

La traducción de esta página aún no se ha actualizado a la versión más reciente. Haga clic aquí para ver la última versión en inglés.

Segmentación por colores utilizando el espacio de color $L^*a^*b^*$

Este ejemplo muestra cómo identificar diferentes colores en una tela analizando el espacio de color $L^*a^*b^*$.

Paso 1: Adquirir una imagen

Lea la imagen `fabric.png`, que es una imagen de tela de colores.

Probar este ejemplo

 Copy Command

```
fabric = imread("fabric.png");  
imshow(fabric)  
title("Fabric")
```

 Get ▼

Fabric



Paso 2: Calcular colores de muestra en el espacio de color $L^*a^*b^*$ de cada región

En la imagen se ven seis colores principales: el color de fondo, rojo, verde, morado, amarillo y magenta. Observe con qué facilidad puede distinguir visualmente estos colores entre sí. El espacio de color $L^*a^*b^*$ (también conocido como CIELAB o CIE $L^*a^*b^*$) permite cuantificar estas diferencias visuales.

El espacio de color $L^*a^*b^*$ se deriva de los valores triestímulos CIE XYZ. El espacio $L^*a^*b^*$ consta de una capa de luminosidad ' L^* ' o brillo, una capa de cromaticidad ' a^* ' que indica dónde se sitúa el color en el eje rojo-verde, y una capa de cromaticidad ' b^* ' que indica dónde se sitúa el color en el eje azul-amarillo.

Su enfoque consiste en elegir una pequeña región de muestra para cada color y calcular el color medio de cada región de muestra en el espacio ' a^*b^* '. Utilizará estos marcadores de color para clasificar cada píxel.

A fin de simplificar este ejemplo, cargue las coordenadas de la región almacenadas en un archivo MAT.

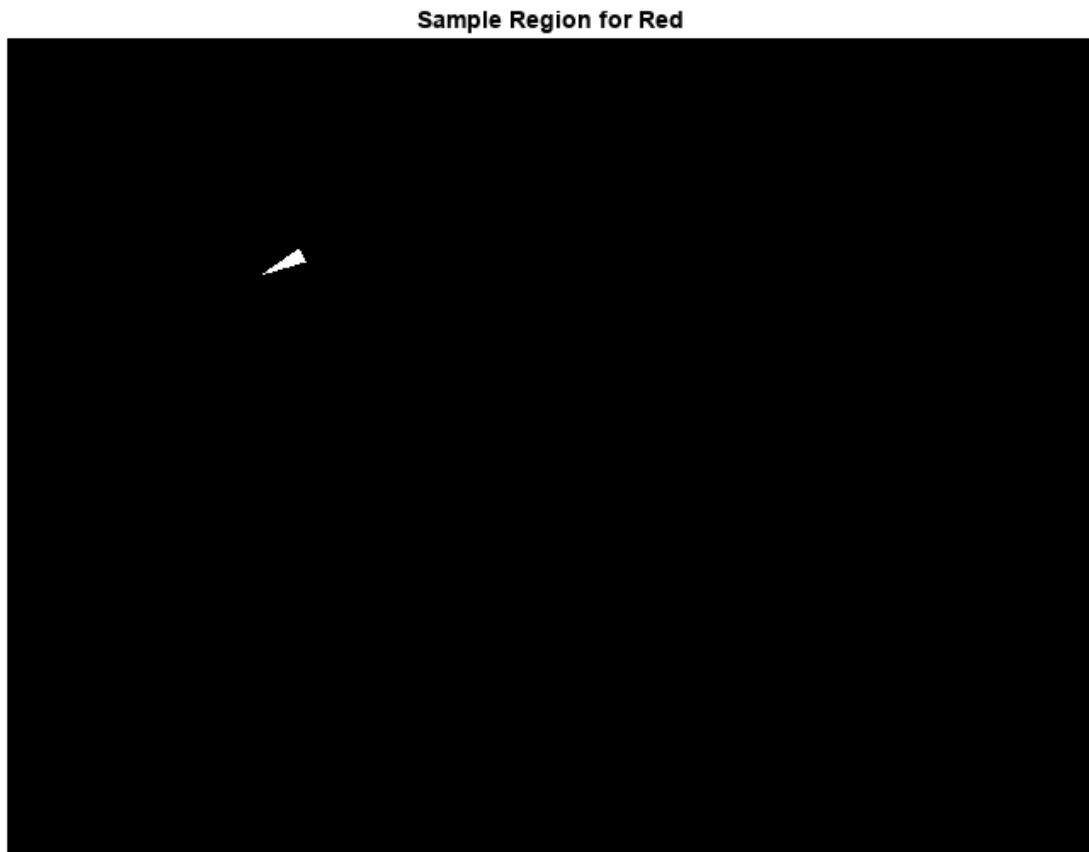
```
load regioncoordinates;

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

for count = 1:nColors
    sample_regions(:,:,count) = roipoly(fabric,region_coordinates(:,1,count), ...
        region_coordinates(:,2,count));
end

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

Get ▼



Convierta su imagen RGB de la tela en una imagen L*a*b* utilizando la función `rgb2lab`.

```
lab_fabric = rgb2lab(fabric);
```

Get ▼

Calcule el valor medio de 'a*' y 'b*' de cada zona que haya extraído con `roipoly`. Estos valores sirven como marcadores de color en el espacio 'a*b*'.


```
a = lab_fabric(:,:,2);
b = lab_fabric(:,:,3);
color_markers = zeros([nColors, 2]);

for count = 1:nColors
    color_markers(count,1) = mean2(a(sample_regions(:,:,count)));
    color_markers(count,2) = mean2(b(sample_regions(:,:,count)));
end
```

Get ▼

Por ejemplo, el color medio de la región roja de la muestra en el espacio a*b* es

```
disp([color_markers(2,1), color_markers(2,2)]);
```

 Get ▼

69.8278 20.1056

Paso 3: Clasificar cada píxel utilizando la regla del vecino más próximo

Cada marcador de color tiene ahora un valor a^* y un valor b^* . Puede clasificar cada píxel de la imagen `lab_fabric` calculando la distancia euclidiana entre ese píxel y cada marcador de color. La distancia más pequeña le indicará que el píxel se aproxima más a ese marcador de color. Por ejemplo, si la distancia entre un píxel y el marcador de color rojo es la menor, el píxel se etiquetaría como píxel rojo.

Cree un arreglo que contenga sus etiquetas de color: 0 = fondo, 1 = rojo, 2 = verde, 3 = púrpura, 4 = magenta y 5 = amarillo.

```
color_labels = 0:nColors-1;
```

 Get ▼

Inicialice las matrices que se utilizarán en la clasificación del vecino más próximo.

```
a = double(a);  
b = double(b);  
distance = zeros([size(a), nColors]);
```

 Get ▼

Realice la clasificación

```
for count = 1:nColors  
    distance(:, :, count) = ( (a - color_markers(count,1)).^2 + ...  
        (b - color_markers(count,2)).^2 ).^0.5;  
end  
  
[~,label] = min(distance,[],3);  
label = color_labels(label);  
clear distance;
```

 Get ▼

Paso 4: Mostrar los resultados de la clasificación del vecino más próximo

La matriz de etiquetas contiene una etiqueta de color para cada píxel de la imagen de la tela. Use la matriz de etiquetas para separar por colores los objetos de la imagen original de la tela.

```
rgb_label = repmat(label,[1 1 3]);  
segmented_images = zeros([size(fabric), nColors], "uint8");  
  
for count = 1:nColors  
    color = fabric;  
    color(rgb_label ~= color_labels(count)) = 0;  
    segmented_images(:, :, :, count) = color;  
end
```

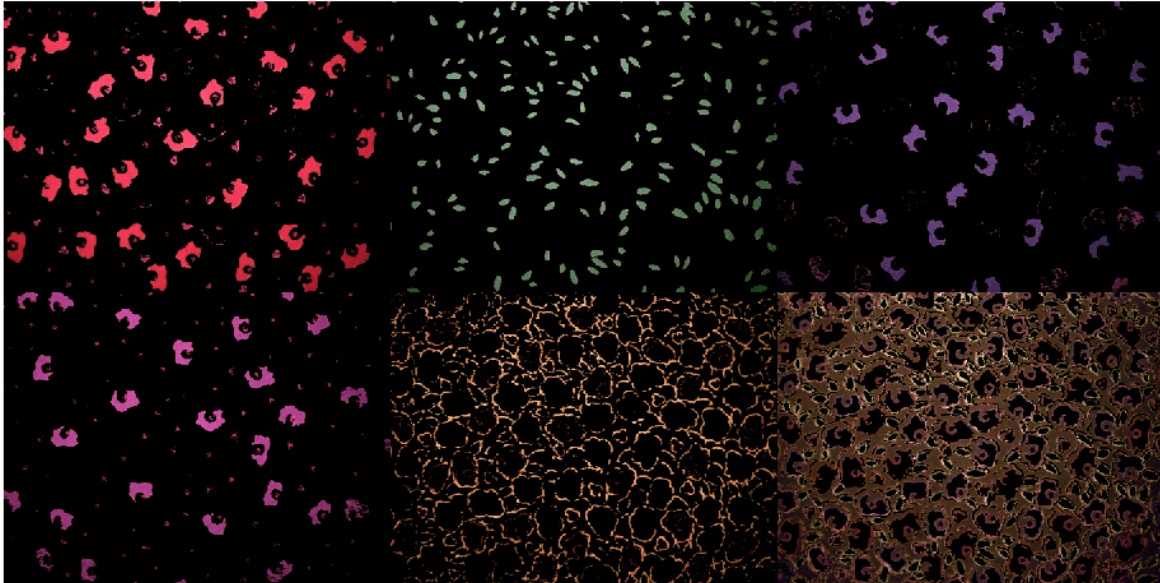
 Get ▼

Muestre los cinco colores segmentados como un montaje. También muestre los píxeles de fondo de la imagen que no están clasificados como un color.

```
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")
```

 Get ▼

Montage of Red, Green, Purple, Magenta, and Yellow Objects, and Background



Paso 5: Mostrar los valores a^* y b^* de los colores etiquetados

Puede ver lo bien que la clasificación del vecino más cercano separó las diferentes poblaciones de color representando los valores a^* y b^* de los píxeles que se clasificaron en colores distintos. Para una mejor visualización, etiquete cada punto con su etiqueta de color. El púrpura no es un valor de color con nombre, así que especifique el color púrpura utilizando una cadena con código de color hexadecimal.

```
purple = "#774998";
plot_labels = ["k", "r", "g", purple, "m", "y"];

figure
for count = 1:nColors
    plot_label = plot_labels(count);
    plot(a(label==count-1),b(label==count-1),".", ...
        MarkerEdgeColor=plot_label,MarkerFaceColor=plot_label);
    hold on
end

title("Scatterplot of Segmented Pixels in  $a^*b^*$  Space");
xlabel("a* Values");
ylabel("b* Values");
```

Get ▼

