




SEGMENTACIÓN POR COLOR

PEDRO JOSÉ DÍAZ GARCÍA

SISTEMAS DE PERCEPCIÓN, 4º GIERM
Ejercicio Práctico 3 (versión básica)



El objetivo de este ejercicio es la realización de un programa en MATLAB capaz de identificar y segmentar los diferentes elementos que aparecen en una imagen en función del color de estos. En este caso, se trabajará con la imagen de la Figura 1, en la que aparecen hasta seis colores distintos. El objetivo final es obtener, para cada uno de esos colores, una imagen en la que aparezcan identificados los elementos del color en cuestión, representando su *bounding box* y su centroide. A continuación, se explica el código desarrollado, que puede encontrarse al final de este documento.



Figura 1. Imagen de trabajo.

El programa realiza una iteración para cada uno de los seis colores presentes; entre iteración e iteración, el usuario deberá pulsar una tecla para que el programa continúe. En primer lugar, se realiza la adquisición de los datos de la imagen con la que se va a trabajar, diferenciando la información de cada uno de los tres canales RGB. Además, utilizando la herramienta **colorThresholder** de MATLAB, se han obtenido unos valores umbrales mínimos y máximos aproximados para cada uno de dichos canales, de manera que, si se discriminan los píxeles de la imagen original en función de esos rangos de valores, la representación resultante únicamente contiene los elementos de un color determinado. Estos valores se han almacenado en la matriz *umbrales*.

Una vez se tienen todos los datos necesarios, se da comienzo al procesamiento de la imagen, que está estructurado en los siguientes pasos.

1. Obtención de la plantilla.

En primer lugar, es necesario calcular una plantilla que, aplicada a la imagen original, permita obtener una imagen en la que solo aparezcan los elementos del color requerido. Para ello se hace uso de los umbrales calculados previamente, creando una matriz de valores lógicos para cada uno de los canales RGB de la imagen original. Según esta matriz, a un píxel le corresponderá un valor 1 si su valor de intensidad en dicho canal de color se encuentra en el rango especificado por los valores umbrales, y un 0 en caso contrario.

```
filtro_rojo=(umbrales(c,1) <= canal_rojo & canal_rojo <= umbrales(c,2));  
filtro_verde=(umbrales(c,3) <= canal_verde & canal_verde <= umbrales(c,4));  
filtro_azul=(umbrales(c,5) <= canal_azul & canal_azul <= umbrales(c,6));
```

Seguidamente, se realiza aplica la función AND a estas tres plantillas, para obtener una plantilla para discriminar únicamente los píxeles cuyas intensidades repeten los tres rangos especificados.

```
plantilla=filtro_rojo & filtro_verde & filtro_azul;
```

Por último, se aplica la plantilla calculada a la imagen original. Previamente es necesario conformar una plantilla para tres canales, ya que la imagen original tiene dimensión 3 (tres canales) y la plantilla que se había obtenido solo dimensión 1.

```
plantilla3=[];  
plantilla3(:,:,1)=plantilla;  
plantilla3(:,:,2)=plantilla;  
plantilla3(:,:,3)=plantilla;  
  
lacasitos=imagen_original.*uint8(plantilla3)
```

En la siguiente imagen se muestra el resultado de la aplicación de la plantilla calculado sobre la imagen original para el caso del color rojo.

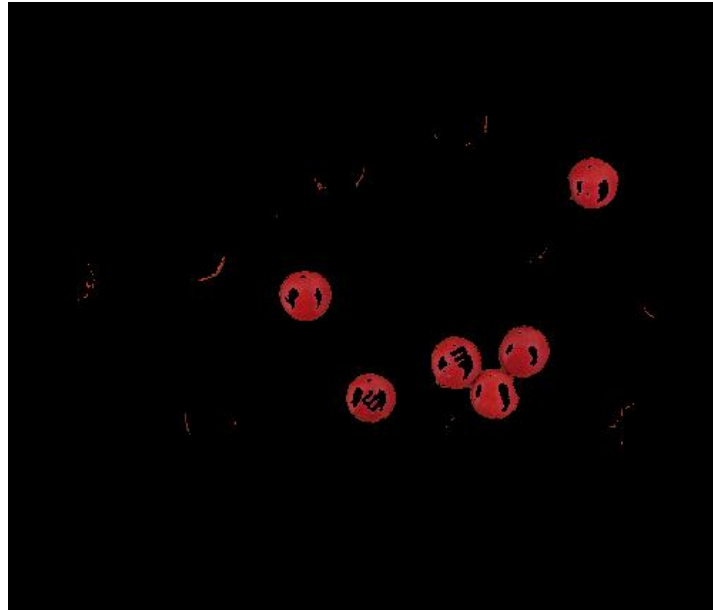


Figura 2. Imagen obtenida tras la aplicación de la plantilla.

2. Acondicionamiento de la imagen.

Como puede comprobarse, si bien en la imagen de la Figura 2 puede apreciarse que se han filtrado bien los elementos de la imagen, aún restan píxeles dispersos que dificultarán la correcta ejecución de un etiquetado posterior de los elementos que aparecen. Por tanto, es necesario un tratamiento previo de la imagen en la que se eliminen dichos píxeles; también conviene eliminar los huecos creados en los elementos a identificar.

Para ello, se llevará a cabo, en primer lugar, una apertura para eliminar los píxeles dispersos, y posteriormente, un cierre para rellenar los huecos de los elementos. Esto queda resuelto en las siguientes líneas de código en las que se hace uso de las funciones **imopen** e **imclose** del *Image Processing Toolbox* de MATLAB, con la previa creación de una máscara para aplicar dichas funciones. El resultado obtenido tras este tratamiento se muestra en la Figura 3.

```
mask=strel('disk',2);  
lacasitos=imopen(lacasitos,mask);  
lacasitos=imclose(lacasitos,mask);
```

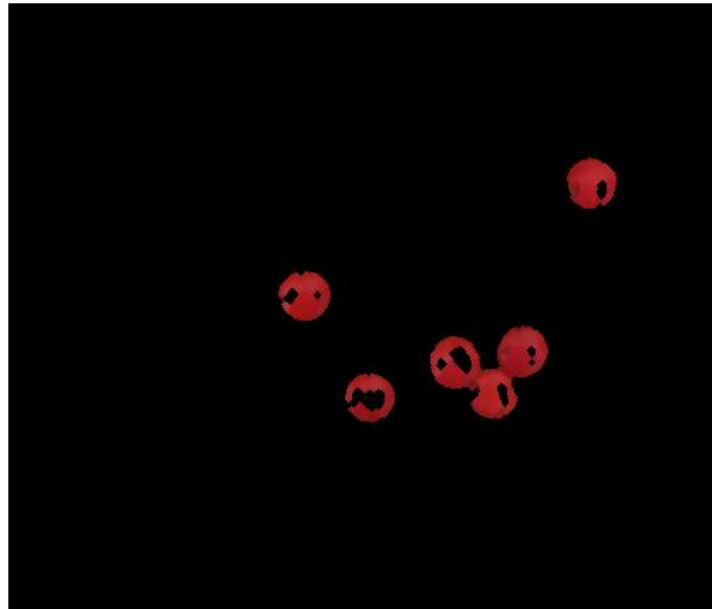


Figura 3. Imagen acondicionada.

3. Etiquetado de los elementos.

En tercer lugar, se realiza un etiquetado, de manera que cada región de píxeles conectados que aparezca en la imagen tendrá asociado un valor como etiqueta; con dicha etiqueta podrá identificarse la región en cuestión. Para ello se hace uso de la función **bwlabel**. Debido a que esta función debe tratar con una imagen binaria, y la imagen obtenida tiene tres canales, se preferido aplicar el etiquetado sobre la plantilla que se aplicó en el paso 1. Obviamente, dicha plantilla también deberá ser previamente sometida al tratamiento descrito en el paso 2. En la figura 4 se representan cada uno de los elementos individualmente.

```
[m_etiq,num_etiq]=bwlabel(255*uint8(imopen(plantilla,mask)) );
```

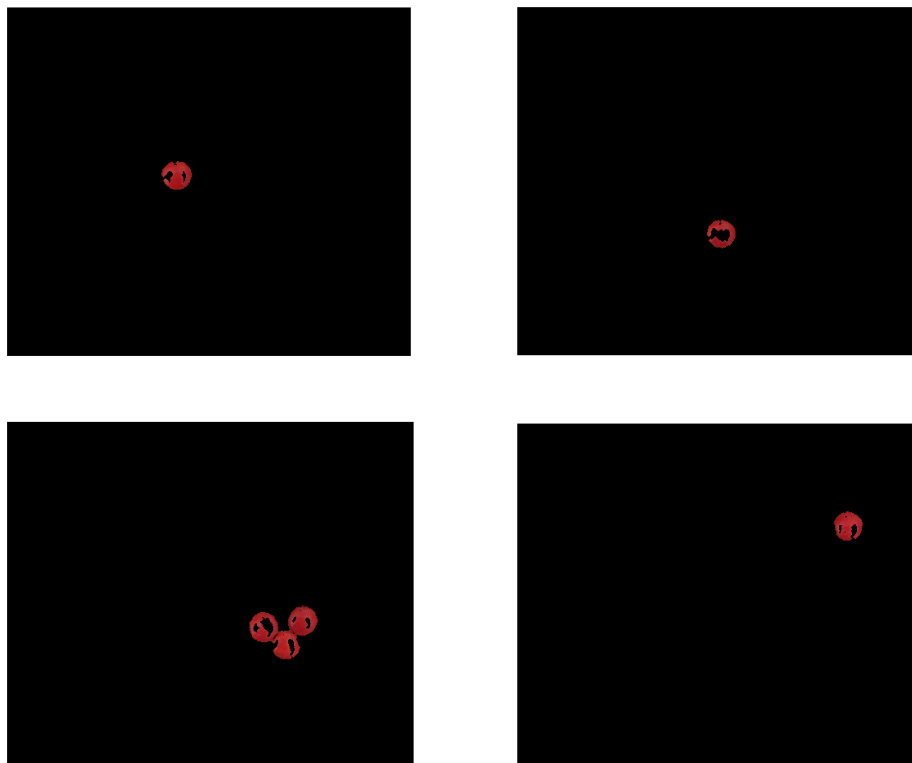


Figura 4. Elementos individualizados.

4. Obtención de propiedades.

Aplicando la función **regionprops** a la matriz de etiquetas obtenida en el paso anterior se obtiene la información pedida a dicha función de cada uno de los elementos etiquetados. En este caso, se requería determinar las dimensiones y posición del *bounding box* de cada elemento, así como la posición de su centroide.

```
propiedades=regionprops(m_etiq, 'Centroid', 'BoundingBox');
```

Una vez se ha obtenido esta información, la consecución del resultado buscado es inmediata sin más que representar estos elementos sobre la imagen original con las siguientes líneas de código. El resultado obtenido para cada uno de los seis colores presentes en la imagen puede verse en la Figura 5.

```
for i=1:num_etiq
    box=propiedades(i).BoundingBox;
    centro=propiedades(i).Centroid;
    centro=round(centro);
    hold on;rectangle('Position',box,'EdgeColor','g','LineWidth',2);
    hold on;plot(centro(1),centro(2),'b.','MarkerSize',20);
end
```

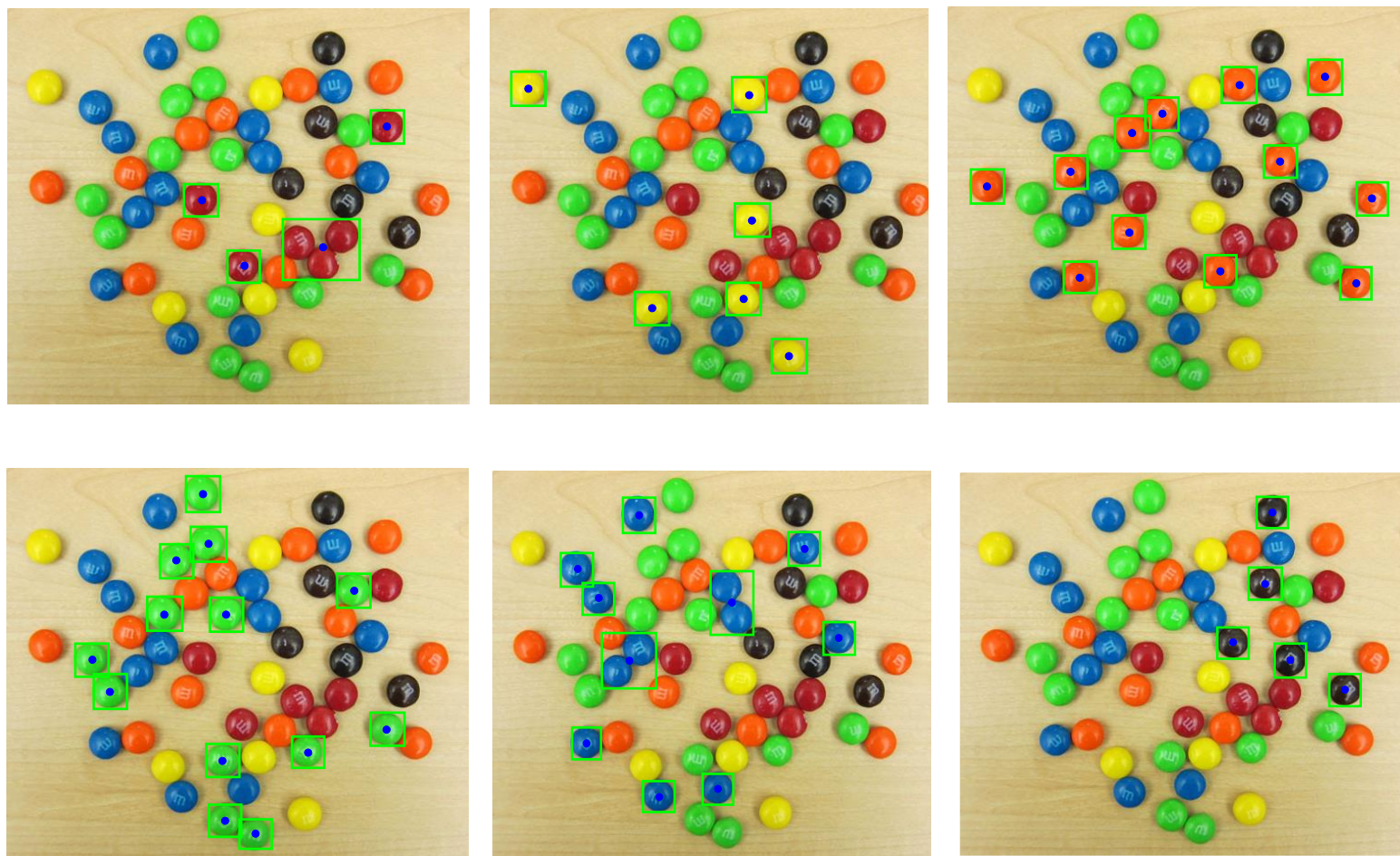


Figura 5. Elementos identificados.

Código completo

```
clear all;close all;clc;

%Obtención de la imagen de trabajo.
imagen_original=imread('imagenDePartida.png');

%Obtención de cada uno de los canales RGB de la imagen.
canal_rojo = imagen_original(:,:,1);
canal_verde = imagen_original(:,:,2);
canal_azul = imagen_original(:,:,3);

%Matriz de valores umbrales obtenidos con la herramienta colorThresholder.
%Orden: {rojo,amarillo, naranja, verde, azul,negro}.
      %Rmin Rmax Gmin Gmax Bmin Bmax
umbrales=[ 90   255   12    64   13   181;
          177   255  156   255    0    64;
          209   255    0   146    0   255;
           0   150  155   255   23   144;
           0    97    0   145  108   255;
          21    91    0    88    0    86];

%Proceso de segmentación para cada uno de los colores.
for c=1:6
% 1. Obtención y aplicación de la plantilla para filtrar uno de los colores.
filtro_rojo=(umbrales(c,1) <= canal_rojo & canal_rojo <= umbrales(c,2));
filtro_verde=(umbrales(c,3) <= canal_verde & canal_verde <= umbrales(c,4));
filtro_azul=(umbrales(c,5) <= canal_azul & canal_azul <= umbrales(c,6));

plantilla=filtro_rojo & filtro_verde & filtro_azul;

plantilla3=[];
plantilla3(:,:,1)=plantilla;
plantilla3(:,:,2)=plantilla;
plantilla3(:,:,3)=plantilla;

lacasitos=imagen_original.*uint8(plantilla3);
% figure(c+1);imshow(lacasitos); %imagen con solo lacasitos de un color

% 2. Acondicionamiento de la imagen: se realiza una apertura para eliminar los
%píxeles dispersos, y un cierre para rellenar los elementos que quedan.
mask=strel('disk',2);
lacasitos=imopen(lacasitos,mask);
% lacasitos=imclose(lacasitos,mask);
% figure(c+2);imshow(lacasitos); %imagen una vez tratada

% 3. Etiquetado de los elementos de la imagen.
[m_etiq,num_etiq]=bwlabeled(255*uint8(imopen(plantilla,mask)));
% figure(c+3); %representa individualmente cada uno de los elementos
% for i=1:num_etiq
%     lacasito_i = imagen_original .* uint8(m_etiq==i);
%     imshow(lacasito_i);
%     pause;
% end

% 4. Obtención de las propiedades de los objetos etiquetados.
propiedades=regionprops(m_etiq,'Centroid','BoundingBox');

%muestra sobre la imagen original el bounding box de cada elemento y su
%centroide.
figure(c+4);imshow(imagen_original);hold on;
for i=1:num_etiq
    box=propiedades(i).BoundingBox;
    centro=propiedades(i).Centroid;
    centro=round(centro);
    hold on;rectangle('Position',box,'EdgeColor','g','LineWidth',2);
    hold on;plot(centro(1),centro(2),'b.','MarkerSize',20);
end
    pause;
end
```