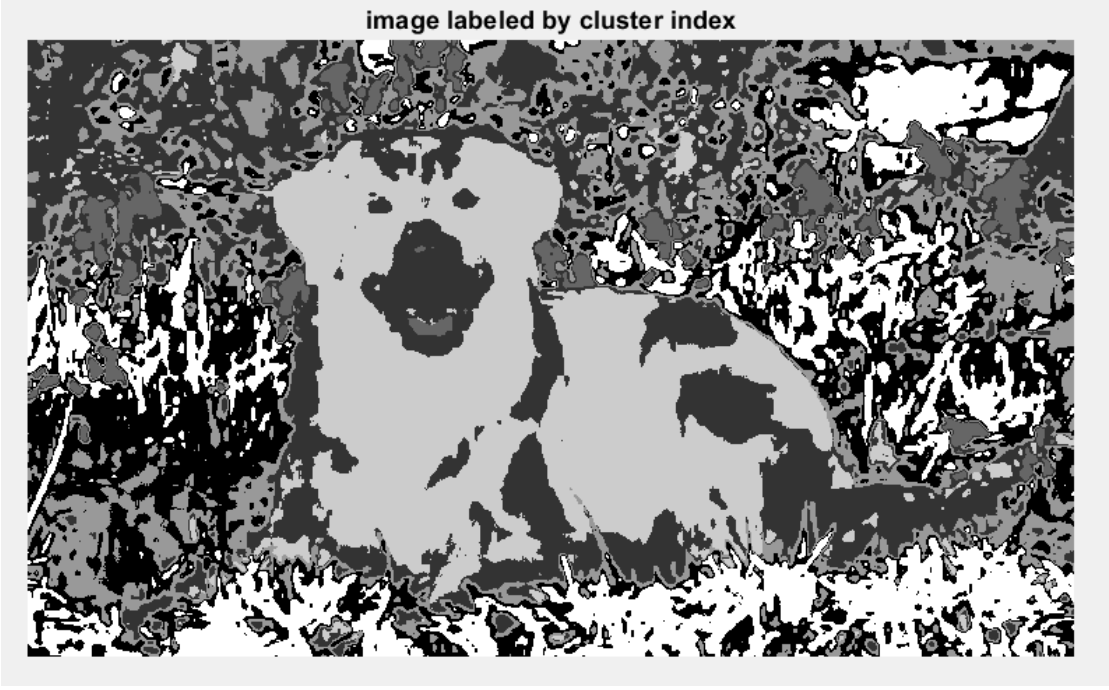
<u>Βήμα 5ο</u> : Εμφάνιση τιμών 'a *' και 'b *' των ετικετών χρωμάτων

```
%{
Μπορούμε να δούμε πόσο καλά η ταξινόμηση του πλησιέστερου
γείτονα διαχώρισε τους διαφορετικούς πληθυσμούς χρωμάτων
σχεδιάζοντας τις τιμές «a *» και «b *»
pixel που ταξινομήθηκαν σε ξεχωριστά χρώματα. Για σκοπούς
εμφάνισης, επισημάνουμε κάθε σημείο με την έγχρωμη
ετικέτα του.
%}
purple = [119/255 73/255 152/255];
plot_labels = {'k', 'r', 'g', purple, 'm', 'y'};

figure
for count = 1:nColors
  plot(AImage(label==count-1),BImage(label==count-
1),'.','MarkerEdgeColor', ...
       plot_labels{count}, 'MarkerFaceColor',
plot_labels{count});
  hold on;
end

title('Scatterplot of the segmented pixels in ''a*b*''
space');
xlabel('''a*'' values');
ylabel('''b*'' values');
```



Στην συνέχεια:

```
% Διαβάζουμε την εικόνα και την μετατρέπουμε σε χώρο
χρωμάτων L * a * b *
I = imread(dog.jpg');
Ilab = rgb2lab(I);

% Εξαγωγή καναλιών a * και b * και αναδιαμόρφωση
ab = double(Ilab(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);

% Τμηματοποίηση χρησιμοποιόντας k-means
nColors = 6;
[cluster_idx, cluster_center] = kmeans(ab,nColors,...
  'distance',     'sqEuclidean', ...
  'Replicates',   3);

% Εμφάνιση του αποτελέσματος
pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster
index')
```



Στο αρχείο «Color_Based_Segmentation_multi_.m» χρησιμοποιούμε την ίδια λογική με τον παραπάνω κώδικα μόνο με μια διαφορά.

Κάνουμε διακριτοποίηση του Χρωματικού Χώρου Lab με βάση ένα σύνολο συναφών εικόνων εκπαίδευσης δηλαδή όχι μόνο με μία εικόνα αλλά με τρεις.

> Τεκμηρίωση κώδικα:

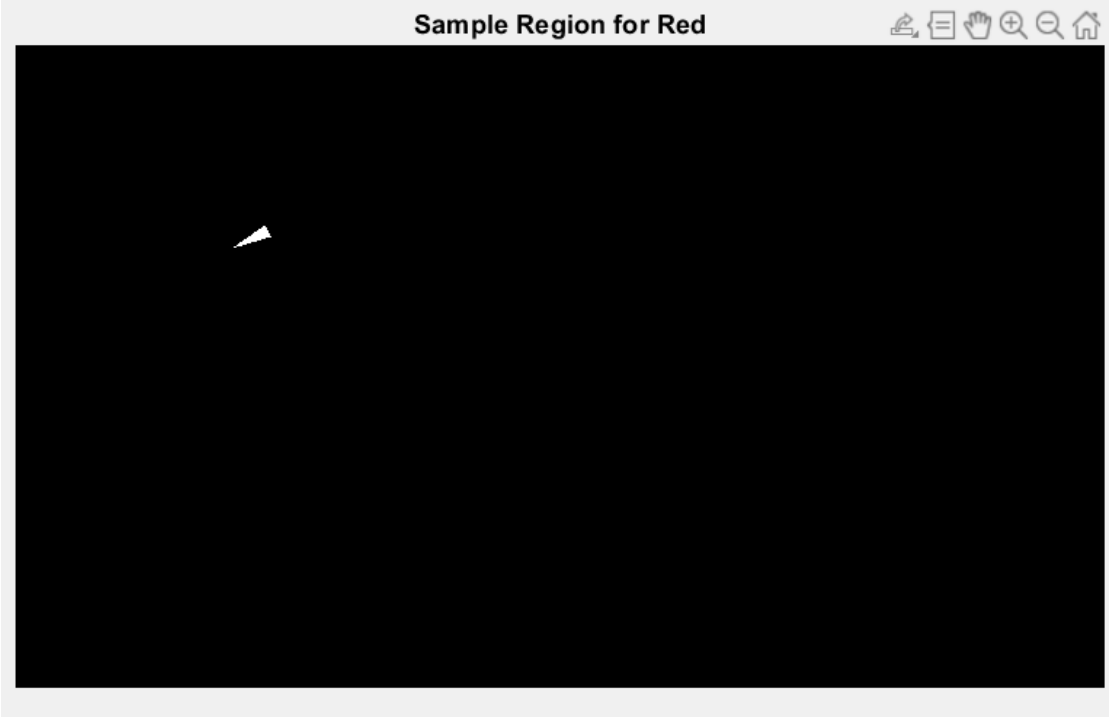
Βήμα 1ο : Αποκτάμε τις εικόνες.

```
% Αποκτάμε τις συναφείς εικόνες. Όλες απεικονίζουν
από έναν σκύλο σε πράσινο τοπίο
source1 = imread("dog1.jpg");
source2 = imread("dog2.jpg");
source3 = imread("dog3.jpg");
subplot(1,3,1), imshow(source1);
title("Dog 1");
subplot(1,3,2), imshow(source2);
title("Dog 2");
subplot(1,3,3), imshow(source3);
title("Dog 3");
```

<u>Βήμα 2ο</u> : Υπολογίζουμε τα δείγματα χρώματος στον χρωματικό χώρο L*a*b για κάθε περιοχή.

```
%{

Μπορούμε να δούμε έξι βασικά χρώματα στην εικόνα: το
χρώμα φόντου, κόκκινο,
πράσινο, μοβ, κίτρινο και ματζέντα.

Ο χρωματικός χώρος L * a * b * (επίσης γνωστός ως CIELAB
ή CIE L * a * b *) σας επιτρέπει να ποσοτικοποιήσετε
αυτές τις οπτικές διαφορές.Ο χώρος χρωμάτων L * a * b *
προέρχεται από τις τιμές τριχίσματος CIE XYZ. Ο χώρος L *
a * b * αποτελείται
από μια φωτεινότητα «L *» ή στρώμα φωτεινότητας, στρώμα
χρωματικότητας «a *»
που υποδεικνύει πού πέφτει το χρώμα κατά μήκος του
κόκκινου-πράσινου άξονα και στρώμα χρωματικότητας «b *»
υποδεικνύοντας πού πέφτει το χρώμα κατά μήκος του μπλε-
κίτρινου άξονα.

Η προσέγγισή μας είναι να επιλέξουμε μια μικρή περιοχή
δείγματος για κάθε
χρώμα και να υπολογίσουμε τον μέσο χρώμα κάθε περιοχής
δείγματος στο χώρο 'a * b *'.Θα χρησιμοποιήσουμε αυτούς
τους χρωματικούς δείκτες για να ταξινομήσουμε
κάθε pixel.
%}

load regioncoordinates;

nColors = 6;
sample_regions1 = false([size(source1,1) size(source1,2)
nColors]);
sample_regions2 = false([size(source2,1) size(source2,2)
nColors]);
sample_regions3 = false([size(source3,1) size(source3,2)
nColors]);

for count = 1:nColors
  sample_regions(:,:,count) =
roipoly(source,region_coordinates(:,1,count), ...

region_coordinates(:,2,count));
end

imshow(sample_regions(:,:,2))
title('Sample Region for Red')
```

```
% Μετατρέπουμε την εικόνα RGB προέλευσης σε εικόνα L * a
* b * χρησιμοποιώντας rgb2lab
labImage = rgb2lab(source);

% Υπολογίζουμε τη μέση τιμή 'a *' και 'b *' για κάθε
περιοχή που εξάγουμε με roipoly.
% Αυτές οι τιμές χρησιμεύουν ως οι χρωματικοί δείκτες μας
στο διάστημα 'a * b *
AImage = labImage(:, :, 2);
BImage = labImage(:, :, 3);

color_markers = zeros([nColors, 2]);

for count = 1:nColors
  color_markers(count,1) =
mean2(AImage(sample_regions(:,:,count)));
  color_markers(count,2) =
mean2(BImage(sample_regions(:,:,count)));
end

% Παράδειγμα το μέσο χρώμα της κόκκινης περιοχής
δείγματος στο διάστημα 'a * b *' είναι:
fprintf('[%0.3f,%0.3f]
\n',color_markers(2,1),color_markers(2,2));
```



<u>Βήμα 3ο</u> : Ταξινόμηση κάθε εικονοστοιχείου χρησιμοποιώντας τον κανόνα του πλησιέστερου γείτονα.

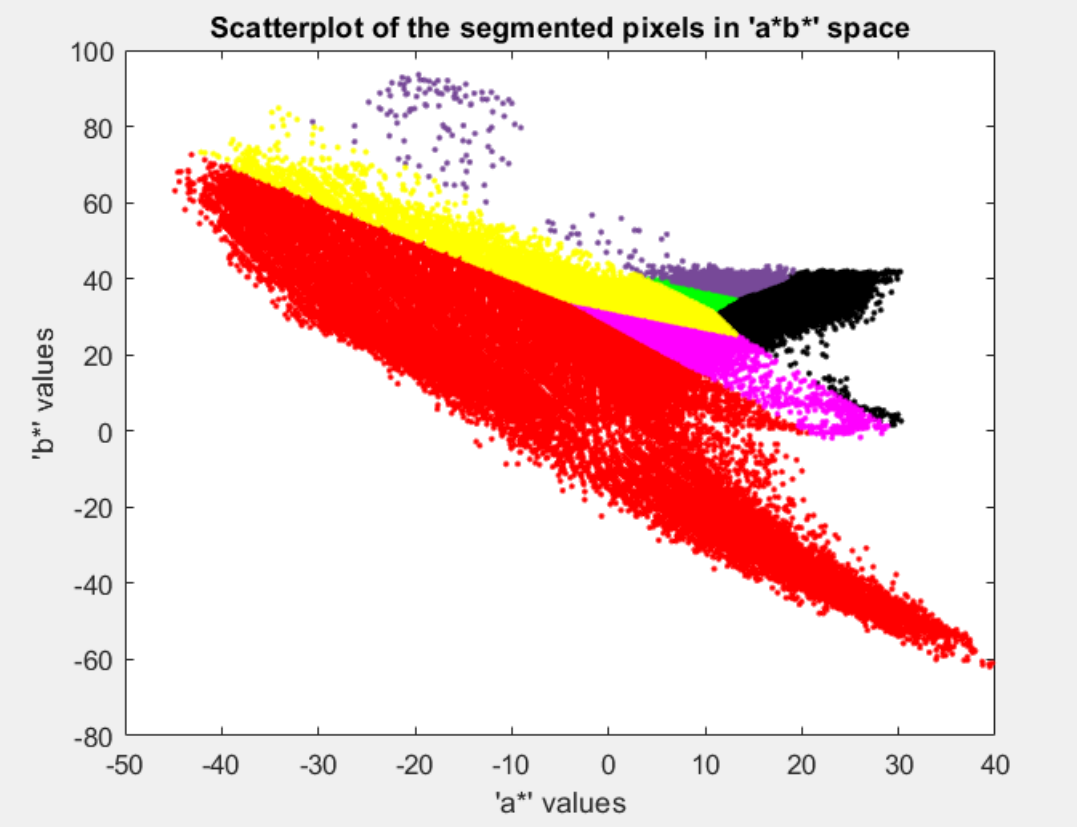
```
%{
Κάθε δείκτης χρώματος έχει τώρα τιμή «a *» και «b *».
Μπορούμε να ταξινομήσουμε κάθε εικονοστοιχείο στην εικόνα
lab_fabric υπολογίζοντας την Ευκλείδεια απόσταση
μεταξύ αυτού του pixel και κάθε χρώματος. Η μικρότερη
απόσταση θα μας πει ότι το pixel ταιριάζει περισσότερο με
αυτόν τον χρωματικό δείκτη.
Για παράδειγμα, εάν η απόσταση μεταξύ ενός
εικονοστοιχείου και του δείκτη κόκκινου χρώματος είναι η
μικρότερη, τότε το εικονοστοιχείο θα επισημαίνεται ως
κόκκινο εικονοστοιχείο.
%}

% Δημιουργούμε έναν πίνακα που περιέχει τις ετικέτες
χρωμάτων μας, δηλαδή, 0 = φόντο, 1 = κόκκινο, 2 =
πράσινο, 3 = μοβ, 4 = ματζέντα και 5 = κίτρινο.
color_labels = 0:nColors-1;

% Αρχικοποιούμε πίνακες που θα χρησιμοποιηθούν στην
ταξινόμηση του πλησιέστερου γείτονα AImage =
double(AImage);
AImage = double(AImage);
BImage = double(BImage);
distance = zeros([size(AImage), nColors]);

% Εκτελούμε ταξινόμηση
for count = 1:nColors
  distance(:,:,count) = ( (AImage -
color_markers(count,1)).^2 + ...
                      (BImage -
color_markers(count,2)).^2 ).^0.5;
end

[~,label] = min(distance,[],3);
label = color_labels(label);
clear distance;
```

<u>Βήμα 4ο</u> : Αποτελέσματα εμφάνισης της ταξινόμησης πλησιέστερου γείτονα.

```
%{
Ο πίνακας ετικετών περιέχει μια έγχρωμη ετικέτα για κάθε
pixel στην εικόνα προέλευσης Χρησιμοποιούμε τη μήτρα
ετικέτας για να διαχωρίσουμε αντικείμενα στο πρωτότυπο
εικόνα πηγής ανά χρώμα.
%}
rgb_label = repmat(label,[1 1 3]);
segmented_images = zeros([size(source),
nColors],'uint8');

for count = 1:nColors
  color = source;
  color(rgb_label ~= color_labels(count)) = 0;
  segmented_images(:,:,:,count) = color;
end

% Εμφανίζουμε τα πέντε τμηματοποιημένα χρώματα ως μοντάζ.
Εμφανίζουμε επίσης
% τα εικονοστοιχεία φόντου στην εικόνα που δεν είναι
ταξινομείται ως χρώμα.
montage({segmented_images(:,:,:,2),segmented_images(:,:,:
,3) ...
segmented_images(:,:,:,4),segmented_images(:,:,:,5) ...
segmented_images(:,:,:,6),segmented_images(:,:,:,1)});
title("Montage of Red, Green, Purple, Magenta, and Yellow
Objects, and Background")
```

Βήμα 5ο : Εμφάνιση τιμών 'a *' και 'b *' των ετικετών χρωμάτων.

```
%{
Μπορούμε να δούμε πόσο καλά η ταξινόμηση του πλησιέστερου
γείτονα διαχώρισε τους διαφορετικούς πληθυσμούς χρωμάτων
σχεδιάζοντας τις τιμές «a *» και «b *»
pixel που ταξινομήθηκαν σε ξεχωριστά χρώματα. Για σκοπούς
εμφάνισης, επισημάνουμε κάθε σημείο με την έγχρωμη
ετικέτα του.
%}
purple = [119/255 73/255 152/255];
plot_labels = {'k', 'r', 'g', purple, 'm', 'y'};

figure
for count = 1:nColors
  plot(AImage(label==count-1),BImage(label==count-
1),'.','MarkerEdgeColor', ...
       plot_labels{count}, 'MarkerFaceColor',
plot_labels{count});
  hold on;
end

title('Scatterplot of the segmented pixels in ''a*b*''
space');
xlabel('''a*'' values');
ylabel('''b*'' values');
```



Στην συνέχεια:

```
% Διαβάζουμε την εικόνα και την μετατρέπουμε σε χώρο
χρωμάτων L * a * b *
I = imread(dog.jpg');
Ilab = rgb2lab(I);
% Εξαγωγή καναλιών a * και b * και αναδιαμόρφωση
ab = double(Ilab(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
% Τμηματοποίηση χρησιμοποιόντας k-means
nColors = 6;
[cluster_idx, cluster_center] = kmeans(ab,nColors,...
  'distance',     'sqEuclidean', ...
  'Replicates',   3);
% Εμφάνιση του αποτελέσματος
pixel_labels = reshape(cluster_idx,nrows,ncols);
imshow(pixel_labels,[]), title('image labeled by cluster
index')
```

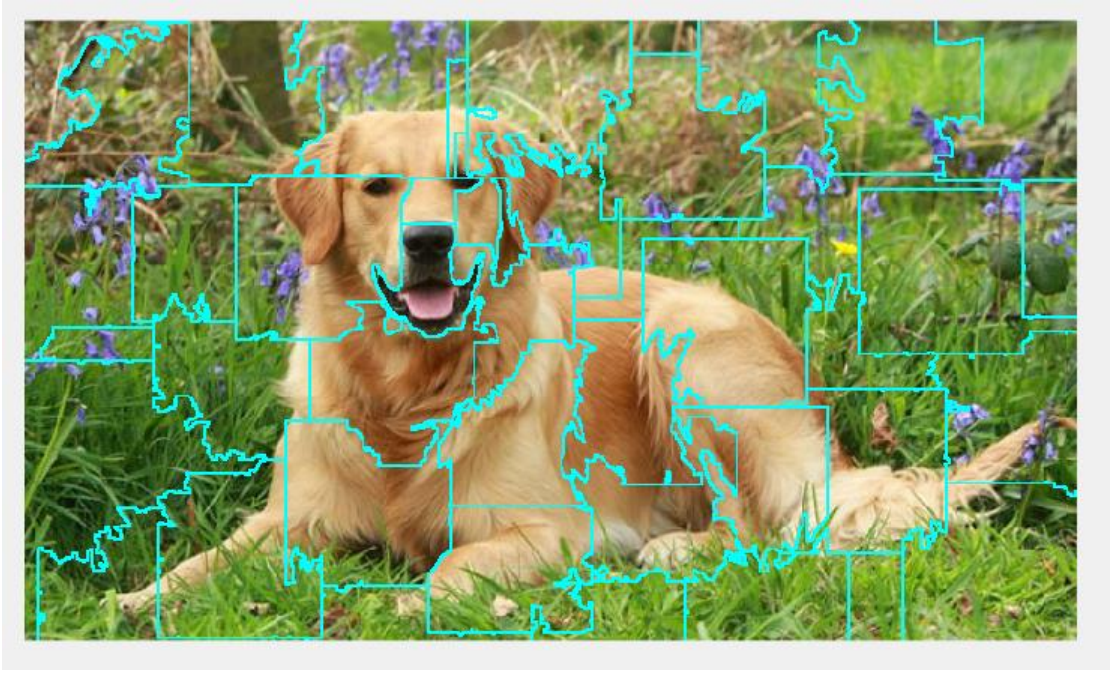
3) Κατάτμηση Εικόνας σε Superpixels σύμφωνα με τον αλγόριθμο SLIC

Πηγή: https://www.mathworks.com/help/images/ref/superpixels.html

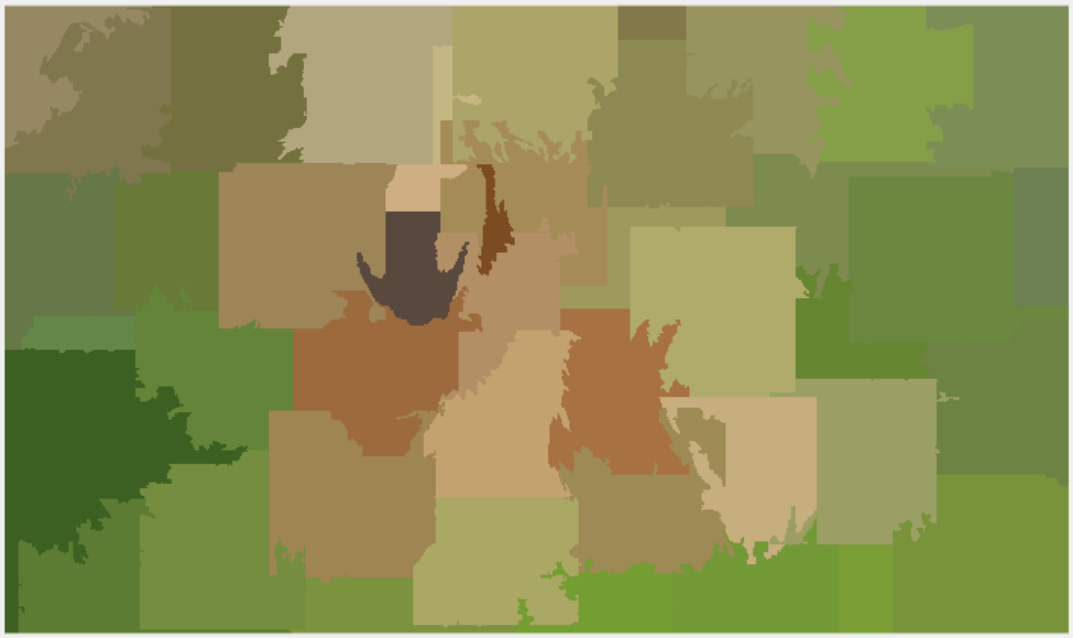
Ο κώικδα για την δραστηριότητα αυτή βρίσκεται στο αρχείο «SLIC_Algorithm_Superpixel_Segmentation.m» .

> Τεκμηρίωση κώδικα:

```
% Μετατρέπουμε την εικόνα RGB προέλευσης σε εικόνα L * a
* b * χρησιμοποιώντας rgb2lab
labImage = rgb2lab(source);

% Υπολογίζουμε τα superpixels της εικόνας.
[L,N] = superpixels(labImage,100,'Method','slic');

% Εμφανίζουμε τα όρια του superpixel που επικαλύπτονται
στην αρχική εικόνα.
figure
bw = boundarymask(B);
imshow(imoverlay(source,bw,'cyan'),'InitialMagnification'
,67);
```



```
% Ορίζουμε το χρώμα κάθε εικονοστοιχείου στην εικόνα
εξόδου στο μέσο χρώμα RGB της περιοχής του superpixel.
outputImage = zeros(size(source),'like',source);
idx = label2idx(L);
numRows = size(source,1);
numCols = size(source,2);
for labelVal = 1:N
    redIdx = idx{labelVal};
    greenIdx = idx{labelVal}+numRows*numCols;
    blueIdx = idx{labelVal}+2*numRows*numCols;
    outputImage(redIdx) = mean(source(redIdx));
    outputImage(greenIdx) = mean(source(greenIdx));
    outputImage(blueIdx) = mean(source(blueIdx));
end

figure
imshow(outputImage,'InitialMagnification',67)
```



4) Εξαγωγή Χαρακτηριστικών Υφής (SURF Features & Gabor Features) ανά Super Pixel.

1. Εξαγωγή Χαρακτηριστικών Υφής (SURF Features) ανά Super Pixel.

1.1 Ανάλυση του αρχείου Superpixel_Texture_Extraction_SURF_Features_.m
1.2 Ανάλυση του αρχείου SURF-Features_Texture_Extraction.m

Πηγές:

https://www.mathworks.com/help/vision/feature-detection-and-extraction.html

https://www.mathworks.com/help/vision/ref/detectsurffeatures.html

https://www.mathworks.com/help/vision/ref/extractfeatures.html

https://www.mathworks.com/help/vision/ref/matchfeatures.html

https://www.mathworks.com/help/vision/ref/showmatchedfeatures.html

https://www.mathworks.com/help/images/ref/imshow.html

https://www.mathworks.com/help/gpucoder/ug/feature-extraction-using-surf.html?s_tid=srchtitle

1.1)

Σε αυτό το αρχείο «Superpixel_Texture_Extraction_SURF_Features_.m» κάναμε εξαγωγή Χαρακτηριστικών Υφής (SURF Features) ανά Super Pixel.

> Τεκμηρίωση κώδικα:

```
% Αποκτάμαι την εικόνα
source = imread("dog.jpg");

%source = rgb2lab(source);

[L,N] = superpixels(source,100);

BW = boundarymask(L);

outputImage = zeros(size(source), 'like', source);
```

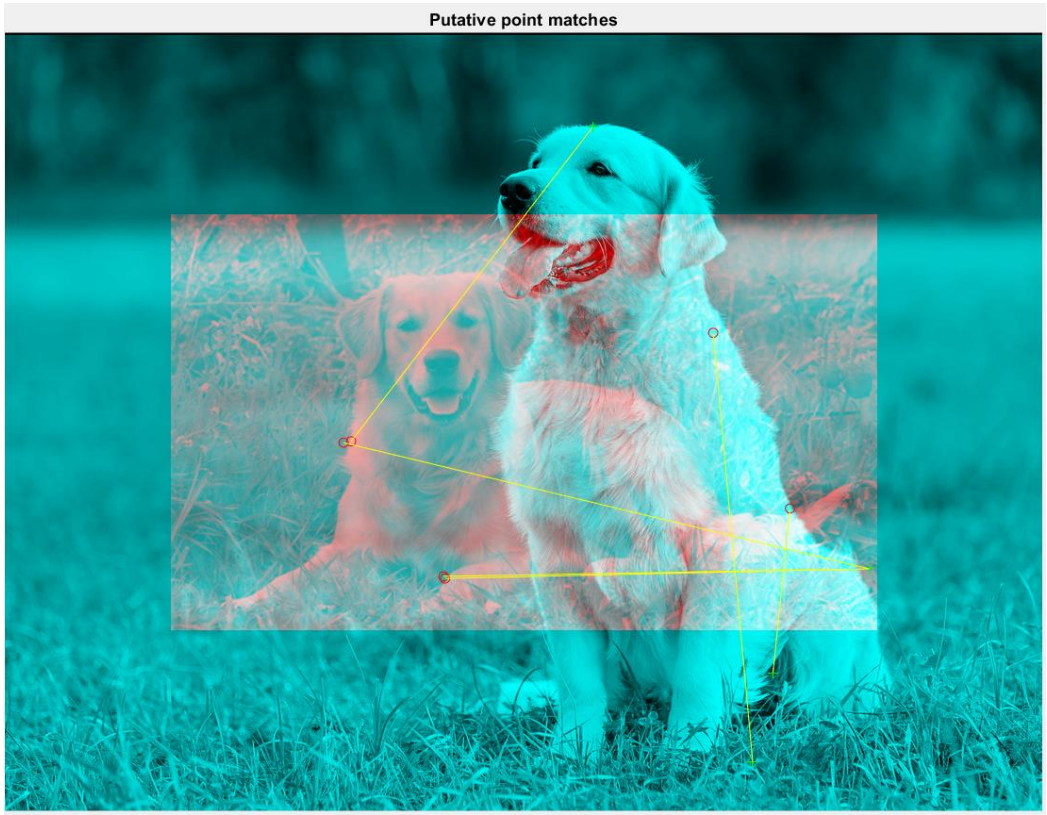
Πρώτο βήμα:

```
%{ Θα χρησιμοποιήσω τη συνάρτηση label2idx για τον
υπολογισμό των δεικτών των pixel σε κάθε σύμπλεγμα
superpixel.Αυτό θα μου επιτρέψει να αποκτήσω πρόσβαση
```

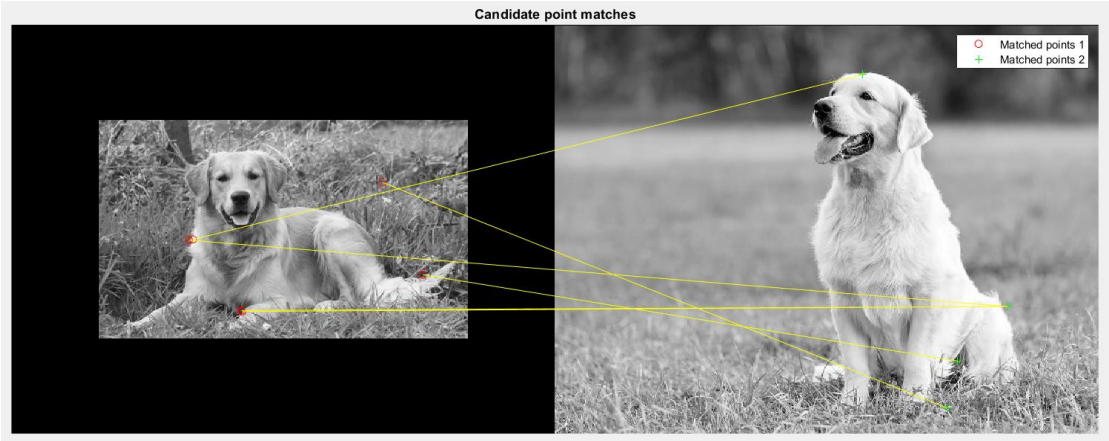
```
στις τιμές των κόκκινων,πράσινων και μπλε στοιχείων
χρησιμοποιώντας γραμμική ευρετηρίαση %}

idx = label2idx(L);
numRows = size(source,1);
numCols = size(source,2);

%{
Για κάθε ένα από τα συμπλέγματα N superpixel,
χρησιμοποιούμε γραμμική
ευρετηρίαση για πρόσβαση στα κόκκινα, πράσινα και μπλε
στοιχεία,
ανακατασκευάζουμε τα αντίστοιχα εικονοστοιχεία ενώ
ανιχνεύουμε τα
χαρακτηριστικά SURF για την έκδοση κλίμακας του γκρι της
εικόνας
και δείχνει τα 10 ισχυρότερα από αυτά, στην εικόνα
εξόδου της κλίμακας του
γκρι.
%}
for labelVal = 1:N
    redIdx = idx{labelVal};
    greenIdx = idx{labelVal}+numRows*numCols;
    blueIdx = idx{labelVal}+2*numRows*numCols;
    outputImage(redIdx) = source(redIdx);
    outputImage(greenIdx) = source(greenIdx);
    outputImage(blueIdx) = source(blueIdx);
    points = detectSURFFeatures(rgb2gray(outputImage));
    imshow(rgb2gray(outputImage)); hold on;
    imshow(outputImage); hold on;
    plot(points.selectStrongest(10));
end
```

<u>Επόμενο βήμα</u>: Βρίσκουμε αντίστοιχα σημεία μεταξύ δύο εικόνων χρησιμοποιώντας τις λειτουργίες SURF

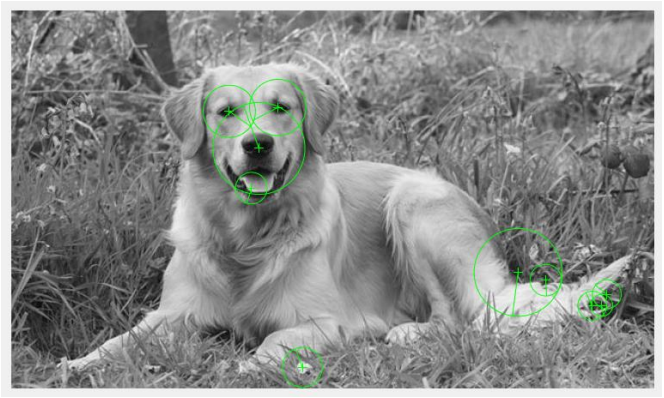
```
% Διαβάζουμε τις εικόνες
img = imread('dog.jpg');
source1 = rgb2gray(img);
img2 = imread('dog2.jpg');
source2 = rgb2gray(img2);

% Εντοπισμός χαρακτηριστικών SURF
points1 = detectSURFFeatures(source1);
points2 = detectSURFFeatures(source2);

% Εξαγωγή χαρακτηριστικών
[f1, vpts1] = extractFeatures(source1, points1);
[f2, vpts2] = extractFeatures(source2, points2);

% Αντιστοίχισης χαρακτηριστικών
indexPairs = matchFeatures(f1, f2) ;
matchedPoints1 = vpts1(indexPairs(:, 1));
matchedPoints2 = vpts2(indexPairs(:, 2));

% Οπτικοποιούμε τις υποψήφιες αντιστοιχήσεις
figure; ax = axes;
showMatchedFeatures(source1,source2,matchedPoints1,matche
dPoints2,'Parent',ax);
title(ax, 'Putative point matches');
legend(ax,'Matched points 1','Matched points 2');
```



```
figure; ax = axes;
showMatchedFeatures(source1,source2,matchedPoints1,matche
dPoints2,'montage','Parent',ax);
title(ax, 'Candidate point matches');
legend(ax, 'Matched points 1','Matched points 2');
```




```
% Αποκτάμαι την εικόνα
img = imread('dog.jpg');
source = rgb2gray(img);

% Εξαγωγή λειτουργιών SURF από μια εικόνα
points = detectSURFFeatures(source);
[features, valid_points] =
extractFeatures(source,points);

% Οπτικοποιούμε τα 10 ισχυρότερα SURF χαραλτηριστικά,
συμπεριλαμβανομένων των κλιμάκων και του προσανατολισμού
τους που καθορίστηκαν κατά τη διαδικασία εξαγωγής
περιγραφέα.
figure;
imshow(source);
hold on;
strongestPoints = valid_points.selectStrongest(10);
strongestPoints.plot('showOrientation',true);
```



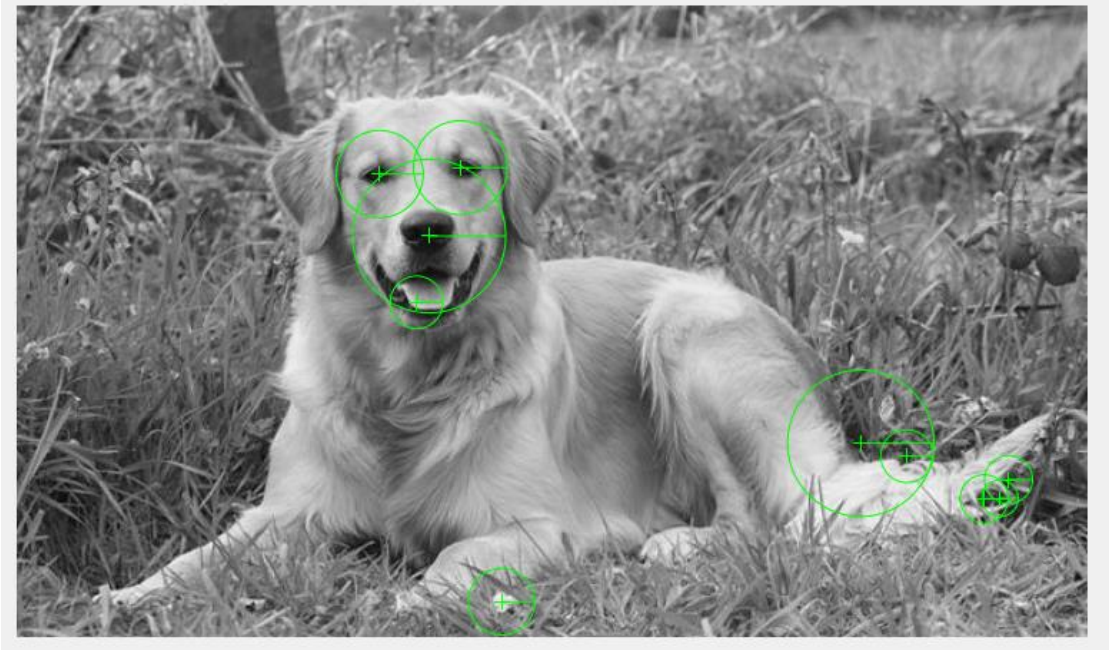
1.2)

Σε αυτό το αρχείο « SURF_Features_Texture_Extraction.m» κάναμε εξαγωγή Χαρακτηριστικών Υφής (SURF Features) από μία ασπρόμαυρη εικόνα και από μια έγχρωμη και στην συνέχεια τα αντιστοιχίσαμε.

<u>> Τεκμηρίωση κώδικα:</u>

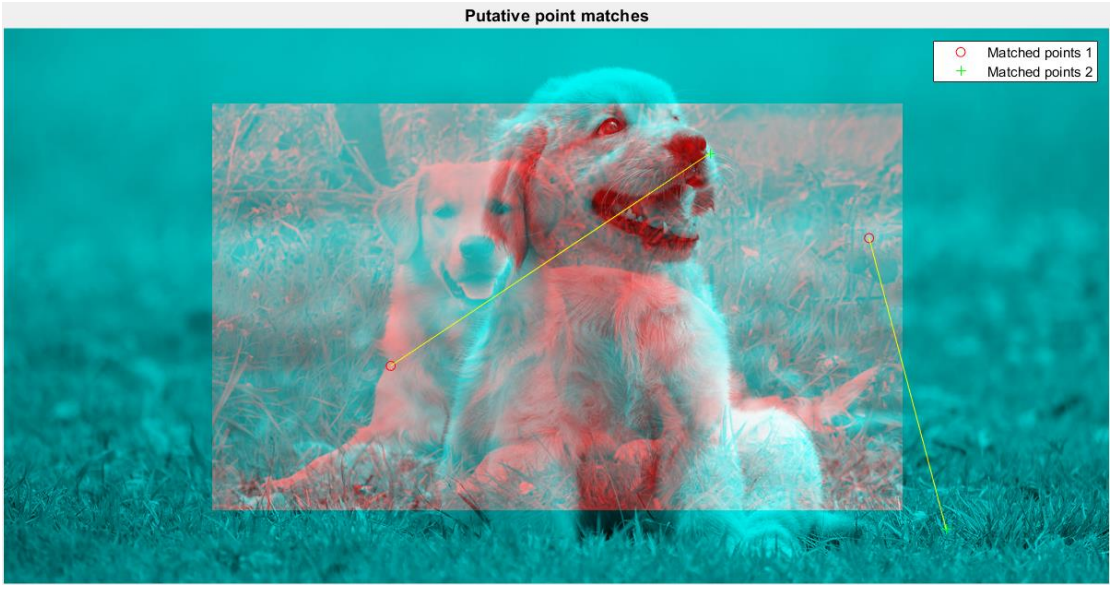
```
% Αποκτάμαι τις εικόνες
source1 = imread("dog.jpg");
source1 = rgb2gray(source1);
img = imread("dog1.jpg");
source2 = rgb2gray(img);

[L1,N1] = superpixels(source1,100);
[L2,N2] = superpixels(source2,100);

BW1 = boundarymask(L1);
BW2 = boundarymask(L2);

outputImage1 = zeros(size(source1), 'like', source1);
outputImage2 = zeros(size(img), 'like', img);

%{
Θα χρησιμοποιήσω τη συνάρτηση label2idx για να υπολογίσω
τους δείκτες των pixel σε κάθε σύμπλεγμα superpixel.
Αυτό θα μου επιτρέψει να αποκτήσω πρόσβαση στις τιμές των
κόκκινων, πράσινων και μπλε στοιχείων χρησιμοποιώντας
γραμμική ευρετηρίαση
%}
idx1 = label2idx(L1);

idx2 = label2idx(L2);
numRows = size(img,1);
numCols = size(img,2);

%{
Για καθένα από τα συμπλέγματα N superpixel,
χρησιμοποιούμε γραμμική ευρετηρίαση για να
ανακατασκευάσουμε τα αντίστοιχα pixel,
κατά την ανίχνευση / εξαγωγή χαρακτηριστικών SURF και την
εμφάνιση των 10 ισχυρότερων από αυτά, στην εικόνα σε
κλίμακα του γκρι
%}
for labelVal1 = 1:N1
    Idx = idx1{labelVal1};
    outputImage1(Idx) = source1(Idx);
    points1 = detectSURFFeatures(outputImage1);
    [f1, vpts1] = extractFeatures(outputImage1, points1);
    figure(1);
```

```
    imshow(outputImage1); hold on;
    strongestPoints1 = points1.selectStrongest(10);
    strongestPoints1.plot('showOrientation',true);
    grid;
End
```

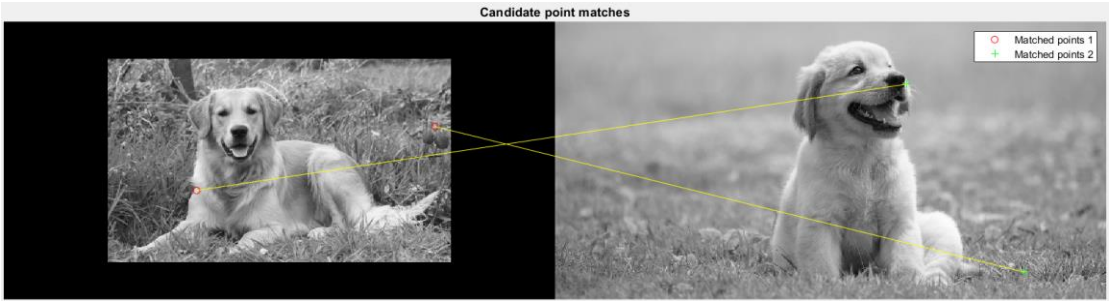


```
%{
Για καθένα από τα συμπλέγματα N superpixel,
χρησιμοποιήστε γραμμική ευρετηρίαση για πρόσβαση στα
κόκκινα, πράσινα και μπλε στοιχεία,
ανακατασκευάστε τα αντίστοιχα εικονοστοιχεία κατά την
ανίχνευση / εξαγωγή χαρακτηριστικών SURF για την έκδοση
κλίμακας του γκρι της εικόνας
και δείχνει τα 10 ισχυρότερα από αυτά, στην εικόνα
κλίμακας του γκρι εξόδου
%}
for labelVal = 1:N2
    redIdx = idx2{labelVal};
    greenIdx = idx2{labelVal}+numRows*numCols;
    blueIdx = idx2{labelVal}+2*numRows*numCols;
    outputImage2(redIdx) = img(redIdx);
    outputImage2(greenIdx) = img(greenIdx);
    outputImage2(blueIdx) = img(blueIdx);
    points2 = detectSURFFeatures(rgb2gray(outputImage2));
    [f2, vpts2] = extractFeatures(rgb2gray(outputImage2),
    points2);
    figure(2);
    imshow(outputImage2); hold on;
    strongestPoints2 = points2.selectStrongest(10);
    strongestPoints2.plot('showOrientation',true);
    grid;
end
```

```
% Αντιστοιχούμε τα χαρακτηριστικά και από τις δύο μερίες.
indexPairs1 = matchFeatures(f1, f2);
indexPairs2 = matchFeatures(f2, f1);
matchedPoints1 = vpts1(indexPairs1(:, 1));
matchedPoints2 = vpts2(indexPairs1(:, 2));
matchedPoints3 = vpts1(indexPairs2(:, 2));
matchedPoints4 = vpts2(indexPairs2(:, 1));

% Οπτικοποιούμε τα υποψήφια χαρακτηριστικά
figure; ax = axes;
showMatchedFeatures(source1,source2,matchedPoints1,matche
dPoints2,'Parent',ax);
showMatchedFeatures(source1,source2,matchedPoints3,matche
dPoints4,'Parent',ax);
title(ax, 'Putative point matches');
legend(ax,'Matched points 1','Matched points 2');
```



```
figure; ax = axes;
showMatchedFeatures(source1,source2,matchedPoints1,matche
dPoints2,'montage','Parent',ax);
showMatchedFeatures(source1,source2,matchedPoints3,matche
dPoints4,'montage','Parent',ax);
title(ax, 'Candidate point matches');
legend(ax, 'Matched points 1','Matched points 2');
```



2. Εξαγωγή Χαρακτηριστικών Υφής (Gabor Features) ανά Super Pixel.

2.1 Ανάλυση του αρχείου Superpixel_Texture_Extraction_Gabor_Features_.m

2.2 Ανάλυση του αρχείου Gabor_Features_Texture_Extraction_From_Rgb_Image.m

2.3 Ανάλυση του αρχείου Gabor_Features_Texture_Extraction_From_Gray_Image.m

Πηγές:

https://www.mathworks.com/help/images/texture-segmentation-using-gabor-filters.html

https://www.mathworks.com/help/images/ref/gabor.html

https://www.mathworks.com/help/images/ref/imgaborfilt.html

2.1)

Σε αυτό το αρχείο «Superpixel_Texture_Extraction_Gabor_Features_.m» κάναμε . Εξαγωγή Χαρακτηριστικών Υφής (Gabor Features) ανά Super Pixel.

> Τεκμηρίωση κώδικα:

Βήμα 1ο :

```
% Διαβάζουμε και εμφανίζουμε την εικόνα εισαγωγής.
source = imread('dog.jpg');
igray = rgb2gray(source);
figure
imshow(source)
```

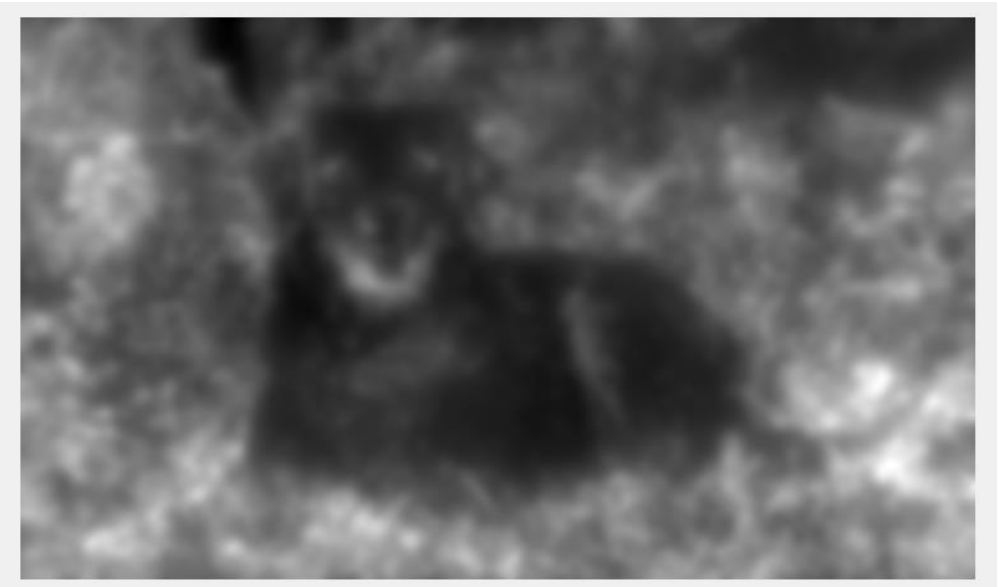
Βήμα 2ο : Σχεδίαση σειρά φίλτρων Gabor

```
%{
Σχεδιάζουμε μια σειρά φίλτρων Gabor που συντονίζονται σε
διαφορετικές συχνότητες και προσανατολισμούς.
Το σύνολο συχνοτήτων και προσανατολισμών έχει σχεδιαστεί
για να εντοπίζει διαφορετικά, περίπου ορθογώνια,
υποσύνολα πληροφοριών συχνότητας και προσανατολισμού στην
εικόνα εισαγωγής. Τακτικά δείγματα προσανατολισμών
μεταξύ [0,150] μοίρες σε βήματα των 30 μοιρών. Δείγμα
μήκους κύματος σε αυξανόμενες δυνάμεις του δύο εκκίνησης
από 4 / sqrt (2) έως το μήκος της εικόνας εισαγωγής.
%}
isize = size(source);
numRows = isize(1);
numCols = isize(2);

wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);

g = gabor(wavelength,orientation);

%{
Εξαγάγουμε χαρακτηριστικά μεγέθους Gabor από την εικόνα
προέλευσης. Όταν
εργάζόμαστε με φίλτρα Gabor, είναι σύνηθες να εργάζόμαστε
με την απόκριση μεγέθους κάθε φίλτρου. Η απόκριση
μεγέθους Gabor αναφέρεται επίσης μερικές φορές ως "Gabor
Energy".
Κάθε εικόνα εξόδου μεγέθους MxN Gabor στο gabormag (:,:,
ind) είναι η έξοδος του αντίστοιχου φίλτρου Gabor g
(ind).
%}
gabormag = imgaborfilt(igray,g);
```

<u>Βήμα 3</u>ο : Μετα-επεξεργασία των εικόνων μεγέθους Gabor σε χαρακτηριστικά Gabor

```
%{
Για να χρησιμοποιήσουμε τις αποκρίσεις μεγέθους Gabor ως
χαρακτηριστικά για χρήση στην ταξινόμηση, απαιτείται
κάποια μετα-επεξεργασία.
Αυτή η επεξεργασία δημοσιεύσεων περιλαμβάνει εξομάλυνση
Gauss, προσθέτοντας επιπλέον χωρικές πληροφορίες στο
σύνολο χαρακτηριστικών,
αναδιαμόρφωση του συνόλου λειτουργιών μας στη φόρμα που
αναμένεται από τις
συναρτήσεις pca και kmeans και ομαλοποίηση των
πληροφοριών χαρατκηριστικών σε μια κοινή διακύμανση και
μέσο όρο.

Κάθε εικόνα μεγέθους Gabor περιέχει κάποιες τοπικές
παραλλαγές, ακόμη και σε περιοχές με σταθερή υφή.
Αυτές οι τοπικές παραλλαγές θα απορρίψουν την
τμηματοποίηση. Μπορούμε να αντισταθμίσουμε αυτές τις
παραλλαγές χρησιμοποιώντας απλό
Φιλτράρισμα χαμηλού περάσματος Gauss για εξομάλυνση των
πληροφοριών μεγέθους Gabor. Επιλέγουμε ένα σίγμα που
ταιριάζει
στο φίλτρο Gabor που εξήγαγε κάθε δυνατότητα.
Παρουσιάζουμε έναν όρο εξομάλυνσης K που ελέγχει πόσο
εξομάλυνση
εφαρμόζεται στις αποκρίσεις μεγέθους Gabor.
%}
for i = 1:length(g)
    sigma = 0.5*g(i).Wavelength;
    K = 3;
    gabormag(:,:,i) =
imgaussfilt(gabormag(:,:,i),K*sigma);
end


%{
Κατά την κατασκευή συνόλων χαρακτηριστικών Gabor για
ταξινόμηση, είναι χρήσιμο να προσθέσουμε έναν χάρτη
πληροφοριών χωρικής θέσης τόσο στα X όσο και στα Y.
Αυτές οι πρόσθετες πληροφορίες επιτρέπουν στον ταξινομητή
να προτιμά ομαδοποιήσεις που είναι κοντά μεταξύ τους
χωρικά.
%}
X = 1:numCols;
Y = 1:numRows;
[X,Y] = meshgrid(X,Y);
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);
```

```
%{
Αναμορφώνουμε τα δεδομένα σε έναν πίνακα X της φόρμας που
αναμένεται από τη συνάρτηση kmeans. Κάθε εικονοστοιχείο
στο πλέγμα εικόνας είναι ξεχωριστό σημείο δεδομένων,
και κάθε επίπεδο στο μεταβλητό χαρακτηριστικό είναι ένα
ξεχωριστό χαρακτηριστικό. Σε αυτό το παράδειγμα, υπάρχει
μια ξεχωριστή δυνατότητα για κάθε φίλτρο
στην τράπεζα φίλτρων Gabor, συν δύο επιπλέον δυνατότητες
από τις χωρικές πληροφορίες που προστέθηκαν στο
προηγούμενο βήμα.
Συνολικά, υπάρχουν 24 χαρακτηριστικά Gabor και 2 χωρικά
χαρακτηριστικά για κάθε pixel στην εικόνα εισαγωγής.
%}
numPoints = numRows*numCols;
X = reshape(featureSet,numRows*numCols,[]);

% Ομαλοποίηση των χαρακτηριστικών ως μηδενικός μέσος
όρος, διακύμανση μονάδας.
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));

%{
Οπτικοποιούμε το σύνολο δυνατοτήτων. Για να κατανοήσουμε
πώς μοιάζουν τα χαρακτηριστικά μεγέθους Gabor, μπορεί να
χρησιμοποιηθεί η Ανάλυση Κύριων Συστατικών
για μετακίνηση από μια αναπαράσταση 26-D κάθε
εικονοστοιχείου στην εικόνα εισόδου σε τιμή έντασης 1-D
για κάθε εικονοστοιχείο.
%}
coeff = pca(X);
feature2DImage = reshape(X*coeff(:,1),numRows,numCols);
figure
imshow(feature2DImage,[])
```



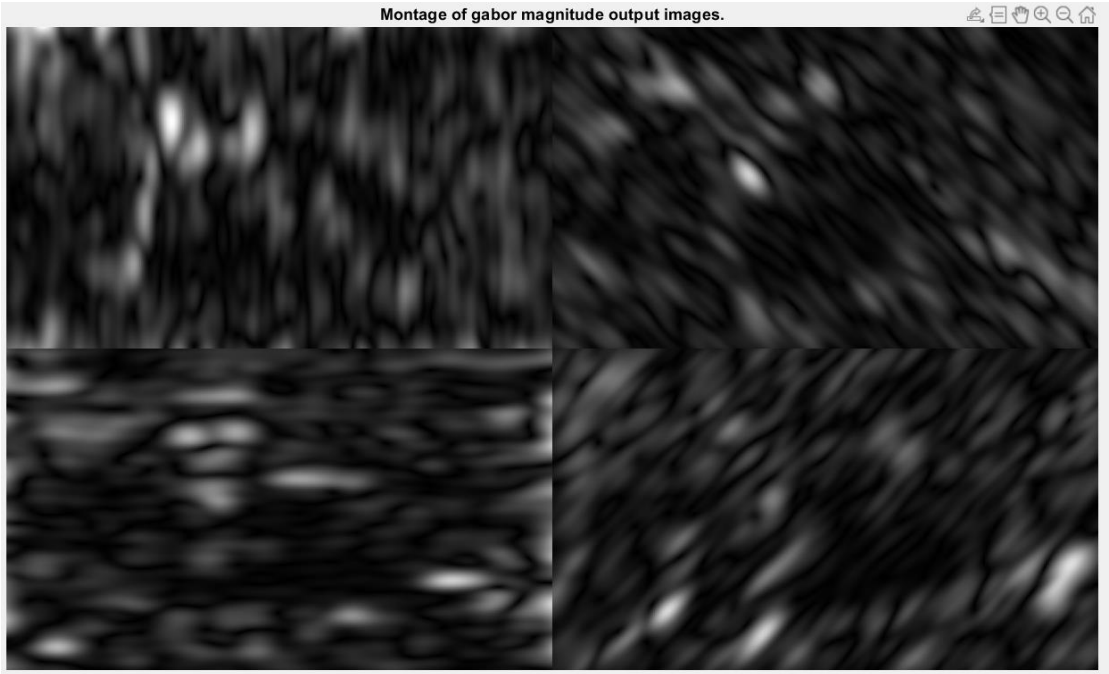
Βήμα 4ο Ταξινόμηση χαρακτηριστικών υφής Gabor χρησιμοποιώντας kmeans

```
%{
Επαναλαμβάνουμε την ομαδοποίηση k-means πέντε φορές για
να αποφύγουμε τοπικά ελάχιστα κατά την αναζήτηση μέσων
που ελαχιστοποιούν την αντικειμενική λειτουργία.
Η μόνη προηγούμενη πληροφορία που υποτίθεται σε αυτό το
παράδειγμα είναι πόσες διαφορετικές περιοχές υφής
υπάρχουν στην εικόνα που είναι τμηματοποιημένη.
Υπάρχουν δύο διαφορετικές περιοχές σε αυτήν την
περίπτωση.
%}
L = kmeans(X,2,'Replicates',5);

% Οπτικοποιούμε την τμηματοποίηση χρησιμοποιώντας
label2rgb.
L = reshape(L,[numRows numCols]);
figure
imshow(label2rgb(L))
```

```
%{
Οπτικοποιούμε την τμηματοποιημένη εικόνα χρησιμοποιώντας
imshowpair. Εξετάζουμε τις εικόνες προσκηνίου και φόντου
που προκύπτουν από τη μάσκα BW που σχετίζεται
με την ετικέτα matrix L.
%}
Aseg1 = zeros(size(source),'like',source);
Aseg2 = zeros(size(source),'like',source);
BW = L == 2;
BW = repmat(BW,[1 1 3]);
Aseg1(BW) = source(BW);
Aseg2(~BW) = source(~BW);
figure
imshowpair(Aseg1,Aseg2,'montage');
```

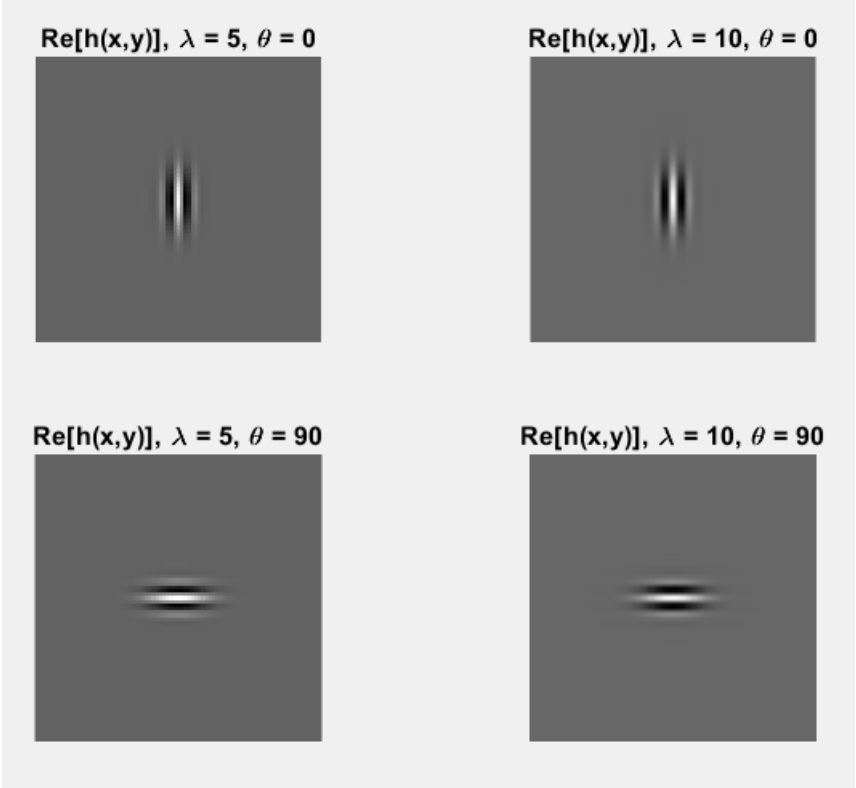
Βήμα 5ο : % Κατασκευή σειράς φίλτρου Gabor και εφαρμογή σε εικόνα εισαγωγής

```
source = imread('dog.jpg');
source = rgb2gray(source);
% Δημιουργία μιας σειράς φίλτρων Gabor
wavelength = 20;
orientation = [0 45 90 135];
g = gabor(wavelength,orientation);

% Εφαρμόζουμε τα φίλτρα στην εικόνα
outMag = imgaborfilt(source,g);

% Εμφάνιση των αποτελεσμάτων
outSize = size(outMag);
outMag = reshape(outMag,[outSize(1:2),1,outSize(3)]);
figure, montage(outMag,'DisplayRange',[]);
title('Montage of gabor magnitude output images.');
```



Βήμα 6ο : Κατασκευή φίλτρου Gabor Array και οπτικοποιούμε το μήκος κύματος και τον προσανατολισμό

```
% Δημιουργία σειράς φίλτρων Gabor
g = gabor([5 10],[0 90]);

% Οπτικοποιούμε το πραγματικό μέρος του χωρικού πυρήνα
της συνέλιξης κάθε φίλτρου Gabor στον πίνακα
figure;
subplot(2,2,1)
for p = 1:length(g)
    subplot(2,2,p);
    imshow(real(g(p).SpatialKernel),[]);
    lambda = g(p).Wavelength;
    theta  = g(p).Orientation;
    title(sprintf('Re[h(x,y)], \\lambda = %d, \\theta =
%d',lambda,theta));
end
```

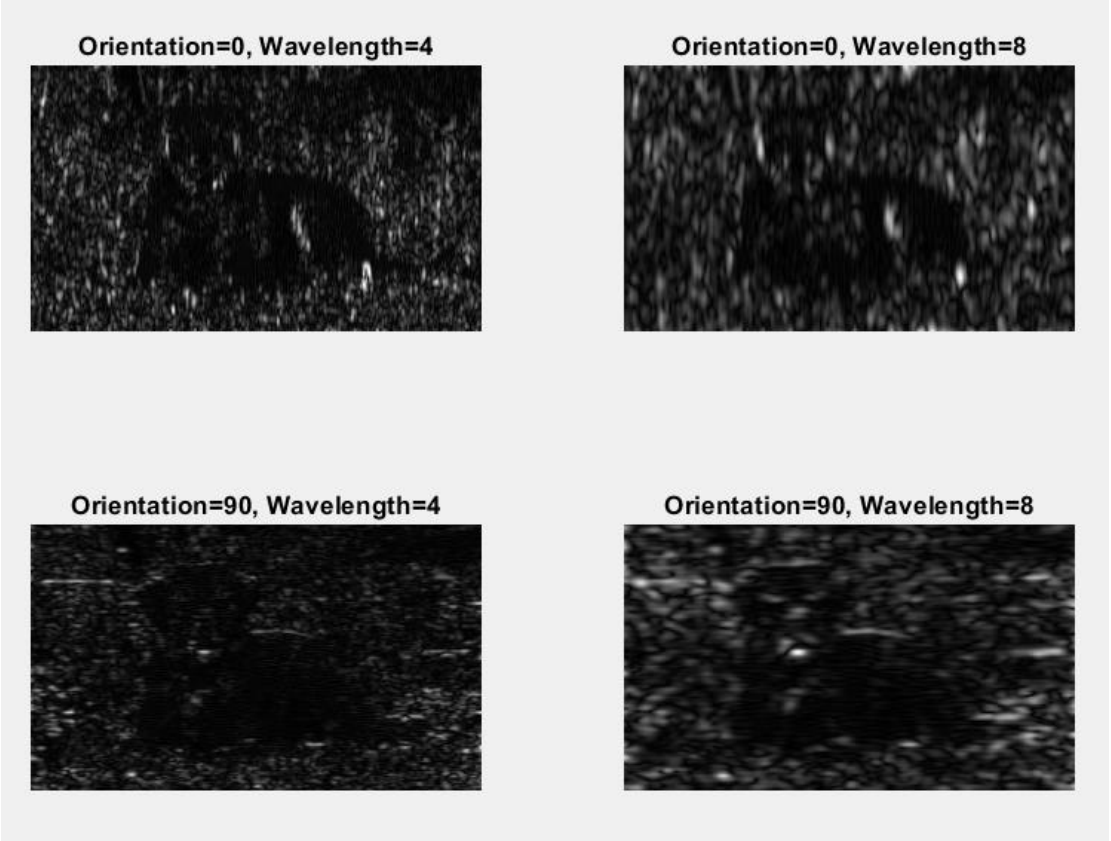


Βήμα 7ο : Εφαρμογή ενός φίλτρου Gabor σε εικόνα εισαγωγής

```
% Διαβάζουμε την εικόνα στο χώρο εργασίας
source = imread('dog1.jpg');


% Μετατροπή εικόνας σε κλίμακα του γκρι.
source = rgb2gray(source);


% Εφαρμογή φίλτρου Gabor στην εικόνα.
wavelength = 4;
orientation = 90;
[mag,phase] = imgaborfilt(source,wavelength,orientation);

% Εμφάνιση αρχικής εικόνας με γραφικές παραστάσεις του
μεγέθους και της φάσης που υπολογίζονται από το φίλτρο
Gabor
figure
subplot(1,3,1);
imshow(source);
title('Original Image');
subplot(1,3,2);
imshow(mag,[])
title('Gabor magnitude');
subplot(1,3,3);
imshow(phase,[]);
title('Gabor phase');
```

Βήμα 8ο : Εφαρμογή συστοιχίας φίλτρων Gabor στην εικόνα εισαγωγής

```
% Διαβάζουμε την εικόνα στο workspace
source = imread('dog.jpg');
source = rgb2gray(source);
% Δημιουργία σειράς φίλτρων Gabor, που ονομάζεται τράπεζα
φίλτρων. Αυτή η τράπεζα φίλτρων περιέχει δύο
προσανατολισμούς και δύο μήκη κύματος
gaborArray = gabor([4 8],[0 90]);

% Εφαρμογή φίλτρων στην εισαγωγή εικόνας
gaborMag = imgaborfilt(source,gaborArray);

% Αποτελέσματα εμφάνισης. Το σχήμα δείχνει την απόκριση
μεγέθους για κάθε φίλτρο
figure
subplot(2,2,1);
for p = 1:4
    subplot(2,2,p)
    imshow(gaborMag(:,:,p),[]);
    theta = gaborArray(p).Orientation;
    lambda = gaborArray(p).Wavelength;
    title(sprintf('Orientation=%d,
Wavelength=%d',theta,lambda));
end
```



2.2)

Σε αυτό το αρχείο «Gabor_Features_Texture_Extraction_From_Rgb_Image.m» κάνουμε εξαγωγή των χαρακτηριστικών Gabor από μια έγχρωμη εικόνα .

Αρχικά αυτό το κάνουμε έτσι ώστε να εξάγουμε τα χαρακτηριστικά από τα έγχρωμα superpixels για να εκπαιδεύσουμε το SVM.

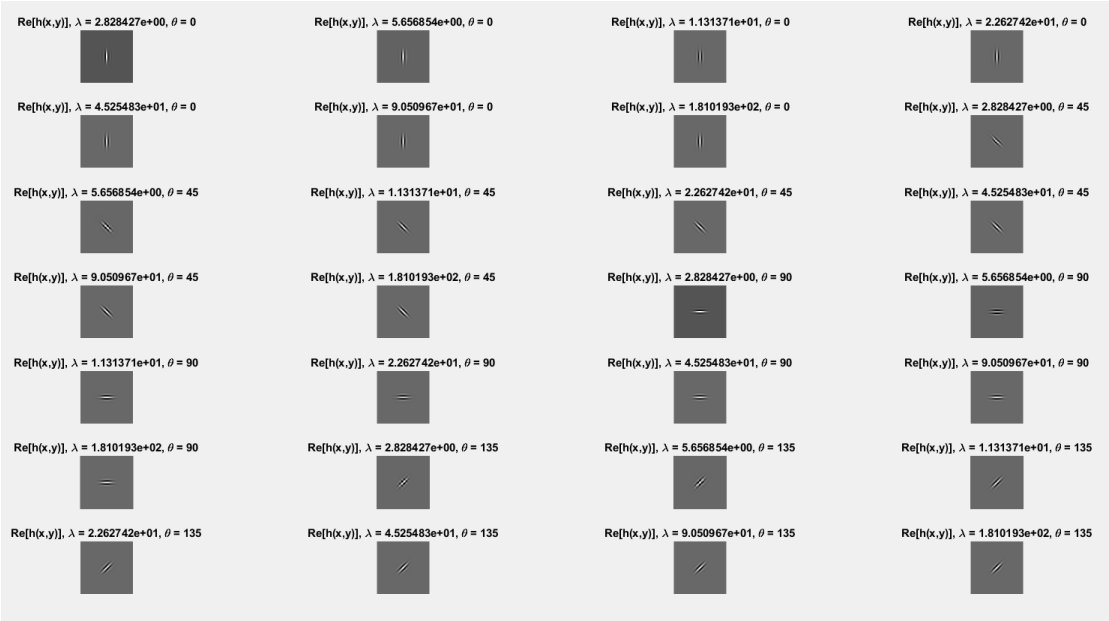
> Τεκμηρίωση κώδικα :

```
% Ανάγνωση εικόνας
img = imread('dog.jpg');

source = rgb2gray(img);
```

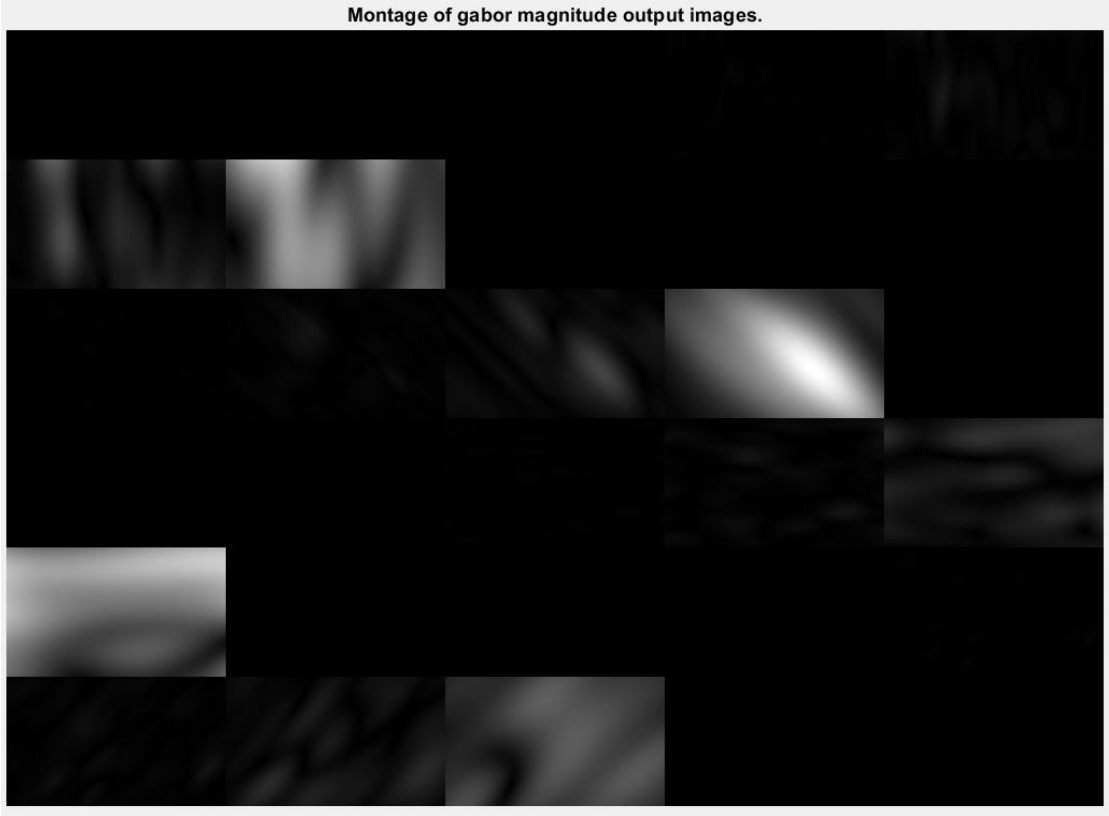
Βήμα 1ο: Σχεδίαση σειράς φίλτρων Gabor

```
%{
Σχεδιάζουμε μια σειρά φίλτρων Gabor που συντονίζονται
σε διαφορετικές συχνότητες και προσανατολισμούς.
Το σύνολο συχνοτήτων και προσανατολισμών έχει
σχεδιαστεί για να εντοπίζει διαφορετικά, περίπου
ορθογώνια, υποσύνολα πληροφοριών συχνότητας και
προσανατολισμού στην εικόνα εισαγωγής. Τακτικά
δείγματα προσανατολισμών μεταξύ [0,150] μοίρες σε
βήματα των 30 μοιρών. Δείγμα μήκους κύματος σε
αυξανόμενες δυνάμεις δύο εκκίνησης από 4 / sqrt (2)
έως το μήκος της εικόνας εισαγωγής.
%}
isize = size(source);
numRows = isize(1);
numCols = isize(2);

wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);

c = length(wavelength);
r = length(orientation);

g = gabor(wavelength,orientation);
```

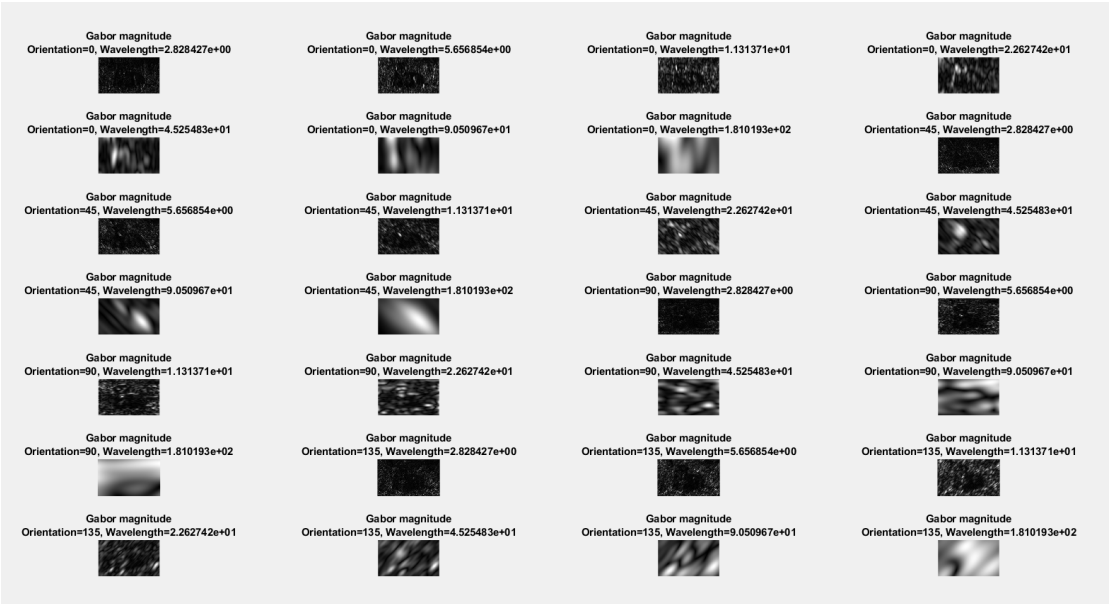
```
% Οπτικοποιούμε το πραγματικό μέρος του χωρικού
πυρήνα της συνέλιξης κάθε φίλτρου Gabor στον πίνακα

figure(1);
subplot(c,r,1)
for p = 1:length(g)
    subplot(c,r,p);
    imshow(real(g(p).SpatialKernel),[]);
    lambda = g(p).Wavelength;
    theta  = g(p).Orientation;
    title(sprintf('Re[h(x,y)], \\lambda = %d, \\theta
= %d',lambda,theta));
end
```



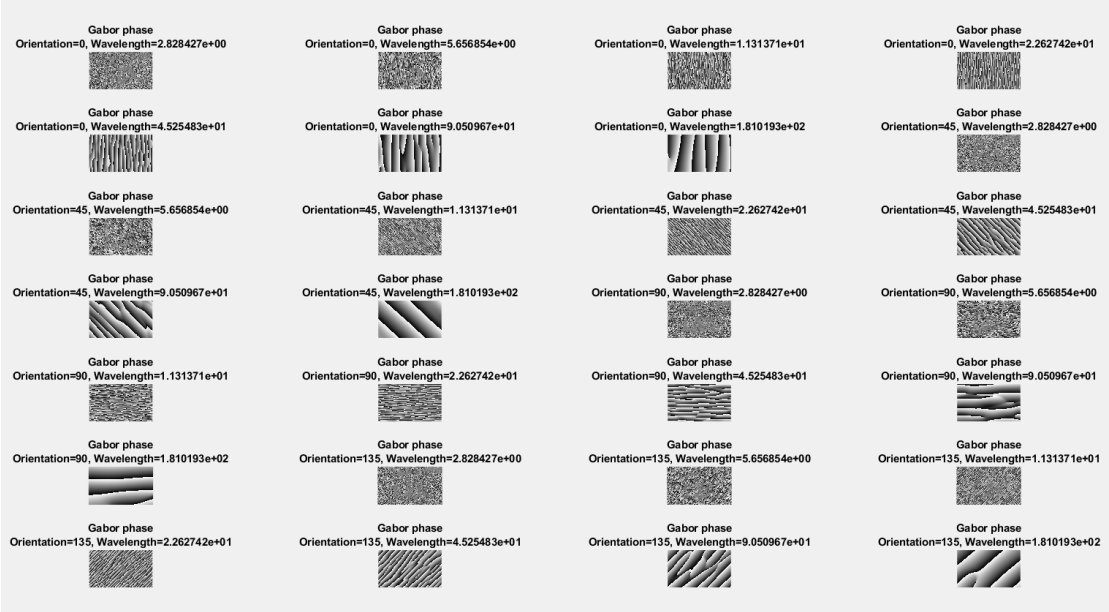
```
% Εμφάνιση των αποτελεσμάτων μεγέθους

gabormag = imgaborfilt(source,g);
outSize = size(gabormag);
gm = reshape(gabormag,[outSize(1:2),1,outSize(3)]);
figure(2), montage(gm,'DisplayRange',[]);
title('Montage of gabor magnitude output images.');
```



```
% Εμφάνιση του μεγέθους που υπολογίζεται από το φίλτρο
Gabor

figure(3);
subplot(c,r,1)
for p = 1:length(g)
    [mag,phase] = imgaborfilt(source,g(p));
    subplot(c,r,p);
    imshow(mag,[])
    theta = g(p).Orientation;
    lambda = g(p).Wavelength;
    title(sprintf('Gabor magnitude\nOrientation=%d,
Wavelength=%d',theta,lambda));
end
```



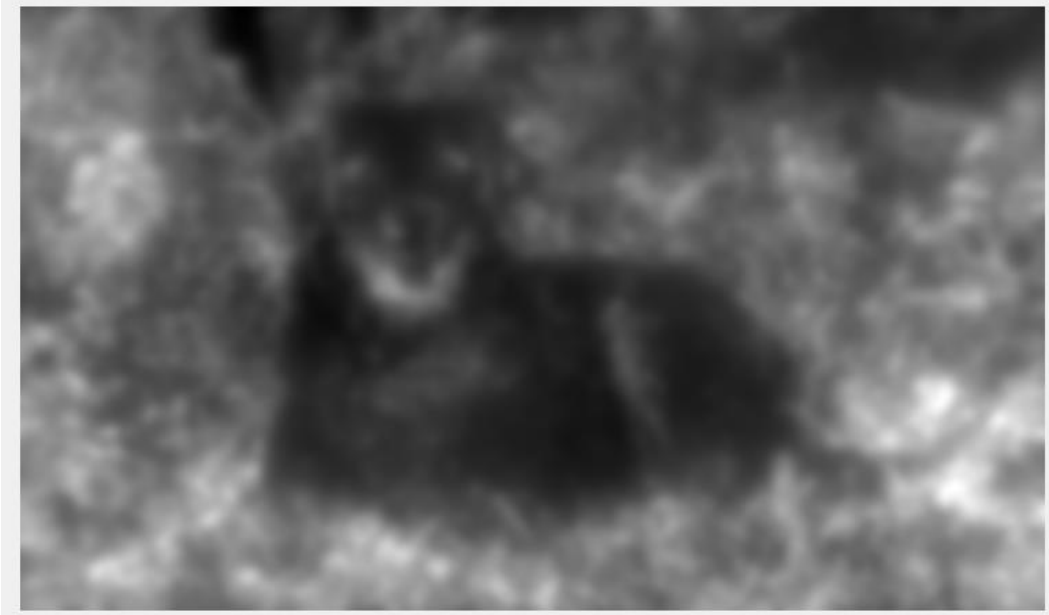
```
% Εμφάνιση της φάσης που υπολογίζεται από το φίλτρο Gabor

figure(4);
subplot(c,r,1)
for p = 1:length(g)
    [mag,phase] = imgaborfilt(source,g(p));
    subplot(c,r,p);
    imshow(phase,[]);
    theta = g(p).Orientation;
    lambda = g(p).Wavelength;
    title(sprintf('Gabor phase\nOrientation=%d,
Wavelength=%d',theta,lambda));
end
```



<u>Βήμα 2ο</u>: Μετα-επεξεργασία των εικόνων μεγέθους Gabor σε χαρακτηριστικά Gabor

```
%{
Για να χρησιμοποιήσουμε τις αποκρίσεις μεγέθους Gabor ως
χαρακτηριστικά για χρήση στην ταξινόμηση, απαιτείται
κάποια μετα-επεξεργασία.
Αυτή η επεξεργασία δημοσιεύσεων περιλαμβάνει εξομάλυνση
Gauss, προσθέτοντας επιπλέον χωρικές πληροφορίες στο
σύνολο χαρακτηριστικών,αναδιαμόρφωση του συνόλου
λειτουργιών μας στη φόρμα που αναμένεται από τις
συναρτήσεις pca και kmeans και ομαλοποίηση των
πληροφοριών χαρατκηριστικών σε μια κοινή διακύμανση και
μέσο όρο.

Κάθε εικόνα μεγέθους Gabor περιέχει κάποιες τοπικές
παραλλαγές, ακόμη και σε περιοχές με σταθερή υφή.
Αυτές οι τοπικές παραλλαγές θα απορρίψουν την
τμηματοποίηση. Μπορούμε να αντισταθμίσουμε αυτές τις
παραλλαγές χρησιμοποιώντας απλό φιλτράρισμα χαμηλού
περάσματος Gauss για εξομάλυνση των πληροφοριών μεγέθους
Gabor. Επιλέγουμε ένα σίγμα που ταιριάζει
στο φίλτρο Gabor που εξήγαγε κάθε δυνατότητα.
Παρουσιάζουμε έναν όρο εξομάλυνσης K που ελέγχει πόσο
εξομάλυνσηεφαρμόζεται στις αποκρίσεις μεγέθους Gabor.
%}
for i = 1:length(g)
    sigma = 0.5*g(i).Wavelength;
    K = 3;
    gabormag(:,:,i) =
imgaussfilt(gabormag(:,:,i),K*sigma);
end

%{
Κατά την κατασκευή συνόλων χαρακτηριστικών Gabor για
ταξινόμηση, είναι χρήσιμο να προσθέσουμε έναν χάρτη
πληροφοριών χωρικής θέσης τόσο στα X όσο και στα Y.
Αυτές οι πρόσθετες πληροφορίες επιτρέπουν στον ταξινομητή
να προτιμά ομαδοποιήσεις που είναι κοντά μεταξύ τους
χωρικά.
%}
X = 1:numCols;
Y = 1:numRows;
[X,Y] = meshgrid(X,Y);
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);
```

```
%{
Αναμορφώνουμε τα δεδομένα σε έναν πίνακα X της φόρμας που
αναμένεται από τη συνάρτηση kmeans. Κάθε εικονοστοιχείο
στο πλέγμα εικόνας είναι ξεχωριστό σημείο δεδομένων,
και κάθε επίπεδο στο μεταβλητό χαρακτηριστικό είναι ένα
ξεχωριστό χαρακτηριστικό. Σε αυτό το παράδειγμα, υπάρχει
μια ξεχωριστή δυνατότητα για κάθε φίλτροστην τράπεζα
φίλτρων Gabor, συν δύο επιπλέον δυνατότητες από τις
χωρικές πληροφορίες που προστέθηκαν στο προηγούμενο βήμα.
Συνολικά, υπάρχουν 24 χαρακτηριστικά Gabor και 2 χωρικά
χαρακτηριστικά για κάθε pixel στην εικόνα εισαγωγής.
%}
numPoints = numRows*numCols;
X = reshape(featureSet,numRows*numCols,[]);

% Ομαλοποίηση των χαρακτηριστικών ως μηδενικός μέσος
όρος, διακύμανση μονάδας.
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));

%{
Οπτικοποιούμε το σύνολο δυνατοτήτων. Για να κατανοήσουμε
πώς μοιάζουν τα χαρακτηριστικά μεγέθους Gabor, μπορεί να
χρησιμοποιηθεί η Ανάλυση Κύριων Συστατικών για μετακίνηση
από μια αναπαράσταση 26-D κάθε εικονοστοιχείου στην
εικόνα εισόδου σε τιμή έντασης 1-D για κάθε
εικονοστοιχείο.
%}
coeff = pca(X);
feature2DImage = reshape(X*coeff(:,1),numRows,numCols);
figure(5)
imshow(feature2DImage,[])
```



<u>Βήμα 3ο</u>: Ταξινόμηση χαρακτηριστικών υφής Gabor χρησιμοποιώντας kmeans

```
%{
Επαναλαμβάνουμε την ομαδοποίηση k-means πέντε φορές
για να αποφύγουμε τοπικά ελάχιστα κατά την αναζήτηση
μέσων που ελαχιστοποιούν την αντικειμενική
λειτουργία.
Η μόνη προηγούμενη πληροφορία που υποτίθεται σε αυτό
το παράδειγμα είναι πόσες διαφορετικές περιοχές υφής
υπάρχουν στην εικόνα που είναι τμηματοποιημένη.
Υπάρχουν δύο διαφορετικές περιοχές σε αυτήν την
περίπτωση.
%}
L = kmeans(X,2,'Replicates',5);

% Οπτικοποιούμε την τμηματοποίηση χρησιμοποιώντας
label2rgb.
L = reshape(L,[numRows numCols]);
figure(6)
imshow(label2rgb(L))
```

```
%{
Οπτικοποιούμε την τμηματοποιημένη εικόνα
χρησιμοποιώντας imshowpair. Εξετάζουμε τις εικόνες
προσκηνίου και φόντου που προκύπτουν από τη μάσκα BW
που σχετίζεται
με την ετικέτα matrix L.
%}
Aseg1 = zeros(size(img),'like',img);
Aseg2 = zeros(size(img),'like',img);
BW = L == 2;
BW = repmat(BW,[1 1 3]);
Aseg1(BW) = img(BW);
Aseg2(~BW) = img(~BW);
figure(7)
imshowpair(Aseg1,Aseg2,'montage');
```

2.3)

Σε αυτό το αρχείο «Gabor_Features_Texture_Extraction_From_Gray_Image.m» κάνουμε εξαγωγή των χαρακτηριστικών Gabor από μια ασπρόμαυρη εικόνα .

Αυτό το κάνουμε για να εξάγουμε χαρακτηριστικά από τα ασπρόμαυρα superpixels και να τα περάσουμε στο εκπαιδευμένο SVM για να μαντέψει το χρώμα των superpixel το οποίο θα είναι ένα από τα διακριτά χρώματα.

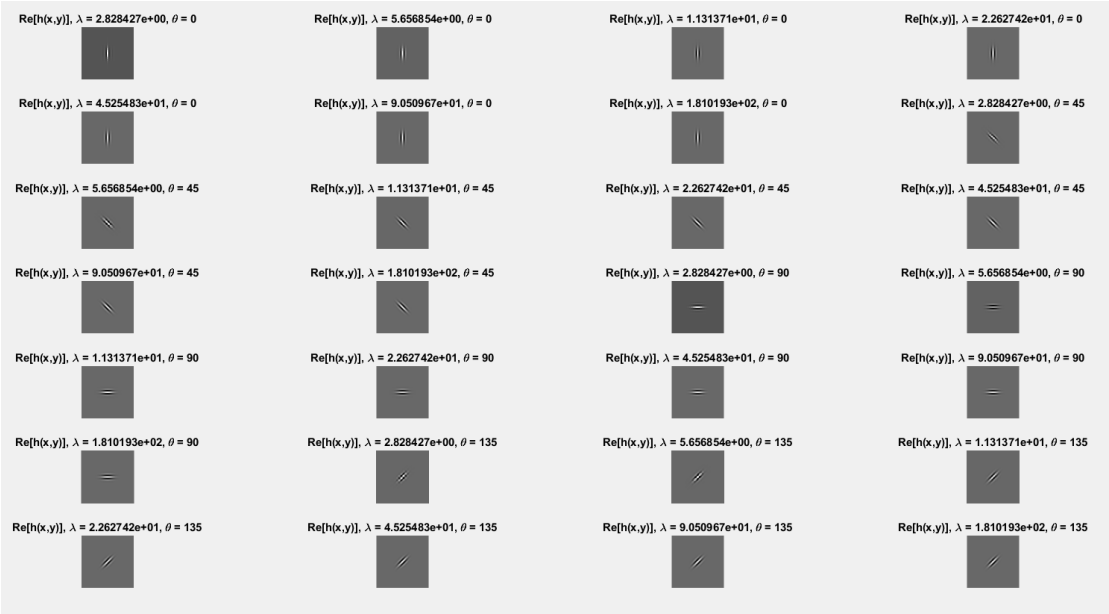
> Τεκμηρίωση κώδικα:

```
% Ανάγνωση εικόνας
source = imread('dog.jpg');
source = rgb2gray(source);
```

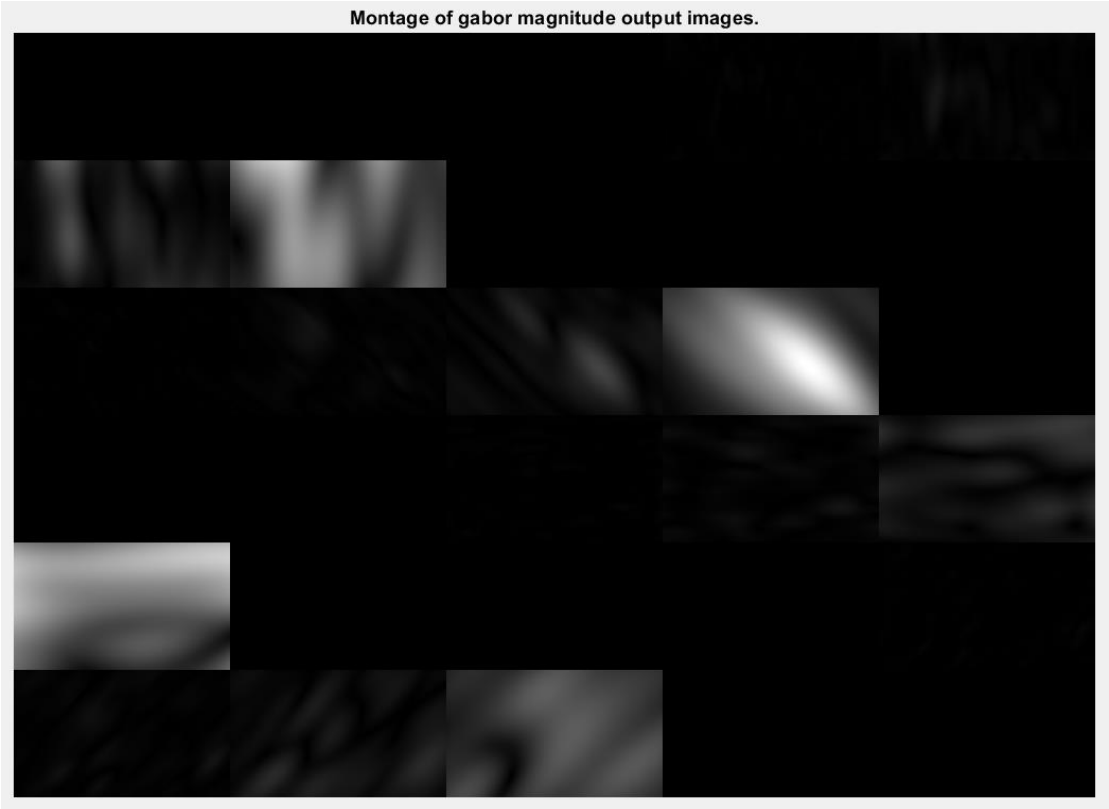
Βήμα 1ο : Σχεδίαση σειράς φίλτρων Gabor

```
%{
Σχεδιάζουμε μια σειρά φίλτρων Gabor που συντονίζονται σε
διαφορετικές συχνότητες και προσανατολισμούς.
Το σύνολο συχνοτήτων και προσανατολισμών έχει σχεδιαστεί
για να εντοπίζει διαφορετικά, περίπου ορθογώνια,
υποσύνολα πληροφοριών συχνότητας και προσανατολισμού στην
εικόνα εισαγωγής. Τακτικά δείγματα προσανατολισμών
μεταξύ [0,150] μοίρες σε βήματα των 30 μοιρών. Δείγμα
μήκους κύματος σε αυξανόμενες δυνάμεις δύο εκκίνησης
από 4 / sqrt (2) έως το μήκος της εικόνας εισαγωγής.
%}
isize = size(source);
numRows = isize(1);
numCols = isize(2);

wavelengthMin = 4/sqrt(2);
wavelengthMax = hypot(numRows,numCols);
n = floor(log2(wavelengthMax/wavelengthMin));
wavelength = 2.^(0:(n-2)) * wavelengthMin;

deltaTheta = 45;
orientation = 0:deltaTheta:(180-deltaTheta);

c = length(wavelength);
r = length(orientation);

g = gabor(wavelength,orientation);
```

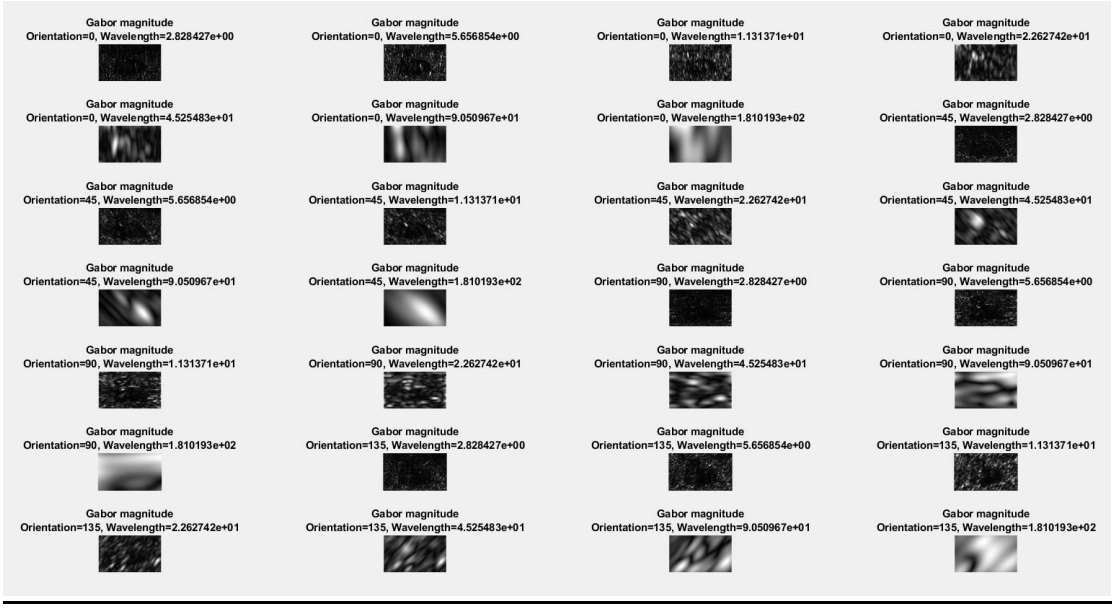
```
% Οπτικοποιούμε το πραγματικό μέρος του χωρικού πυρήνα
της συνέλιξης κάθε φίλτρου Gabor στον πίνακα
figure(1);
subplot(c,r,1)
for p = 1:length(g)
    subplot(c,r,p);
    imshow(real(g(p).SpatialKernel),[]);
    lambda = g(p).Wavelength;
    theta  = g(p).Orientation;
    title(sprintf('Re[h(x,y)], \\lambda = %d, \\theta =
%d',lambda,theta));
end
```



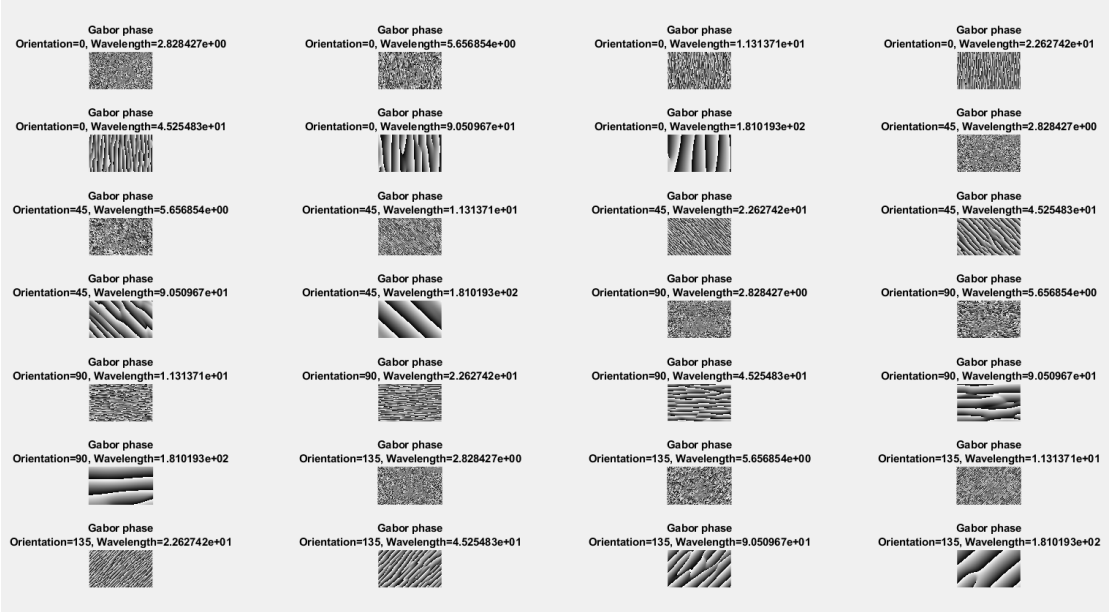
```
% Εμφάνιση των αποτελεσμάτων μεγέθους

gabormag = imgaborfilt(source,g);
outSize = size(gabormag);
gm = reshape(gabormag,[outSize(1:2),1,outSize(3)]);
figure(2), montage(gm,'DisplayRange',[]);
title('Montage of gabor magnitude output images.');
```



```
% Εμφάνιση του μεγέθους που υπολογίζεται από το
φίλτρο Gabor
figure(3);
subplot(c,r,1)
for p = 1:length(g)
    [mag,phase] = imgaborfilt(source,g(p));
    subplot(c,r,p);
    imshow(mag,[])
    theta = g(p).Orientation;
    lambda = g(p).Wavelength;
    title(sprintf('Gabor magnitude\nOrientation=%d,
Wavelength=%d',theta,lambda));
end
```



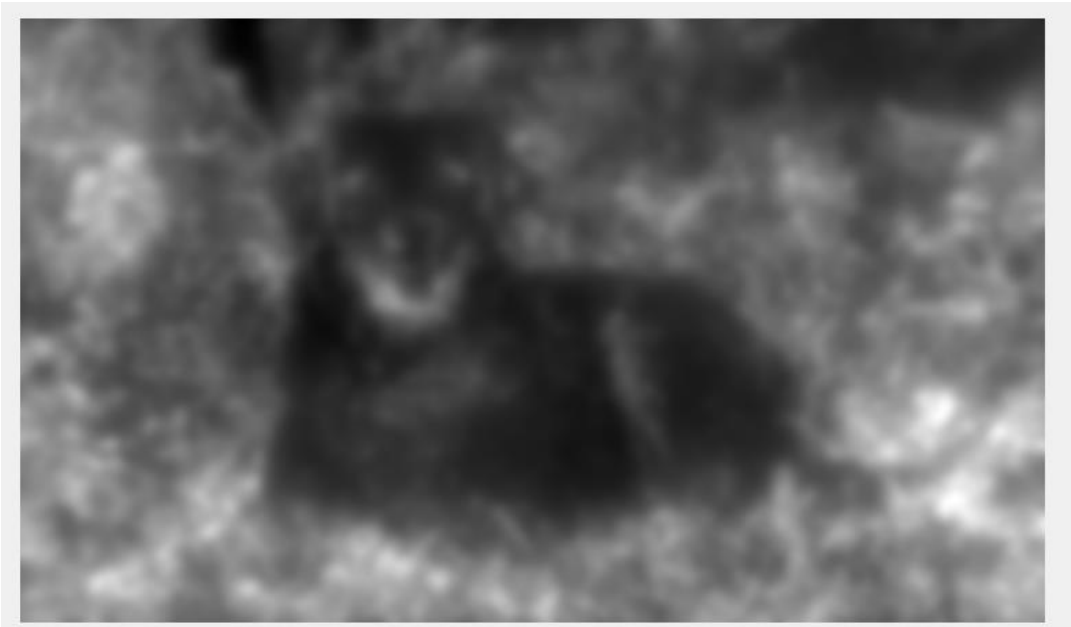
```
% Εμφάνιση της φάσης που υπολογίζεται από το φίλτρο Gabor

figure(4);
subplot(c,r,1)
for p = 1:length(g)
    [mag,phase] = imgaborfilt(source,g(p));
    subplot(c,r,p);
    imshow(phase,[]);
    theta = g(p).Orientation;
    lambda = g(p).Wavelength;
    title(sprintf('Gabor phase\nOrientation=%d,
Wavelength=%d',theta,lambda));
end
```



<u>Βήμα 2ο</u>: Μετα-επεξεργασία των εικόνων μεγέθους Gabor σε χαρακτηριστικά Gabor

```
%{
Για να χρησιμοποιήσουμε τις αποκρίσεις μεγέθους Gabor ως
χαρακτηριστικά για χρήση στην ταξινόμηση, απαιτείται
κάποια μετα-επεξεργασία.
Αυτή η επεξεργασία δημοσιεύσεων περιλαμβάνει εξομάλυνση
Gauss, προσθέτοντας επιπλέον χωρικές πληροφορίες στο
σύνολο χαρακτηριστικών,αναδιαμόρφωση του συνόλου
λειτουργιών μας στη φόρμα που αναμένεται από τις
συναρτήσεις pca και kmeans και ομαλοποίηση των
πληροφοριών χαρατκηριστικών σε μια κοινή διακύμανση και
μέσο όρο.

Κάθε εικόνα μεγέθους Gabor περιέχει κάποιες τοπικές
παραλλαγές, ακόμη και σε περιοχές με σταθερή υφή.
Αυτές οι τοπικές παραλλαγές θα απορρίψουν την
τμηματοποίηση. Μπορούμε να αντισταθμίσουμε αυτές τις
παραλλαγές χρησιμοποιώντας απλό φιλτράρισμα χαμηλού
περάσματος Gauss για εξομάλυνση των πληροφοριών μεγέθους
Gabor. Επιλέγουμε ένα σίγμα που ταιριάζει
στο φίλτρο Gabor που εξήγαγε κάθε δυνατότητα.
Παρουσιάζουμε έναν όρο εξομάλυνσης K που ελέγχει πόσο
εξομάλυνσηεφαρμόζεται στις αποκρίσεις μεγέθους Gabor.
%}
for i = 1:length(g)
    sigma = 0.5*g(i).Wavelength;
    K = 3;
    gabormag(:,:,i) =
imgaussfilt(gabormag(:,:,i),K*sigma);
end

%{
Κατά την κατασκευή συνόλων χαρακτηριστικών Gabor για
ταξινόμηση, είναι χρήσιμο να προσθέσουμε έναν χάρτη
πληροφοριών χωρικής θέσης τόσο στα X όσο και στα Y.
Αυτές οι πρόσθετες πληροφορίες επιτρέπουν στον ταξινομητή
να προτιμά ομαδοποιήσεις που είναι κοντά μεταξύ τους
χωρικά.
%}
X = 1:numCols;
Y = 1:numRows;
[X,Y] = meshgrid(X,Y);
featureSet = cat(3,gabormag,X);
featureSet = cat(3,featureSet,Y);
```

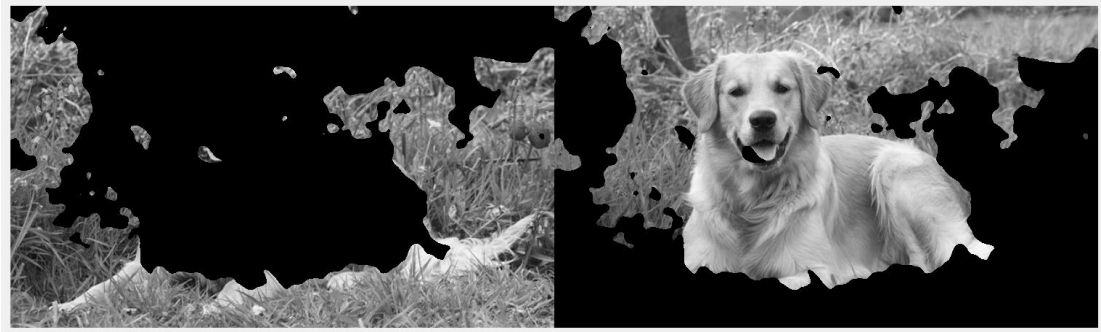
```
%{
Αναμορφώνουμε τα δεδομένα σε έναν πίνακα X της φόρμας που
αναμένεται από τη συνάρτηση kmeans. Κάθε εικονοστοιχείο
στο πλέγμα εικόνας είναι ξεχωριστό σημείο δεδομένων,
και κάθε επίπεδο στο μεταβλητό χαρακτηριστικό είναι ένα
ξεχωριστό χαρακτηριστικό. Σε αυτό το παράδειγμα, υπάρχει
μια ξεχωριστή δυνατότητα για κάθε φίλτροστην τράπεζα
φίλτρων Gabor, συν δύο επιπλέον δυνατότητες από τις
χωρικές πληροφορίες που προστέθηκαν στο προηγούμενο βήμα.
Συνολικά, υπάρχουν 24 χαρακτηριστικά Gabor και 2 χωρικά
χαρακτηριστικά για κάθε pixel στην εικόνα εισαγωγής.
%}
numPoints = numRows*numCols;
X = reshape(featureSet,numRows*numCols,[]);

% Ομαλοποίηση των χαρακτηριστικών ως μηδενικός μέσος
όρος, διακύμανση μονάδας.
X = bsxfun(@minus, X, mean(X));
X = bsxfun(@rdivide,X,std(X));

%{
Οπτικοποιούμε το σύνολο δυνατοτήτων. Για να κατανοήσουμε
πώς μοιάζουν τα χαρακτηριστικά μεγέθους Gabor, μπορεί να
χρησιμοποιηθεί η Ανάλυση Κύριων Συστατικών για μετακίνηση
από μια αναπαράσταση 26-D κάθε εικονοστοιχείου στην
εικόνα εισόδου σε τιμή έντασης 1-D για κάθε
εικονοστοιχείο.
%}
coeff = pca(X);
feature2DImage = reshape(X*coeff(:,1),numRows,numCols);
figure(5)
imshow(feature2DImage,[])
```



Βήμα 3ο: Ταξινόμηση χαρακτηριστικών υφής Gabor χρησιμοποιώντας kmeans

```
%{
Επαναλαμβάνουμε την ομαδοποίηση k-means πέντε φορές
για να αποφύγουμε τοπικά ελάχιστα κατά την αναζήτηση
μέσων που ελαχιστοποιούν την αντικειμενική
λειτουργία.
Η μόνη προηγούμενη πληροφορία που υποτίθεται σε αυτό
το παράδειγμα είναι πόσες διαφορετικές περιοχές υφής
υπάρχουν στην εικόνα που είναι τμηματοποιημένη.
Υπάρχουν δύο διαφορετικές περιοχές σε αυτήν την
περίπτωση.
%}
L = kmeans(X,2,'Replicates',5);

% Οπτικοποιούμε την τμηματοποίηση χρησιμοποιώντας
label2rgb.
L = reshape(L,[numRows numCols]);
figure(6)
imshow(label2rgb(L))
```

```
%{
Οπτικοποιούμε την τμηματοποιημένη εικόνα
χρησιμοποιώντας imshowpair. Εξετάζουμε τις εικόνες
προσκηνίου και φόντου που προκύπτουν από τη μάσκα BW
που σχετίζεται
με την ετικέτα matrix L.
%}
Aseg1 = zeros(size(img),'like',img);
Aseg2 = zeros(size(img),'like',img);
BW = L == 2;
BW = repmat(BW,[1 1 3]);
Aseg1(BW) = img(BW);
Aseg2(~BW) = img(~BW);
figure(7)
imshowpair(Aseg1,Aseg2,'montage');
```



5) Εκμάθηση Τοπικών Μοντέλων Πρόγνωσης Χρώματος με Χρήση Ταξινομητών SVM

5.1 Εκπαίδεύουμε τους ταξινομητές SVM χρησιμοποιώντας πυρήνα Gaussian

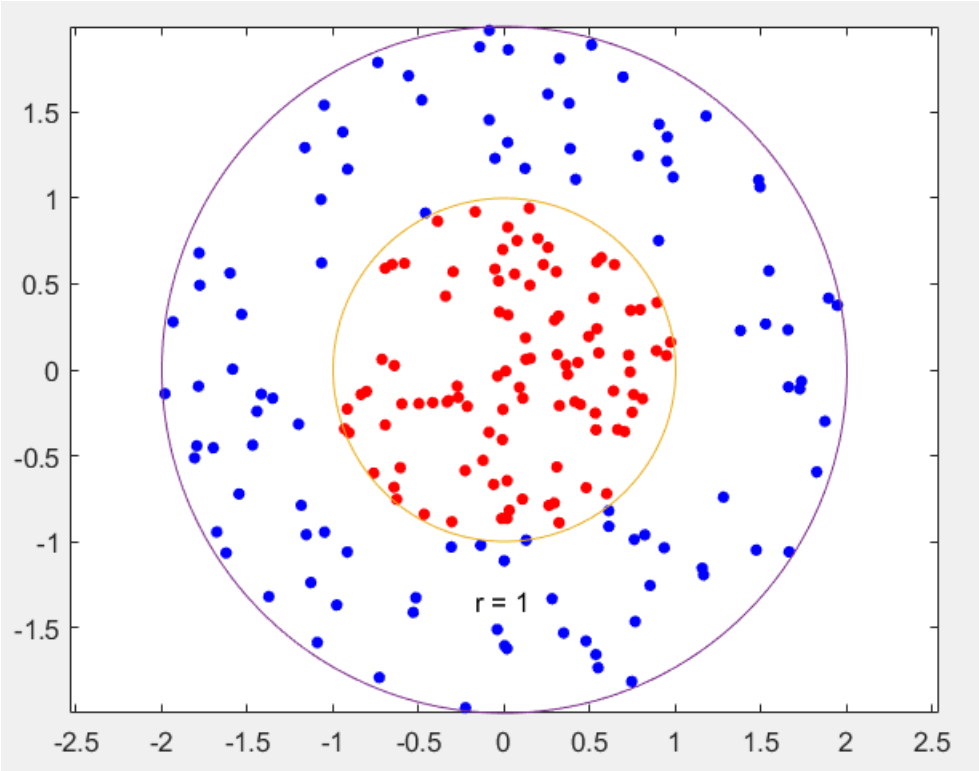
Πηγή:
https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html?search=Train%20SVM%20Classifiers%20Using%20a%20Gaussian%20Kernel&s_tid=srchtitle#bss0s6_-1

Αυτό το παράδειγμα δείχνει πώς να δημιουργήσουμε έναν μη γραμμικό ταξινομητή με τη λειτουργία πυρήνα Gaussian. Πρώτα, δημιουργούμεμια κατηγορία σημείων μέσα στο δίσκο μονάδας σε δύο διαστάσεις και μια άλλη κλάση σημείων στον δακτύλιο από την ακτίνα 1 έως την ακτίνα 2. Στη συνέχεια, δημιουργεί έναν ταξινομητή με βάση τα δεδομένα με τον πυρήνα λειτουργίας ακτινικής βάσης Gaussian. Ο προεπιλεγμένος γραμμικός ταξινομητής είναι προφανώς ακατάλληλος για αυτό το πρόβλημα, καθώς το μοντέλο είναι κυκλικά συμμετρικό. Ορίστε την παράμετρο περιορισμού πλαισίου σε Inf για να κάνουμε μια αυστηρή ταξινόμηση, που σημαίνει ότι δεν υπάρχουν εσφαλμένα ταξινομημένα σημεία. Άλλες λειτουργίες του πυρήνα μπορεί να μην λειτουργεί με αυτόν τον αυστηρό περιορισμό του κουτιού, καθώς ενδέχεται να μην είναι σε θέση να παράσχει αυστηρή ταξινόμηση.
Ακόμα κι αν ο ταξινομητής rbf μπορεί να διαχωρίσει τις κλάσεις, το αποτέλεσμα μπορεί να είναι υπερβολικό.

> Τεκμηρίωση κώδικα(SVM_Classifier_Training_Gaussian_Kernel.m)

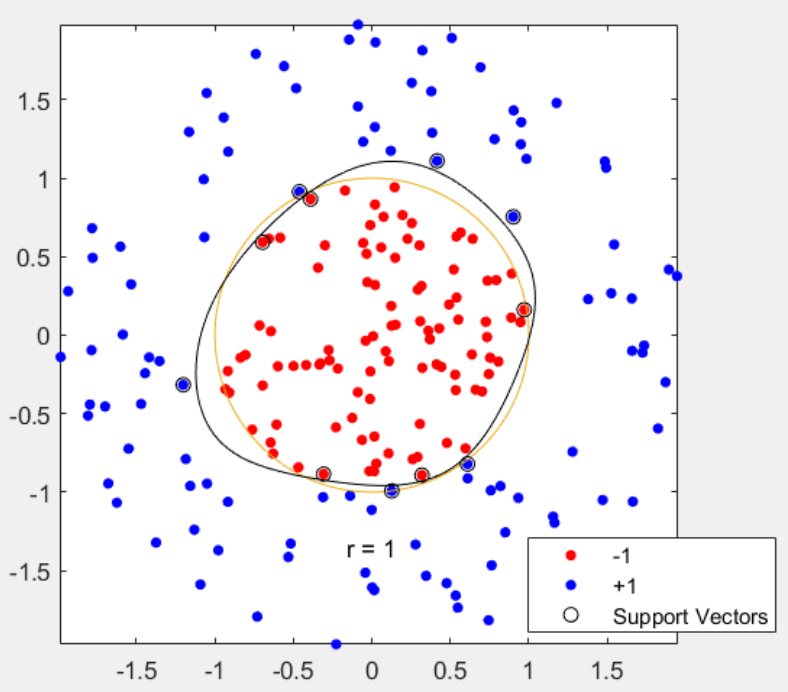
```
%{
Δημιουργούμε 100 πόντους που κατανέμονται ομοιόμορφα στο
δίσκο μονάδας. Για να το κάνουμε αυτό, δημιουργούμε μια
ακτίνα r ως τετραγωνική ρίζα
μιας ομοιόμορφης τυχαίας μεταβλητής, δημιουργούμε μια
γωνία t ομοιόμορφα στο (0,) και τοποθετούμε το σημείο στο
(r cos (t), r sin (t)).
%}
rng(1); % Για αναπαραγωγιμότητα
r = sqrt(rand(100,1)); % Ακτίνα κύκλου
t = 2*pi*rand(100,1);  % Γωνία
data1 = [r.*cos(t), r.*sin(t)]; % Πόντοι
```

```
%{
Δημιούργούμε 100 πόντους ομοιόμορφα κατανεμημένους στο
δακτύλιο.
Η ακτίνα είναι πάλι ανάλογη με μια τετραγωνική ρίζα,
αυτή τη φορά μια τετραγωνική ρίζα της ομοιόμορφης
κατανομής από 1 έως 4.
%}
r2 = sqrt(3*rand(100,1)+1); % Ακτίνα κύκλου
t2 = 2*pi*rand(100,1);      % Γωνία
data2 = [r2.*cos(t2), r2.*sin(t2)]; % Πόντοι

%Σχεδιάζουμε τα σημεία και τους κύκλους των ακτίνων 1 και
2 για σύγκριση
figure;
plot(data1(:,1),data1(:,2),'r.','MarkerSize',15)
hold on
plot(data2(:,1),data2(:,2),'b.','MarkerSize',15)
ezpolar(@(x)1);ezpolar(@(x)2);
axis equal
hold off
```

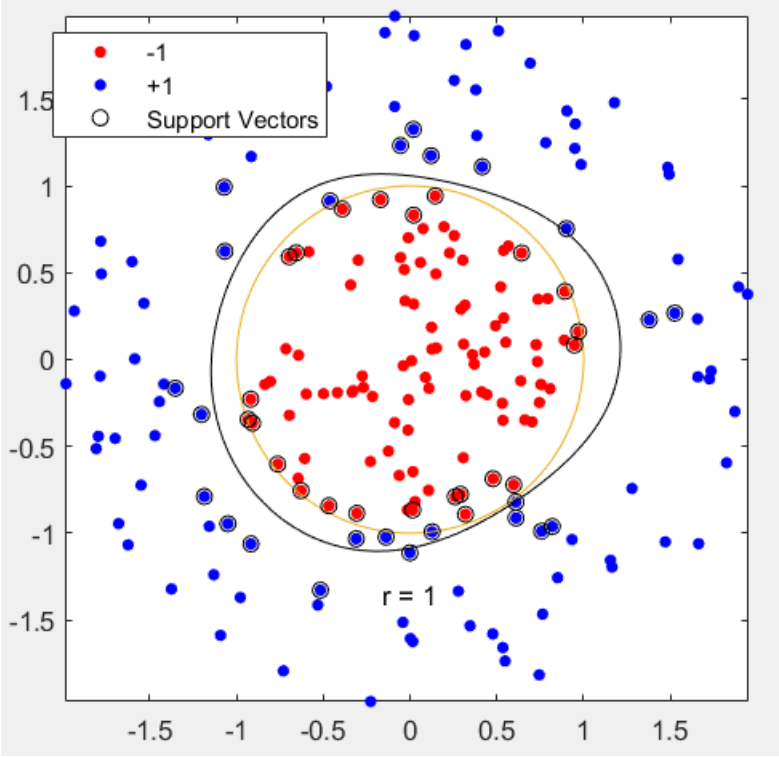



```
%Τοποθέτηούμε τα δεδομένα σε έναν πίνακα και δημιουργήστε
ένα διάνυσμα ταξινομήσεων
data3 = [data1;data2];
theclass = ones(200,1);
theclass(1:100) = -1;
```

```
%{
Εκπαίδεύουμε έναν ταξινομητή SVM με το KernelFunction σε
«rbf» και το BoxConstraint σε Inf.
Σχεδιάζουμε το όριο απόφασης και επισημάνετε τα
διανύσματα υποστήριξης
%}
%Εκπαίδεύουμε τον ταξινομητή SVM
cl = fitcsvm(data3,theclass,'KernelFunction','rbf',...
    'BoxConstraint',Inf,'ClassNames',[-1,1]);

% Προβλέπουμε το σκορ στο πλέγμα
d = 0.02;
[x1Grid,x2Grid] =
meshgrid(min(data3(:,1)):d:max(data3(:,1)),...
    min(data3(:,2)):d:max(data3(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(cl,xGrid);

% Σχεδιάζουμε τα δεδομένα και το όριο της απόφασης
figure;
h(1:2) =
gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) =
plot(data3(cl.IsSupportVector,1),data3(cl.IsSupportVector
,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[
0 0],'k');
legend(h,{'-1','+1','Support Vectors'});
axis equal
hold off
```



```
%Το fitcsvm δημιουργεί έναν ταξινομητή που βρίσκεται
κοντά σε έναν κύκλο ακτίνας 1. Η διαφορά οφείλεται στα
τυχαία δεδομένα εκπαίδευσης.

%{
Η εκπαίδευση με τις προεπιλεγμένες παραμέτρους κάνει ένα
σχεδόν κυκλικό όριο ταξινόμησης,
αλλά ένα που ταξινομεί εσφαλμένα ορισμένα δεδομένα
εκπαίδευσης. Επίσης, η προεπιλεγμένη τιμή του
BoxConstraint είναι 1,
και, επομένως, υπάρχουν περισσότεροι φορείς υποστήριξης.
%}
cl2 = fitcsvm(data3,theclass,'KernelFunction','rbf');
[~,scores2] = predict(cl2,xGrid);

figure;
h(1:2) =
gscatter(data3(:,1),data3(:,2),theclass,'rb','.');
hold on
ezpolar(@(x)1);
h(3) =
plot(data3(cl2.IsSupportVector,1),data3(cl2.IsSupportVect
or,2),'ko');
contour(x1Grid,x2Grid,reshape(scores2(:,2),size(x1Grid)),
[0 0],'k');
legend(h,{'-1','+1','Support Vectors'});
axis equal
hold off
```



5.2 Ανάλυση εικόνων χρησιμοποιώντας μηχανήματα γραμμικής υποστήριξης φορέα

Πηγή:
https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html?search=analyze%20images%20using%20linear%20support%20vector%20machines&s_tid=srchtitle#buoz4f1-1

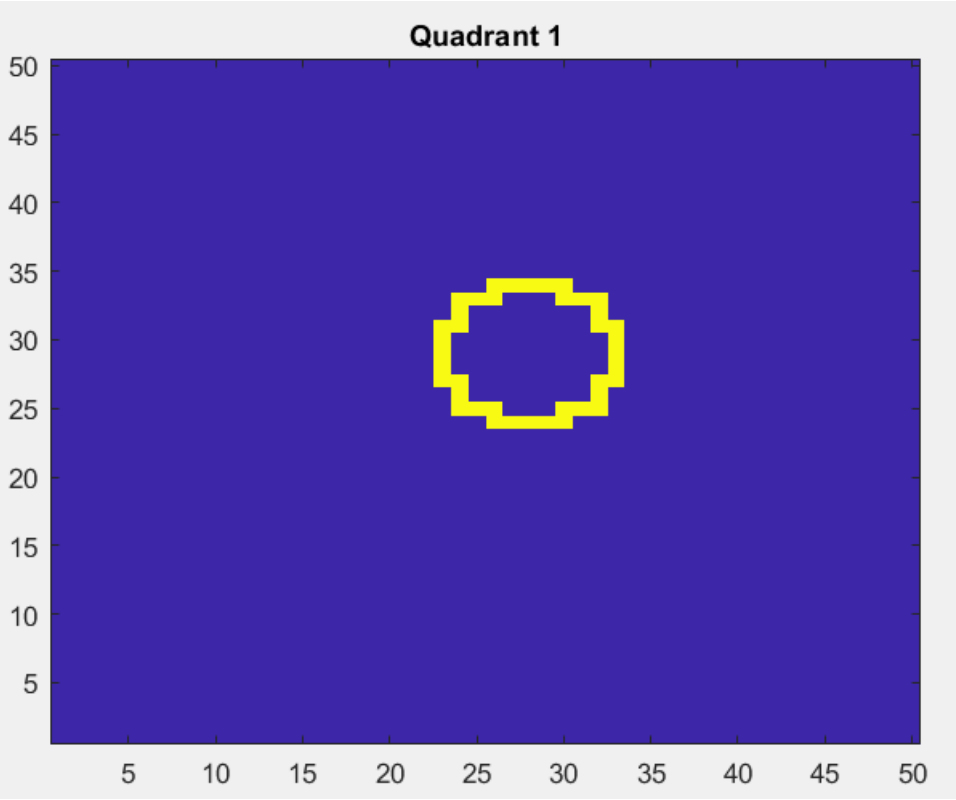
> Τεκμηρίωση κώδικα αρχείου (SVM_Analyzation_Linear_.m):

Αυτό το παράδειγμα δείχνει πώς να προσδιορίσουμε ποιο τεταρτημόριο μιας εικόνας καταλαμβάνει ένα σχήμα εκπαιδεύοντας ένα μοντέλο κωδικών εξόδου διόρθωσης σφαλμάτων (ECOC) που αποτελείται από γραμμικούς δυαδικούς μαθητές SVM.
Αυτό το παράδειγμα απεικονίζει επίσης την κατανάλωση χώρου στο δίσκο των μοντέλων ECOC που αποθηκεύουν διανύσματα υποστήριξης,
τις ετικέτες τους και τους εκτιμώμενους συντελεστές.

Δημιουργία του συνόλου δεδομένων

```
%{
Τυχαία τοποθέτηση ενός κύκλου με ακτίνα πέντε σε μια
εικόνα 50 προς 50. Δημιουργία 5000 εικόνων. Δημιουργία
μιας ετικέτας για κάθε εικόνα που δείχνει το τεταρτημόριο
που καταλαμβάνει ο κύκλος.
Το τεταρτημόριο 1 βρίσκεται επάνω δεξιά, το τεταρτημόριο
2 βρίσκεται επάνω αριστερά, το τεταρτημόριο 3 βρίσκεται
κάτω αριστερά και το τεταρτημόριο 4 είναι κάτω δεξιά.
Οι προβλέψεις είναι οι εντάσεις κάθε εικονοστοιχείου.
%}
d = 50;  % Ύψος και πλάτος των εικόνων σε pixel
n = 5e4; % Το μέγεθος του δείγματος

X = zeros(n,d^2); % Προκατανομή προμετρίας
Y = zeros(n,1);   % Προκατανομή ετικετών
theta = 0:(1/d):(2*pi);
r = 5;            % Ακτίνα κύκλου
rng(1);           % Για αναπαραγωγιμότητα

for j = 1:n;
    figmat = zeros(d);                          % Κενή
εικόνα
    c = datasample((r + 1):(d - r - 1),2);    % Κέντρο
τυχαίων κύκλων
    x = r*cos(theta) + c(1);                    % Κάνουμε
τον κύκλο
    y = r*sin(theta) + c(2);
```

```
    idx = sub2ind([d d],round(y),round(x));  % Μετατροπή
σε γραμμική ευρετηρίαση
    figmat(idx) = 1;                         %
Σχεδιάζουμε τον κύκλο
    X(j,:) = figmat(:);                  % Αποθήκευουμε τα
δεδομένα
    Y(j) = (c(2) >= floor(d/2)) + 2*(c(2) < floor(d/2)) +
...
        (c(1) < floor(d/2)) + ...
        2*((c(1) >= floor(d/2)) & (c(2) < floor(d/2))); %
Προσδιορίζουμε το τεταρτημόριο
end


%Σχεδιάζουμε μια παρατήρηση.
figure;
imagesc(figmat);
h = gca;
h.YDir = 'normal';
title(sprintf('Quadrant %d',Y(end)));
```

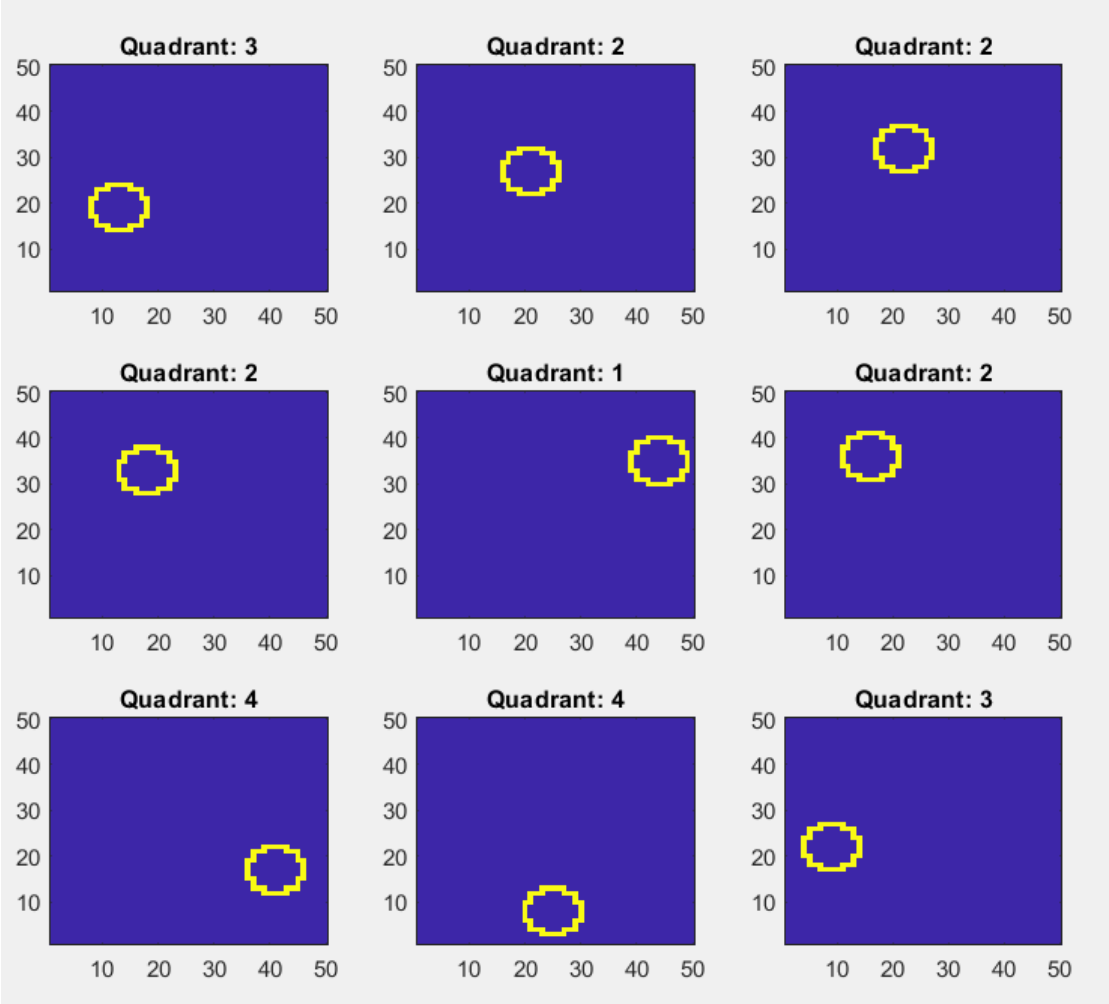


Εκπαιδεύουμε το ECOC μοντέλο

```
%Χρησιμοποιούμε ένα δείγμα αναμονής 25% και καθορίζουμε
τους δείκτες προπόνησης και δείγματος.
p = 0.25;
CVP = cvpartition(Y,'Holdout',p); % Κατάτμηση δεδομένων
πολλαπλής επικύρωσης
```

```
isIdx = training(CVP);              % Δείκτες δείγματος
εκπαίδευσης
oosIdx = test(CVP);                 % Δείκτες δείγματος
δοκιμής
%{
Δημιουργούμε ένα πρότυπο SVM που καθορίζει την αποθήκευση
των διανυσμάτων υποστήριξης των δυαδικών μαθητών.
Διαβάζουμε και τα δεδομένα εκπαίδευσης στο fitcecoc για
να εκπαιδεύσουμε το μοντέλο.
Προσδιορίζουμε το σφάλμα ταξινόμησης δείγματος
εκπαίδευσης.
%}
t = templateSVM('SaveSupportVectors',true);
MdlSV = fitcecoc(X(isIdx,:),Y(isIdx),'Learners',t);
isLoss = resubLoss(MdlSV)

%{
Το MdlSV είναι ένα εκπαιδευμένο μοντέλο
ClassificationECOC multiclass.
Αποθηκεύει τα δεδομένα εκπαίδευσης και τα διανύσματα
υποστήριξης κάθε δυαδικού μαθητή.
Για μεγάλα σύνολα δεδομένων, όπως αυτά στην ανάλυση
εικόνας, το μοντέλο μπορεί να καταναλώνει πολλή μνήμη.
%}
%Προσδιορίζουμε την ποσότητα χώρου στο δίσκο που
καταναλώνει το μοντέλο ECOC.
infoMdlSV = whos('MdlSV');
mbMdlSV = infoMdlSV.bytes/1.049e6

%Βελτιώνουμε την απόδοση του μοντέλου
%{
Μπορούμε να αξιολογήσουμε την απόδοση εκτός δείγματος.
Μπορούμε επίσης να αξιολογήσουμε εάν το μοντέλο ήταν
υπερβολικό με ένα συμπαγές μοντέλο που δεν περιέχει τα
διανύσματα υποστήριξης, τις σχετικές παραμέτρους τους και
τα δεδομένα εκπαίδευσης.
%}

%Απορρίπτουμε τους φορείς υποστήριξης και τις σχετικές
παραμέτρους από το εκπαιδευμένο μοντέλο ECOC. Στη
συνέχεια, απορρίπτουμε τα δεδομένα εκπαίδευσης από το
μοντέλο που προκύπτει χρησιμοποιώντας το compact.
Mdl = discardSupportVectors(MdlSV);
CMdl = compact(Mdl);
info = whos('Mdl','CMdl');
[bytesCMdl,bytesMdl] = info.bytes;
memReduction = 1 - [bytesMdl bytesCMdl]/infoMdlSV.bytes
```

```
%{
Σε αυτήν την περίπτωση, η απόρριψη των διανυσμάτων
υποστήριξης μειώνει την κατανάλωση μνήμης κατά περίπου
3%. Η συμπύκνωση και η απόρριψη διανυσμάτων υποστήριξης
μειώνει το μέγεθος κατά περίπου 99,99%.

Ένας εναλλακτικός τρόπος διαχείρισης διανυσμάτων
υποστήριξης είναι να μειωθεί ο αριθμός τους κατά τη
διάρκεια της προπόνησης, καθορίζοντας έναν μεγαλύτερο
περιορισμό κουτιού, όπως το 100.
Αν και τα μοντέλα SVM που χρησιμοποιούν λιγότερα
διανύσματα υποστήριξης είναι πιο επιθυμητά και
καταναλώνουν λιγότερη μνήμη, αυξάνοντας την τιμή του
περιορισμού κουτιού τείνει να αυξήσει τον χρόνο
προπόνησης.
%}

%Κατάργηση του MdlSV και του Mdl από τον χώρο εργασίας.
clear Mdl MdlSV;

%Υπολογίζουμε το σφάλμα ταξινόμησης του δείγματος
αναμονής.
%Σχεδιάζουμε ένα δείγμα των προβλέψεων αναμονής.
oosLoss = loss(CMdl,X(oosIdx,:),Y(oosIdx))
yHat = predict(CMdl,X(oosIdx,:));
nVec = 1:size(X,1);
oosIdx = nVec(oosIdx);

figure;
for j = 1:9;
    subplot(3,3,j)
    imagesc(reshape(X(oosIdx(j),:),[d d]));
    h = gca;
    h.YDir = 'normal';
    title(sprintf('Quadrant: %d',yHat(j)))
end
text(-1.33*d,4.5*d + 1,'Predictions','FontSize',17)
```




```
%Το μοντέλο δεν κατατάσσει εσφαλμένα παρατηρήσεις
δείγματος αναμονής
```

5.3 Σχεδίαση μεταγενέστερων περιοχών πιθανότητας για μοντέλα ταξινόμησης SVM

Πηγή:
https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html?search=Plot%20Posterior%20Probability%20Regions%20for%20SVM%20Classification%20Models&s_tid=srchtitle#buq4yzy-1

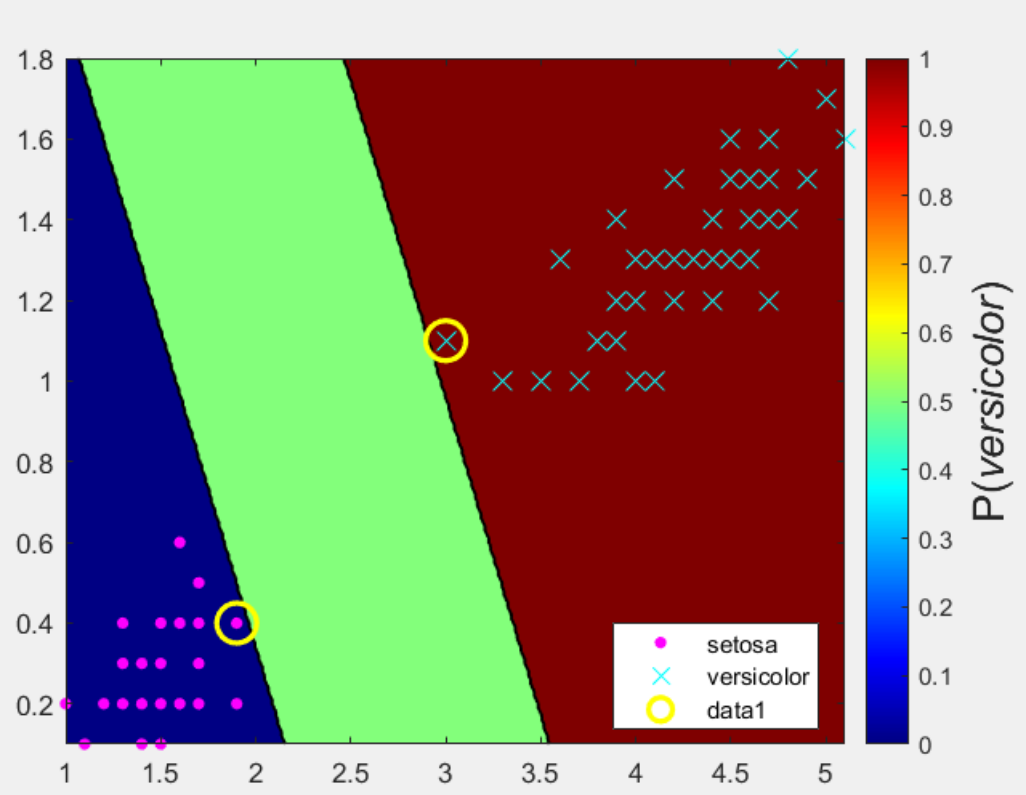
> Τεκμηρίωση κώδικα αρχείου
(SVM_Classification_Models_Plot_Posterior_Probability_Regions.m):

Αυτό το παράδειγμα δείχνει πώς να προβλέψουμε τις οπίσθιες πιθανότητες των μοντέλων SVM σε ένα πλέγμα παρατηρήσεων, και στη συνέχεια σχεδιασμός των οπίσθιων πιθανοτήτων πάνω από το πλέγμα. Η χάραξη οπίσθιας πιθανότητας εκθέτει όρια αποφάσεων.

```
%Φόρτωση του συνόλου δεδομένων ίριδας του Fisher.
Εκπαίδευση του ταξινομητή χρησιμοποιώντας τα μήκη και τα
πλάτη του πέταλου και αφαίρεση των ειδών των παρθενικών
δεδομένων.
load fisheriris
classKeep = ~strcmp(species,'virginica');
X = meas(classKeep,3:4);
y = species(classKeep);

%Εκπαίδευση ενός ταξινομητή SVM χρησιμοποιώντας τα
δεδομένα. Είναι καλή πρακτική να καθορίζετε τη σειρά των
τάξεων.
SVMModel =
fitcsvm(X,y,'ClassNames',{'setosa','versicolor'});

%Υπολογίσμός της βέλτιστης συνάρτησης μετασχηματισμού
βαθμολογίας.
rng(1); %Για αναπαραγωγιμότητα
[SVMModel,ScoreParameters] = fitPosterior(SVMModel);

%Προειδοποίηση: Οι τάξεις διαχωρίζονται τέλεια. Ο
βέλτιστος μετασχηματισμός από οπίσθιο σε οπίσθιο επίπεδο
είναι μια συνάρτηση βημάτων.
ScoreParameters;
```

```
%{
Η βέλτιστη συνάρτηση μετασχηματισμού βαθμολογίας είναι η
συνάρτηση βήματος επειδή οι τάξεις είναι διαχωρίσιμες. Τα
πεδία LowerBound και UpperBound of Score υποδεικνύουν το
κατώτερο και το ανώτερο τελικό σημείο του διαστήματος των
βαθμολογιών που αντιστοιχούν στις παρατηρήσεις εντός των
υπερπλάνων διαχωρισμού τάξης (το περιθώριο). Καμία
παρατήρηση κατάρτισης δεν εμπίπτει στο περιθώριο. Εάν
υπάρχει νέα βαθμολογία στο διάστημα,
τότε το λογισμικό εκχωρεί στην αντίστοιχη παρατήρηση μια
θετική τάξη οπίσθια πιθανότητα, δηλαδή την τιμή στο πεδίο
PositiveClassProbability των ScoreParameters.

Ορίζουμε ένα πλέγμα τιμών στον παρατηρούμενο χώρο
πρόβλεψης. Πρόβλεψη των οπίσθιων πιθανοτήτων για κάθε
παρουσία στο πλέγμα.
%}
xMax = max(X);
xMin = min(X);
d = 0.01;
[x1Grid,x2Grid] =
meshgrid(xMin(1):d:xMax(1),xMin(2):d:xMax(2));

[~,PosteriorRegion] =
predict(SVMModel,[x1Grid(:),x2Grid(:)]);

%Σχεδίαση της περιοχής θετικής τάξης, της οπίσθιας
πιθανότητας και των δεδομένων εκπαίδευσης
figure;
contourf(x1Grid,x2Grid,...

reshape(PosteriorRegion(:,2),size(x1Grid,1),size(x1Grid,2
)));
h = colorbar;
h.Label.String = 'P({\it{versicolor}})';
h.YLabel.FontSize = 16;
caxis([0 1]);
colormap jet;

hold on
gscatter(X(:,1),X(:,2),y,'mc','.x',[15,10]);
sv = X(SVMModel.IsSupportVector,:);
plot(sv(:,1),sv(:,2),'yo','MarkerSize',15,'LineWidth',2);
axis tight
hold off

%{
Σε εκμάθηση δύο τάξεων, εάν οι τάξεις είναι διαχωρίσιμες,
τότε υπάρχουν τρεις περιοχές: μία όπου οι παρατηρήσεις
έχουν θετική τάξη οπίσθια πιθανότητα 0, μία όπου είναι 1,
και μια όπου είναι η θετική τάξη προηγούμενη πιθανότητα
```

%}

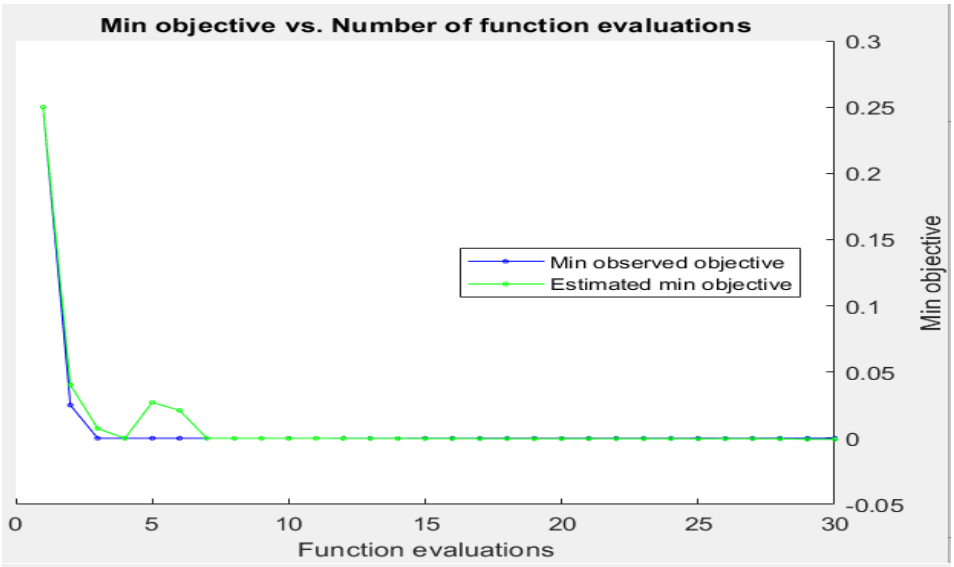

1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
1
1.5
2
2.5
3
3.5
4
4.5
5
setosa
versicolor
data1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
P(versicolor)

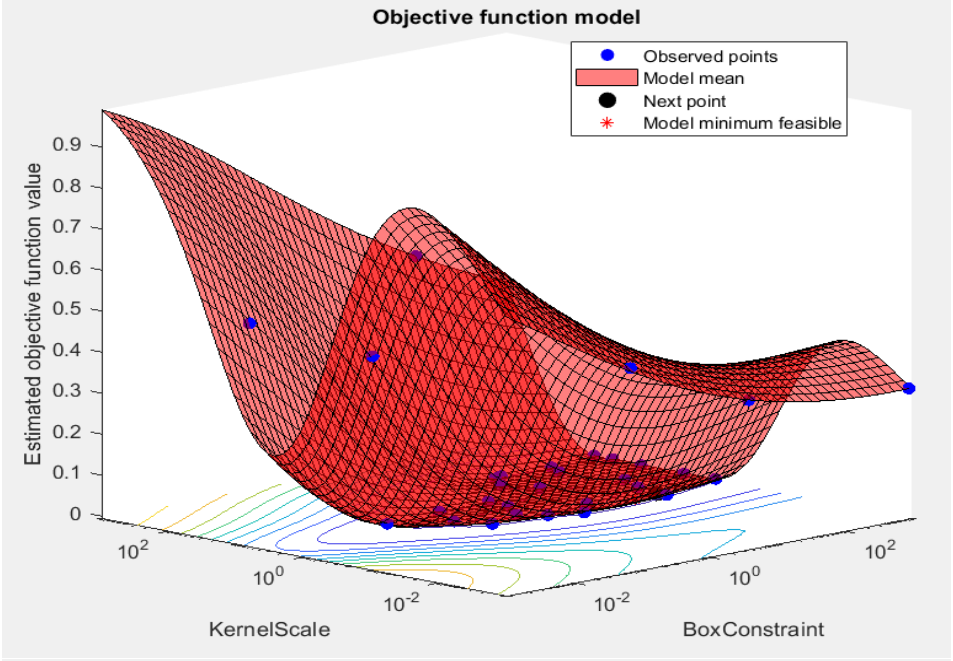
5.4 Ανάλυση του SVM Classifier

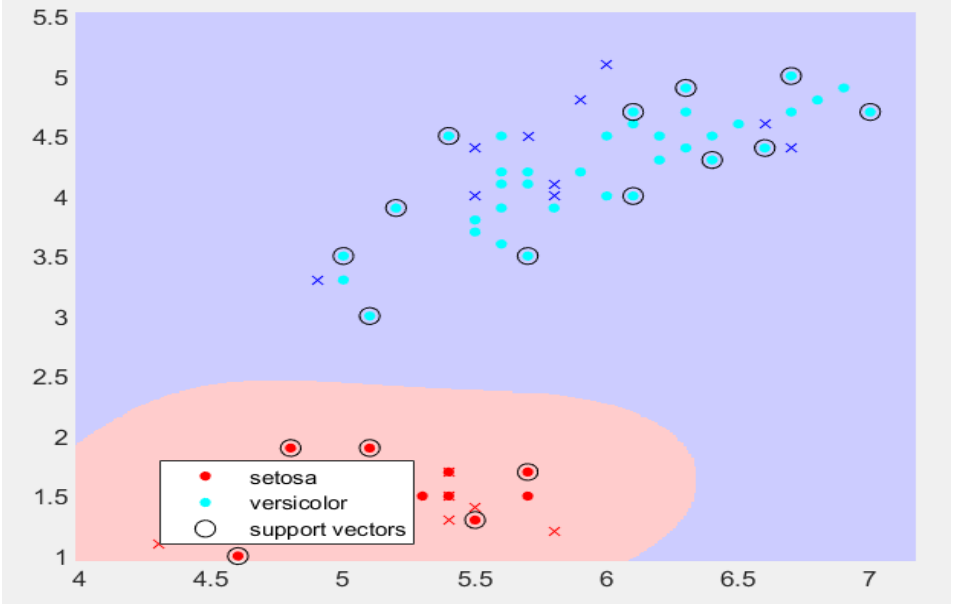
> Τεκμηρίωση κώδικα αρχείου(svm_classifier.m):

```
%% προετοιμασία συνόλου δεδομένων

load fisheriris

species_num = grp2idx(species);
%%

% δυαδική ταξινόμηση
X = randn(100,10);
X(:,[1,3,5,7]) = meas(1:100,:); % 1, 3, 5, 7
y = species_num(1:100);

rand_num = randperm(size(X,1));
X_train = X(rand_num(1:round(0.8*length(rand_num))),:);
y_train = y(rand_num(1:round(0.8*length(rand_num))),:);

X_test =
X(rand_num(round(0.8*length(rand_num))+1:end),:);
y_test =
y(rand_num(round(0.8*length(rand_num))+1:end),:);
%%διαχωρισμός CV

c = cvpartition(y_train,'k',5);
%% επιλογή χαρακτηριστικών

opts = statset('display','iter');
classf = @(train_data, train_labels, test_data,
test_labels)...
    sum(predict(fitcsvm(train_data,
train_labels,'KernelFunction','rbf'), test_data) ~=
test_labels);

[fs, history] = sequentialfs(classf, X_train, y_train,
'cv', c, 'options', opts,'nfeatures',2);
%% Η καλύτερη υπερπαραμέτρος

X_train_w_best_feature = X_train(:,fs);

Md1 =
fitcsvm(X_train_w_best_feature,y_train,'KernelFunction','
rbf','OptimizeHyperparameters','auto',...
'HyperparameterOptimizationOptions',struct('AcquisitionFu
nctionName',...
'expected-improvement-plus','ShowPlots',true)); % Bayes'
Optimization
```

```
%% Τελική δοκιμή
X_test_w_best_feature = X_test(:,fs);
test_accuracy_for_iter =
sum((predict(Md1,X_test_w_best_feature) ==
y_test))/length(y_test)*100

%% υπερπλάνο

figure;
hgscatter =
gscatter(X_train_w_best_feature(:,1),X_train_w_best_featu
re(:,2),y_train);
hold on;
h_sv=plot(Md1.SupportVectors(:,1),Md1.SupportVectors(:,2)
,'ko','markersize',8);


% δεδομένα συνόλου δοκιμής

gscatter(X_test_w_best_feature(:,1),X_test_w_best_feature
(:,2),y_test,'rb','xx')

% επίπεδο απόφασης
XLIMs = get(gca,'xlim');
YLIMs = get(gca,'ylim');
[xi,yi] =
meshgrid([XLIMs(1):0.01:XLIMs(2)],[YLIMs(1):0.01:YLIMs(2)
]);
dd = [xi(:), yi(:)];
pred_mesh = predict(Md1, dd);
redcolor = [1, 0.8, 0.8];
bluecolor = [0.8, 0.8, 1];
pos = find(pred_mesh == 1);
h1 = plot(dd(pos,1),
dd(pos,2),'s','color',redcolor,'Markersize',5,'MarkerEdge
Color',redcolor,'MarkerFaceColor',redcolor);
pos = find(pred_mesh == 2);
h2 = plot(dd(pos,1),
dd(pos,2),'s','color',bluecolor,'Markersize',5,'MarkerEdg
eColor',bluecolor,'MarkerFaceColor',bluecolor);
uistack(h1,'bottom');
uistack(h2,'bottom');
legend([hgscatter;h_sv],{'setosa','versicolor','support
vectors'})
```


Min objective vs. Number of function evaluations
Min observed objective
Estimated min objective
Min objective
Function evaluations
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
5
10
15
20
25
30


Objective function model
Observed points
Model mean
Next point
Model minimum feasible
Estimated objective function value
KernelScale
BoxConstraint
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10^2
10^0
10^{-2}


setosa
versicolor
support vectors
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
4
4.5
5
5.5
6
6.5
7

5.5 Βελτιστοποιούμε μια προσαρμογή ταξινόμησης SVM χρησιμοποιώντας τη βελτιστοποίηση Bayesian

Πηγή:
https://www.mathworks.com/help/stats/optimize-an-svm-classifier-fit-using-bayesian-optimization.html

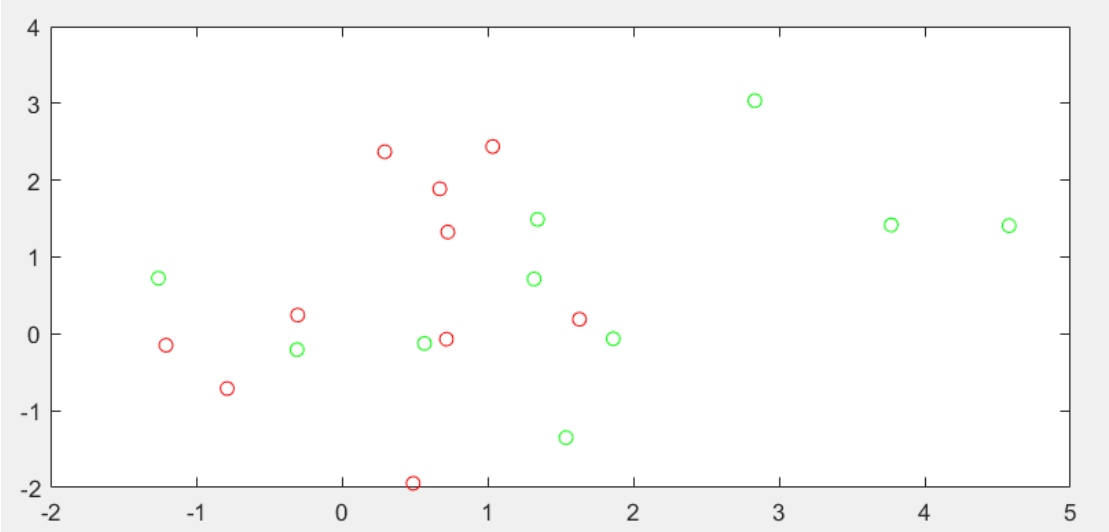
> Τεκμηρίωση κώδικα αρχείου(SVM_Classifier_Fit_Bayesian_Optimization.m):

Αυτό το παράδειγμα δείχνει πώς να βελτιστοποιήσουμε μια ταξινόμηση SVM χρησιμοποιώντας τη συνάρτηση fitcsvm και το ζεύγος τιμών-τιμών OptimizeHyperparameters.
Η ταξινόμηση λειτουργεί σε τοποθεσίες σημείων από ένα μοντέλο μείγματος Gauss.
Στο The Elements of Statistic Learning, Hastie, Tibshirani, and Friedman (2009), η σελίδα 17 περιγράφει το μοντέλο.
Το μοντέλο ξεκινά με τη δημιουργία 10 σημείων βάσης για μια "πράσινη" τάξη, διανεμημένη ως ανεξάρτητη κανονική 2-D με μέση τιμή (1,0)
και διακύμανση μονάδας. Παράγει επίσης 10 σημεία βάσης για μια "κόκκινη" τάξη, διανεμημένη ως ανεξάρτητη 2-D κανονική με μέση τιμή (0,1)
και διακύμανση μονάδας. Για κάθε τάξη (πράσινο και κόκκινο), δημιουργούμε 100 τυχαία σημεία ως εξής:

-Επιλέγουμε ένα σημείο βάσης m του κατάλληλου χρώματος ομοιόμορφα τυχαία.

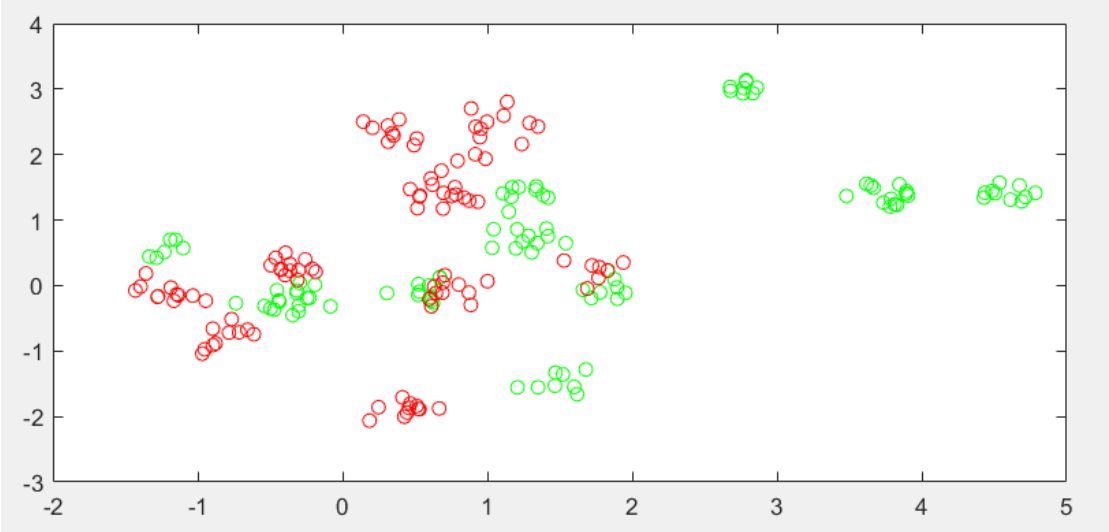
- Δημιουργία ενός ανεξάρτητου τυχαίου σημείου με κανονική κατανομή 2-D με μέση τιμή m και διακύμανση I / 5, όπου I είναι ο πίνακας ταυτότητας 2 προς 2.
Σε αυτό το παράδειγμα, χρησιμοποιούμε μια διακύμανση I / 50 για να δείξουμε το πλεονέκτημα της βελτιστοποίησης με μεγαλύτερη σαφήνεια.

```
%Δημιούργούμε τους πόντους και τον ταξινομητή
%Δημιούργούμε τα 10 σημεία βάσης για κάθε τάξη

rng default % Για αναπαραγωγιμότητα
grnpop = mvnrnd([1,0],eye(2),10);
redpop = mvnrnd([0,1],eye(2),10);

%Δείτε τα σημεία βάσης
plot(grnpop(:,1),grnpop(:,2),'go')
hold on
plot(redpop(:,1),redpop(:,2),'ro')
hold off
```



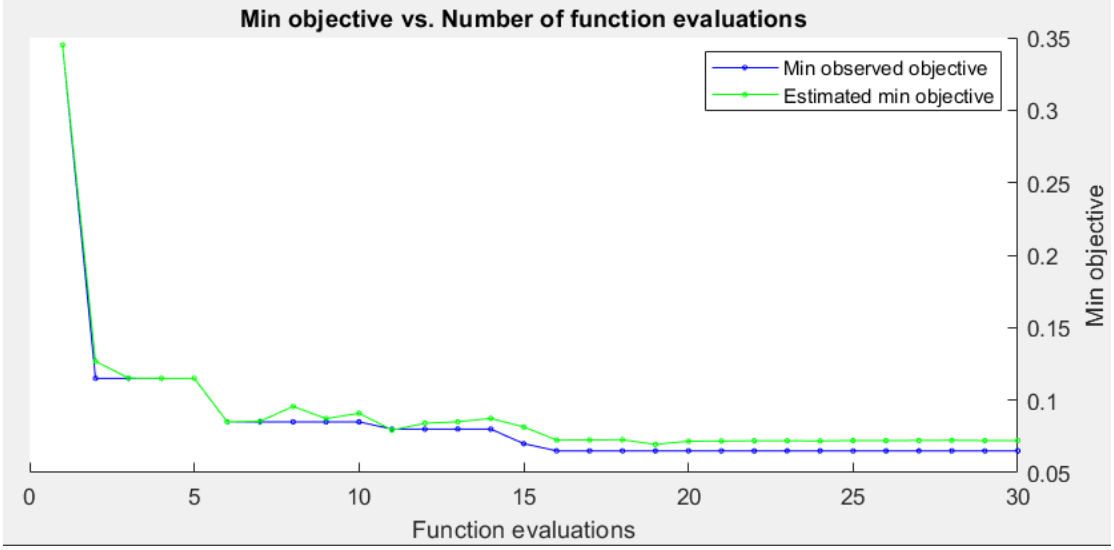
Δεδομένου ότι ορισμένα κόκκινα σημεία βάσης είναι κοντά στα πράσινα σημεία βάσης, μπορεί να είναι δύσκολο να ταξινομηθούν τα σημεία δεδομένων μόνο με βάση την τοποθεσία.

```
%Δημιούργούμε τα 100 σημεία δεδομένων κάθε τάξης.
redpts = zeros(100,2);grnpts = redpts;
for i = 1:100
    grnpts(i,:) =
mvnrnd(grnpop(randi(10),:),eye(2)*0.02);
    redpts(i,:) =
mvnrnd(redpop(randi(10),:),eye(2)*0.02);
end

%Δείχνουμε τα σημεία δεδομένων.
figure
plot(grnpts(:,1),grnpts(:,2),'go')
hold on
plot(redpts(:,1),redpts(:,2),'ro')
hold off
```



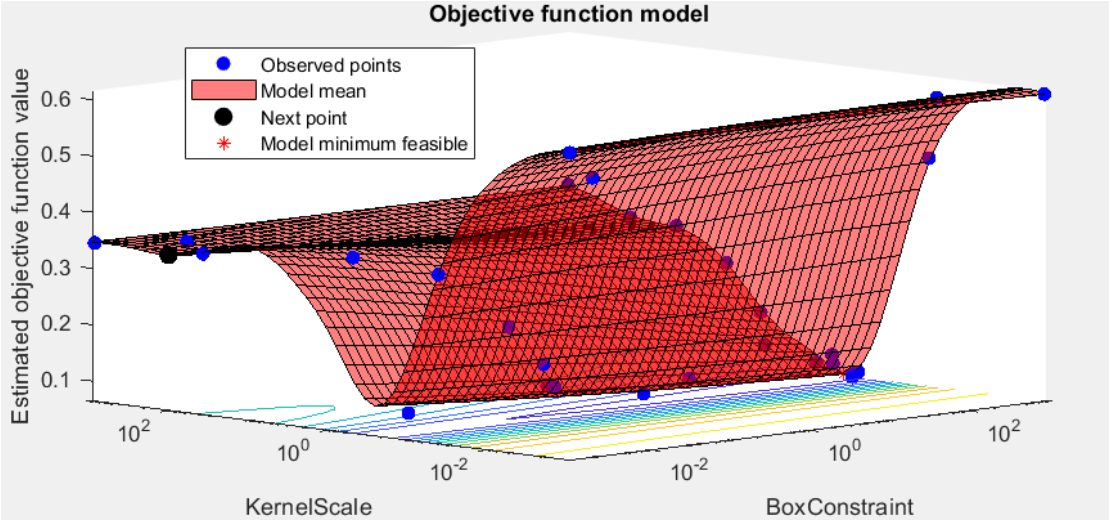
Προετοιμασία δεδομένων για ταξινόμηση

```
%Τοποθέτούμε τα δεδομένα σε μία μήτρα και δημιουργήστε
μια διανυσματική ομάδα που επισημαίνει την κλάση κάθε
σημείου.
cdata = [grnpts;redpts];
grp = ones(200,1);
% Πράσινη ετικέτα 1, κόκκινη ετικέτα -1
grp(101:200) = -1;

%Προετοιμασία διασταυρούμενης επικύρωσης
%Ρυθμίζουμε ένα διαμέρισμα για επικύρωση. Αυτό το βήμα
διορθώνει το τρένο και τα σετ δοκιμών που χρησιμοποιεί η
βελτιστοποίηση σε κάθε βήμα.

c = cvpartition(200,'KFold',10);
```



Βελτιστοποιούμε το Fit

```
%{
Για να βρούμε μια καλή εφαρμογή, που σημαίνει μια με
χαμηλή απώλεια εγκυρότητας, ορίζουμε επιλογές για τη
χρήση βελτιστοποίησης Bayesian.
Χρησιμοποιούμε το ίδιο διαμέρισμα διασταυρούμενης
επικύρωσης c σε όλες τις βελτιστοποιήσεις.
%}

%Για αναπαραγωγιμότητα, χρησιμοποίησε τη λειτουργία
απόκτησης «αναμενόμενη βελτίωση-συν».
opts =
struct('Optimizer','bayesopt','ShowPlots',true,'CVPartiti
on',c,...
    'AcquisitionFunctionName','expected-improvement-
plus');
svmmod = fitcsvm(cdata,grp,'KernelFunction','rbf',...

'OptimizeHyperparameters','auto','HyperparameterOptimizat
ionOptions',opts)

%Βρίσκουμε την απώλεια του βελτιστοποιημένου μοντέλου.
lossnew =
kfoldLoss(fitcsvm(cdata,grp,'CVPartition',c,'KernelFuncti
on','rbf',...

'BoxConstraint',svmmod.HyperparameterOptimizationResults.
XAtMinObjective.BoxConstraint,...

'KernelScale',svmmod.HyperparameterOptimizationResults.XA
tMinObjective.KernelScale))
```



```
%Οπτικοποιούμε τον βελτιστοποιημένο ταξινομητή.
d = 0.02;
[x1Grid,x2Grid] =
meshgrid(min(cdata(:,1)):d:max(cdata(:,1)),...
    min(cdata(:,2)):d:max(cdata(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(svmmod,xGrid);
figure;
h = nan(3,1); % Preallocation
h(1:2) = gscatter(cdata(:,1),cdata(:,2),grp,'rg','+*');
hold on
h(3) = plot(cdata(svmmod.IsSupportVector,1),...
    cdata(svmmod.IsSupportVector,2),'ko');
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[
0 0],'k');
legend(h,{'-1','+1','Support
Vectors'},'Location','Southeast');
axis equal
hold off
```