Práctica 1

NOTA: el código fuente completo se puede consultar en los ficheros adjuntos a este documento. Cada apartado tiene su correspondiente método. Lo mismo para las imágenes.

Se ha usado laversión 2.4.3 de OpenCV

Bernardo Miquel Riera

Joan Gabriel Florit Gomila

ÍNDICE

A) Captura (módulo 1)

Objetivos:

- Realizar una captura básica de imágenes a través de la cámara propia o bien sobre bibliotecas disponibles en Internet.
- Si se usan imágenes de Internet indicar la dirección de procedencia.
- Realizar diferentes sistemas de iluminación de la imagen si las imágenes son propias.

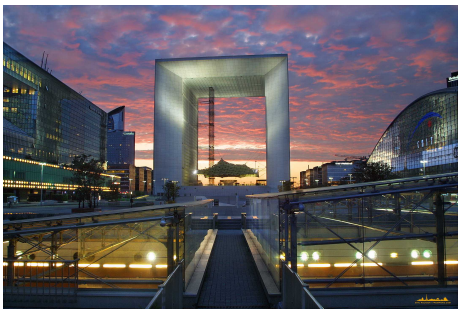
Procesos realizados:

Buscar en Internet imágenes diferentes con iluminaciones dispares. Fuentes:

- fansdeferrari.hautetfort.com ferrari.jpg

- viaje2punto0.com la_defense_dia.jpg

- fromparis.com — la_defense_nit.jpg



- Flickr Àngela Llop — montanya.jpg

- Wikipedia — partenon.jpg

- Viquipèdia — poma.jpg

* Para los dos últimos apartados se usarán imágenes de canicas. También tenis.jpg.

Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.jpg
  - entrada/la_defense_dia.jpg
  - entrada/la_defense_nit.jpg
  - entrada/montanya.jpg
  - entrada/partenon.jpg
  - entrada/poma.jpg

  - entrada/canicas1.jpg
  - entrada/canicas2.jpg
  - entrada/canicas3.jpg
  - entrada/canicas4.jpg
  - entrada/canicas5.jpg

B) Resolución espacial y cuantificación (módulo 2)

Objetivos:

- Modificar la resolución espacial y la cuantificación. Comparar resultados calidad visual versus tamaño fichero.

Procesos realizados:

Las imágenes pueden verse como una matriz de píxeles de n filas y m columnas.

La resolución espacial o muestreo, es el número de píxeles de que dispone la imagen. Es decir: n*m.

```
resize(imgAux, imgAux, Size(), escala, escala, INTER_NEAREST);
```

La resolución en amplitud o cuantificación tiene que ver con el rango de valores con que se representa la intensidad luminosa de los píxeles.

```
imatge.convertTo(imgAux, 8, escala);

imatge.convertTo(imgAux, 8, 1/escala);
```

Para este apartado vamos a elegir una sola imagen (ferrari.png pero convirtiéndola a blanco y negro) puesto que el comportamiento sería el mismo para el resto de ellas. En la siguiente tabla se muestra un resumen de las transformaciones hechas a la imagen y su correspondiente tamaño en disco:

		Resolución espacial			
		1024 x 768	512 x 384	256 x 192	64 x 48
Resolución en amplitud	256 niveles de gris	431 KB (441371 bytes)	125 KB (128046 bytes)	34,8 KB (35669 bytes)	2,62 KB (2693 bytes)
	64 niveles de gris	258 KB (265119 bytes)	79,6 KB (81585 bytes)	23,3 KB (23935 bytes)	1,94 KB (1994 bytes)
	16 niveles de gris	139 KB (142532 bytes)	44,6 KB (45692 bytes)	13,6 KB (14022 bytes)	1,26 KB (1291 bytes)
	2 niveles de gris	36,9 KB (37886 bytes)	13,0 KB (13373 bytes)	4,5 KB (4611 bytes)	0,5 KB (592 bytes)

Observamos como cuanto menor es la resolución espacial, menor es el espacio que ocupa la imagen en disco, ya que la matriz es de un tamaño inferior. También podemos ver como si bajamos la resolución en amplitud, menor es el espacio en disco que ocupa la imagen, puesto que por cada pixel se guarda menos información y el algoritmo de compresión puede hacer un mejor trabajo. Y por último, si se rebajan ambas resoluciones, vemos como lógicamente sigue bajando el tamaño que ocupa la imagen en disco. Así pues, la imagen con una mayor calidad (y mayor tamaño en disco) será la que tiene 256 niveles de gris y un tamaño de 1024x768 píxeles. Y la peor, la imagen con tan sólo 2 niveles de gris y 64x48 píxeles.

Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.png (256 niveles de gris y tamaño 1024x768, en .png)

- Salida:
  - salida/resoluciones/ferrari_1024x768_256.png
  - salida/resoluciones/ferrari_1024x768_64.png
  - salida/resoluciones/ferrari_1024x768_16.png
  - salida/resoluciones/ferrari_1024x768_2.png

  - salida/resoluciones/ferrari_512x384_256.png
  - salida/resoluciones/ferrari_512x384_64.png
  - salida/resoluciones/ferrari_512x384_16.png
  - salida/resoluciones/ferrari_512x384_2.png

  - salida/resoluciones/ferrari_256x192_256.png
  - salida/resoluciones/ferrari_256x192_64.png
  - salida/resoluciones/ferrari_256x192_16.png
  - salida/resoluciones/ferrari_256x192_2.png

  - salida/resoluciones/ferrari_64x48_256.png
  - salida/resoluciones/ferrari_64x48_64.png
  - salida/resoluciones/ferrari_64x48_16.png
  - salida/resoluciones/ferrari_64x48_2.png

C) Color (módulo 2)

Objetivos:

- Sobre una imagen en color de la escena, cambiar de modelo de color RGB a HSL.
- Seleccionar la banda que obtenga más información sobre la intensidad de la imagen en cada modelo.
- Hacer cambios de modelos entre las imágenes.

Procesos realizados:

Para cambiar la imagen entrante, -con un modelo de color RGB-, a un modelo HSL, es suficiente con:

```
cvtColor(imatge_RGB, imatge_HLS, CV_BGR2HLS);
```

Después hemos realizado cambios de modelos entre la imagen original en RGB a los siguientes:

- CIE Lab `cvtColor(imatge_RGB, imatge_CIE, CV_BGR2XYZ);`
- HSV (HSI) `cvtColor(imatge_RGB, imatge_HSV, CV_BGR2HSV);`

Para seleccionar la banda que obtenga más información, primero es necesario separar los canales. Esto lo haremos para los modelos RGB, HSL, CIE y HSV. Los resultados pueden verse en las imágenes de salida (se visualizan en RGB, debido a que es el sistema que usan las pantallas). Seleccionamos la banda que tiene más información por cada modelo en base a lo anterior y lo que sabemos de teoría:

- RGB. Se selecciona la componente de color verde ya que es la que dispone de más información sobre la luminancia.
- HSL. En escala de grises. Nos quedamos con el canal L de luminancia ya que es el que contiene más información.
- CIE XYZ. En escala de grises. Resulta complicado decantarse por un sólo canal, ya que el X e Y son similares a la vista. Más bien el X mejor que el Y.
- HSV (HSI). En escala de grises. El canal V es que aporta más información.

Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.png

- Salida:
  - salida/color/RGB/ferrari_RGB.png
  - salida/color/RGB/ferrari_RGB_R.png
  - salida/color/RGB/ferrari_RGB_G.png
  - salida/color/RGB/ferrari_RGB_B.png

  - salida/color/HSL/ferrari_HSL.png
  - salida/color/HSL/ferrari_HSL_H.png
  - salida/color/HSL/ferrari_HSL_S.png
  - salida/color/HSL/ferrari_HSL_L.png

  - salida/color/CIE/ferrari_CIE.png
  - salida/color/CIE/ferrari_CIE_X.png
  - salida/color/CIE/ferrari_CIE_Y.png
  - salida/color/CIE/ferrari_CIE_Z.png

  - salida/color/HSV/ferrari_HSV.png
  - salida/color/HSV/ferrari_HSV_H.png
  - salida/color/HSV/ferrari_HSV_S.png
  - salida/color/HSV/ferrari_HSV_V.png

D) Umbralización (módulo 3)

Objetivos:

- Abrir una imagen o secuencia de imágenes y realizar una binarización basada en thresholding o umbralizacion básica con el objetivo de separar el fondo de la imagen de los objetos.

Procesos realizados:

Hemos usado la binarización thresholding aplicada a poma.jpg. Primero la hemos convertido a escala de grises.

```
cvtColor(imatge, imgAux1, CV_BGR2GRAY);
```

Luego se le han aplicado diferentes valores para la detección del umbral y separación del fondo (en concreto: 100, 150, 200 y 240). El valor máximo se ha mantenido constante en 255 y el thresholding es del tipo binario. No obstante, OpenCV también nos da la posibilidad de aplicar un threshold binario invertido, truncado, threshold a cero y threshold a cero invertido.

```
threshold(imgAux1, imgAux2, valorsThresh[i], 255, THRESH_BINARY);
```

Los resultados se pueden ver en los ficheros de salida. Si usamos un umbral bajo, parte del objeto se confunde con el fondo debido a la iluminación. Si usamos uno alto, el objeto destaca claramente sobre el fondo.

Ficheros de imágenes:

- Entrada:
  - entrada/poma.jpg
- Salida:
  - salida/umbralización/poma_threshold_100.jpg
  - salida/umbralización/poma_threshold_150.jpg
  - salida/umbralización/poma_threshold_200.jpg
  - salida/umbralización/poma_threshold_240.jpg

E) Ruido (módulo 3)

Objetivos:

- Aplicación de funciones para mejora y eliminación de ruido.
- Perturbar la imagen original con diferentes tipos de ruido y aplicar algoritmos de eliminación del mismo.
- Justificar el mejor método en cada caso.

Procesos realizados:

Las imágenes del mundo real se pueden ver como señales que a menudo contienen alteraciones o valores distorsionados con respecto a una señal que se produciría en un mundo ideal.

En primer lugar hemos cogido la imagen ferrari.png y le hemos aplicado diferentes tipos de ruido (se ignoraran los ruidos de tipo uniforme):

- gaussiano/aditivo (distribución normal de media 100)

```
Mat imgRenouGaussia;

Mat imatge_gaussian_noise = canalsImatge[0].clone();
Mat media = canalsImatge[0].clone();
media.setTo(100);
for (int i = 0; i < imatge.channels(); i++) {
	randn(imatge_gaussian_noise, 128, 30);
	canalsImatge[i] = (imatge_gaussian_noise - media) + canalsImatge[i];
	canalsImatge[i] = (media - imatge_gaussian_noise) + canalsImatge[i];
}
merge(canalsImatge, 3, imgRenouGaussia);
imwrite("ferrari_renou_gaussia_generat.png", imgRenouGaussia);
```

- impulsional/salt&pepper (entre 0 y 255)

```
Mat imgRenouSaltPepper;

Mat imatge_saltpepper_noise = Mat::zeros(imatge.rows, imatge.cols, CV_8U);
randu(imatge_saltpepper_noise, 0, 255);
Mat black = imatge_saltpepper_noise < 12;
Mat white = imatge_saltpepper_noise > 253;
media.setTo(128);
for (int i = 0; i < imatge.channels(); i++) {
	canalsImatge[i].setTo(255, white);
	canalsImatge[i].setTo(0, black);
}
merge(canalsImatge, 3, imgRenouSaltPepper);
imwrite("ferrari_renou_saltpepper_generat.png", imgRenouSaltPepper);
```

La idea es aplicar filtros para eliminar todo el ruido que podamos y mejorar la calidad de la imagen. Los efectos del ruido pueden paliarse con un filtro pasa-bajo, pero debemos tener en cuenta que cuando se suaviza la imagen pueden eliminarse pequeños detalles. Los filtros pueden implementarse tanto en el dominio de las frecuencias como en el dominio espacial, aunque las técnicas más efectivas para eliminar el ruido trabajan en el dominio espacial. Consideramos los siguientes filtros:

- Promedio de imágenes. Si tenemos varias imágenes sucesivas de una escena estática, podemos restarlas entre ellas y observar que la diferencia no da cero. Esos valores no nulos se considerarían ruido.
- Filtro promedio. Por cada píxel, reemplazamos su valor en base al promedio de este pixel junto a los píxeles vecinos considerados por la máscara elegida.
- Filtro mediana. Similar al anterior, sólo que ahora por cada píxel de la imagen, se calcula el valor del píxel en base a la mediana y no la media/promedio.
- Suavizado gaussiano. Similar al filtro promedio, pero la máscara usada representa la forma de una campana gaussiana discretizada.
- Suavizado conservativo. Sacrifica la eliminación de ruido para preservar el detalle de las altas frecuencias (bordes y contornos). Elimina los píxeles aislados excepcionalmente de valor muy bajo o muy alto de intensidad. El filtro asegura que la intensidad de cada píxel esté dentro del rango de intensidades definida por sus vecinos.

Hemos elegido aplicar dos de estos filtros: el filtro gaussiano y el filtro mediana; que ya vienen implementados en OpenCV. Aunque también podemos programar nosotros mismos los demás filtros. Por ejemplo, podríamos haber aprovechado que tenemos la imagen original ferrari.png y usar el filtro promedio de imágenes a las dos imágenes a las que les hemos añadido ruido, restando estas a la original para eliminar por completo el ruido. Cada filtro se ha aplicado a los dos tipos de ruido con diferentes parámetros:

- Filtro gaussiano (gaussian kernel size = 25x25; sigmaX = sigmaY = 1, 3, 5, 7)

```
GaussianBlur(img, imgAux, Size(25,25), 1, 1);

GaussianBlur(img, imgAux, Size(25,25), 3, 3);

GaussianBlur(img, imgAux, Size(25,25), 5, 5);

GaussianBlur(img, imgAux, Size(25,25), 7, 7);
```

Observamos que el filtro gaussiano actúa mejor con el ruido de tipo gaussiano que no con el ruido salt&pepper. SigmaX y SigmaY igual a 1 y 3 son los mejores parámetros.

- Filtro mediana (aperture lineal size = 1, 3, 5, 7)

```
medianBlur(img, imgAux, 1);

medianBlur(img, imgAux, 3);

medianBlur(img, imgAux, 5);

medianBlur(img, imgAux, 7);
```

Observamos que el filtro mediana actúa bien tanto para el ruido de tipo gaussiano como para el ruido salt&pepper. Quizás algo mejor con salt&pepper. Ambos para los parámetros 5 y 7.

* También estuvimos probando los applets de la web de HIPR.

Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.png

- Salida:
  - salida/ruido/ferrari_renougaussia.png
  - salida/ruido/ferrari_renougaussia_gaussianblur_1.png
  - salida/ruido/ferrari_renougaussia_gaussianblur_3.png
  - salida/ruido/ferrari_renougaussia_gaussianblur_5.png
  - salida/ruido/ferrari_renougaussia_gaussianblur_7.png
  - salida/ruido/ferrari_renougaussia_medianblur_1.png
  - salida/ruido/ferrari_renougaussia_medianblur_3.png
  - salida/ruido/ferrari_renougaussia_medianblur_5.png
  - salida/ruido/ferrari_renougaussia_medianblur_7.png

  - salida/ruido/ferrari_renousaltpepper.png
  - salida/ruido/ferrari_renousaltpepper_gaussianblur_1.png
  - salida/ruido/ferrari_renousaltpepper_gaussianblur_2.png
  - salida/ruido/ferrari_renousaltpepper_gaussianblur_3.png
  - salida/ruido/ferrari_renousaltpepper_gaussianblur_4.png
  - salida/ruido/ferrari_renousaltpepper_medianblur_1.png
  - salida/ruido/ferrari_renousaltpepper_medianblur_3.png
  - salida/ruido/ferrari_renousaltpepper_medianblur_5.png
  - salida/ruido/ferrari_renousaltpepper_medianblur_7.png

F) Histograma (módulo 3)

Objetivos:

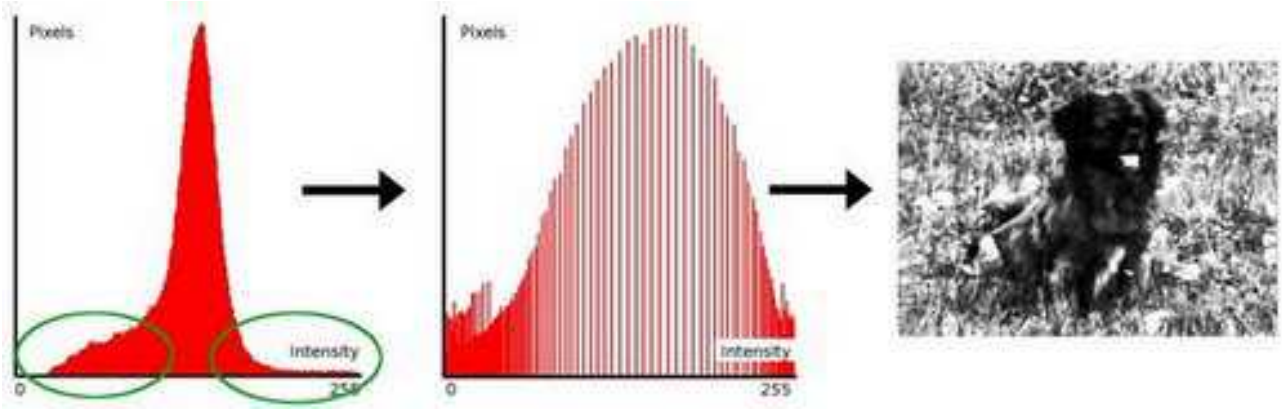
- Realizar operaciones de mejora del contraste de la imagen mediante manipulación del histograma.
- Indicar para que sirve cada una.

Procesos realizados:

Un histograma es una representación gráfica de la distribución de intensidades en una imagen. Cuantifica el número de píxeles por cada valor de intensidad considerado.

A parte de la librería OpenCV, hemos probado a usar un programa de tratamiento de imágenes (Gimp, en concreto), para ver los histogramas de varias imágenes.

Además, en la documentación de OpenCV aparece la función equalizeHist, que según se indica, es un método que mejora el contraste en una imagen automáticamente, con el fin de estirar hacia fuera el rango de intensidad.

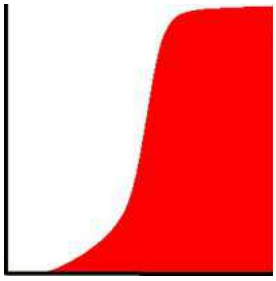


Los círculos verdes indican las intensidades poco pobladas. Después de aplicar la igualdad, obtenemos un histograma como la figura en el centro. La imagen mejorada resultante se muestra a la derecha.

La operación de ecualización implica mapear una distribución (el histograma dado) a otra distribución (una distribución más amplia y más uniforme de los valores de intensidad), por lo que los valores de intensidad son difundidos en todo el rango. Para lograr el efecto de ecualización, dado un histograma H(i), la reasignación debería ser la función de distribución acumulativa (la integral del histograma):

$$H'(i) = \sum_{0 \le j < i} H(j)$$

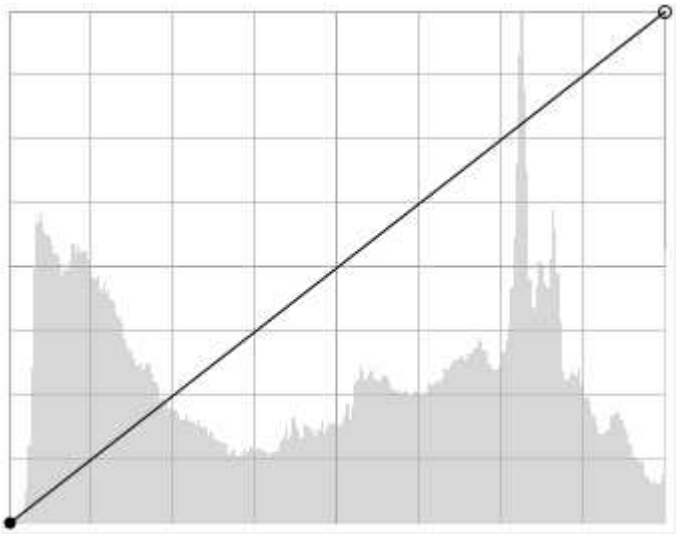
Luego se normaliza H'(i) con el valor de intensidad máximo (en nuestro caso 255). Siguiendo el ejemplo anterior:

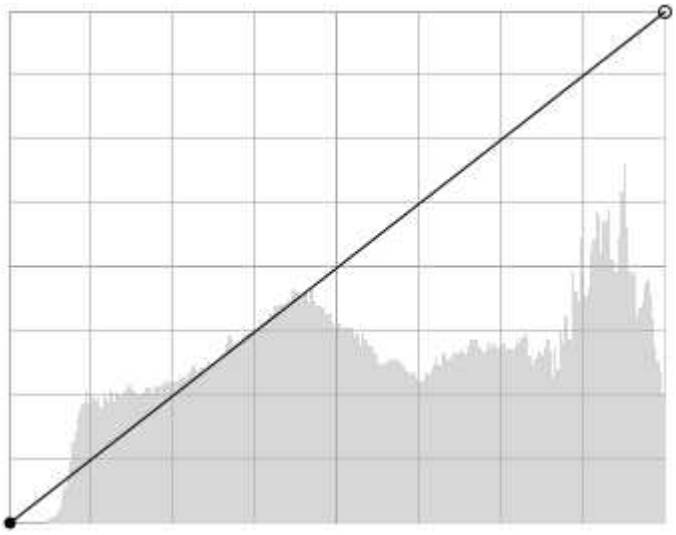


Finalmente, obtenemos los valores de intensidad de la imagen ecualizada:

$$\mathrm{equalized}(x,y) = H'(\mathrm{src}(x,y))$$

Hemos usado ferrari.jpg para aplicarle la función descrita y ver como mejora la imagen. La función se aplica a cada canal RGB por separado y luego se hace un merge. Tenemos el histograma de la imagen original a la izquierda; y a su derecha podemos ver el de la imagen ecualizada:





Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.jpg

- Salida:
  - salida/histograma/ferrari_equalitzat.jpg

G) Contornos (módulo 4)

Objetivos:

- Realizar la detección de contornos mediante filtros básicos.
- Realizar técnicas de realce de contornos.

Procesos realizados:

Existen los siguientes operadores para la detección de contornos:

- Primer orden (basados en la 1ª derivada).
  - Roberts. Usa una máscara de 2x2 que lo hace eficiente computacionalmente aunque sensible al ruido.
  - Prewitt. Hace uso de una máscara de 3x3 para hacerlo menos sensible al ruido. Diseñado especialmente para responder a bordes verticales y horizontales.
  - Sobel. Máscara de 3x3 pero dando más importancia a los píxeles centrales, para hacerlo apto para detectar bordes diagonales.

- Segundo orden (basados en la 2ª derivada).
  - Laplace. Se aplica la 2ª derivada y se busca el paso por el cero (zerocrossing). Es decir: pasaje desde valores positivos a negativos o viceversa, dado que este cambio frecuentemente ocurre en los píxeles del contorno.

- Otros.
  - Canny. Algoritmo que consta de varias etapas: Realizar una convolución de la imagen original junto a una función gaussiana para suavizar; Obtener la 1ª derivada usando algún operador de primer orden; comparar cada pixel con sus vecinos para descartar aquellos que no sean máximos locales; y finalmente se usan dos umbrales para decidir si un pixel pertenece o no a un borde.
  - Ajuste al modelo. Se usan varias máscaras de convolución, cada una de ellas sensible a una orientación de borde.
  - Detección de líneas. Tienen por objeto detectar aquellos píxeles que son parte de una línea en la imagen. Es decir, que son una región oscura limitada en ambos lados por regiones claras o viceversa.

De todos los operadores anteriores vamos a usar uno de cada categoría: Sobel, Laplace y Canny. En todos los casos se ha convertido la imagen RGB a escala de grises. A continuación se presenta el mejor resultado para cada operador después de probar varios parámetros:

- Sobel(src, dst, profundidad_dst, orden_derivada_x, orden_derivada_y, tam_kernel)

```
Mat imgSobel;
Mat grad_x, grad_y;
Mat abs_grad_x, abs_grad_y;
Sobel(imgAux, grad_x, CV_16S, 1, 0, 3);
convertScaleAbs(grad_x, abs_grad_x);
Sobel(imgAux, grad_y, CV_16S, 0, 1, 3);
convertScaleAbs(grad_y, abs_grad_y);
addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, imgSobel);
```

- Laplacian(src, dst, profundidad_dst)

```
Mat imgLaplace;
Laplacian(imatge, imgLaplace, 8);
```

- Canny(src, dst, umbral_T1, umbral_T2)

```
Mat imgCanny;
Canny(imatge, imgCanny, 210, 230);
```

Mientras que la detección de contornos persigue aislar los píxeles pertenecientes al borde de los objetos, el realce de contornos busca resaltar en la imagen original estos píxeles. Si la imagen tuviese ruido, tendríamos que haber aplicado algún filtro para paliarlo, ya que los bordes pertenecen a las altas frecuencias, como el ruido, que se hubiese visto multiplicado. Realizaremos dos pasos:

1. Se substrae de la imagen original una versión suavizada para obtener una imagen de los contornos. Se ha usado filtro mediana:

```
medianBlur(imatge, imatgeSuavitzada, 3);
```

2. Se adiciona a la imagen original la imagen de los contornos anterior, la cual multiplicamos por una constante k para establecer el grado en que realzamos los bordes. Se ha probado con k=1, 3, 5 y 7.

```
for (int k=1; k<=7; k=k+2) {
        imgAux = imatge + k * (imatge - imatgeSuavitzada);
        imwrite("image1", imgAux);
}
```

Cuanto mayor es la constante, más quedan realzados los contornos. Aunque una constante demasiado alta puede dar a la imagen sensación de artificial.

Ficheros de imágenes:

- Entrada:
  - entrada/ferrari.jpg

- Salida:
  - salida/contornos/ferrari_sobel.jpg
  - salida/contornos/ferrari_canny.jpg
  - salida/contornos/ferrari_laplace.jpg
  - salida/contornos/ferrari_realcecontornos_1.jpg
  - salida/contornos/ferrari_realcecontornos_3.jpg
  - salida/contornos/ferrari_realcecontornos_5.jpg
  - salida/contornos/ferrari_realcecontornos_7.jpg

H) Segmentación (módulo 5)

Objetivos:

- Realizar la segmentación de una imagen de niveles de gris.
- Aplicar métodos de umbralización, de crecimiento de regiones y morfológicos.

Procesos realizados:

La segmentación consiste en separar los objetos presentes en una imagen de su fondo y poder distinguirlos entre si a partir de características como los niveles de gris, color, textura o bordes.

Las técnicas usadas para segmentar una imagen se basan en:

- Fijación de umbrales. Basada en la umbralización/thresholding, que toma una imagen de entrada y un parámetro umbral para devolver una imagen binaria, aunque puede extrapolarse a más umbrales.
- Detección de bordes. Hace uso del cambio de intensidad que se produce en los píxeles del borde de un objeto, pero por si solo no determina su forma.
- Crecimiento de regiones. Conjunto de técnicas en la que los píxeles se agrupan en regiones en base a determinados atributos.

Una vez aplicados los algoritmos de segmentación, cada objeto de la imagen puede ser estudiado individualmente para extraer sus características. Eso sí, resulta útil mejorar previamente los resultados obtenidos mediante la segmentación eliminando el ruido producido mediante técnicas morfológicas.

Hemos usado 5 imágenes de canicas para hacer las pruebas.

UMBRALIZACIÓN

Según nos dice el enunciado, debemos aplicar umbralización. En concreto, umbralización adaptativa, que cambia los umbrales que usa según como sea el rango de intensidades de la imagen.

```
adaptiveThreshold(src, dst, maxValue, adaptativeMethod, thresholdType, blockSize, cnst);
```

La mejor combinación de parámetros después de hacer pruebas es la siguiente:

```
adaptiveThreshold(imgCanicas, imgAux, 150, ADAPTIVE_THRESH_MEAN_C, THRESH_BINARY, 111, 15);
```

El resultado es bastante bueno y se puede verse en las imágenes adjuntas. Aún así, hay ruido que deberá desaparecer con transformaciones morfológicas.

CRECIMIENTO DE REGIONES

Continuando con el enunciado, aplicaremos crecimiento de regiones, que a su vez dispone de dos técnicas: regiones a partir de semillas, en la que se eligen unos puntos iniciales para ir añadiendo cada vez píxeles vecinos de propiedades similares; y división y unión de regiones, que se basa en la estructura quadtree.

Usamos la última técnica. Se ha utilizado el algoritmo exacto en la documentación de OpenCV.

Hay que decir, que no hemos conseguido obtener buenos resultados con crecimiento de regiones.

MORFOLÓGICOS

Las transformaciones morfológicas modifican la estructura de las regiones detectadas. Toman como entrada un conjunto de píxeles de la imagen y el elemento estructural (conjunto de puntos dispuestos de diferentes formas según lo que se quiera conseguir).

Clasificación:

- Erosión. Desgaste de los límites de las regiones que pertenecen a los objetos. Los agujeros internos de las regiones se vuelven más grandes y disminuyen las regiones de los objetos.
- Dilatación. Crecimiento de los límites de las regiones que pertenecen a los objetos. Los agujeros internos de las regiones se vuelven más pequeños o desaparecen y aumentan las regiones de los objetos.
- Apertura. Se define como una erosión seguida de una dilatación compartiendo el mismo elemento estructural. Así conseguimos eliminar pequeños elementos debidos al ruido.
- Cierre. Se define como una dilatación seguida de una erosión compartiendo el mismo elemento estructural. Así conseguimos rellenar los agujeros pequeños internos y suavizar los contornos de los objetos.

Nos decantamos por realizar una apertura: primero erosión y luego dilatación. El elemento estructural será una elipse de 3x3 y centro en (2, 2).

```
Mat elementEstr = getStructuringElement(MORPH_ELLIPSE, Size(3, 3), Point(2, 2));
erode(imgAux, imgAux, elementEstr);
dilate(imgAux, imgAux, elementEstr);
```

Hemos aplicado dichas transformaciones morfológicas a partir del resultado obtenido con la segmentación por umbralización ya que nos ha dado un mejor resultado que con crecimiento de regiones. A pesar de todo, los resultados finales no son excesivamente buenos, debido principalmente a los reflejos y transparencias de las canicas.

Ficheros de imágenes:

- Entrada:
  - entrada/canicas1.jpg
  - entrada/canicas2.jpg
  - entrada/canicas3.jpg
  - entrada/canicas4.jpg
  - entrada/canicas5.jpg

- Salida:
  - salida/segmentacion/segmentacion_umbralizacion_canicas1.jpg
  - salida/segmentacion/segmentacion_umbralizacion_canicas2.jpg
  - salida/segmentacion/segmentacion_umbralizacion_canicas3.jpg
  - salida/segmentacion/segmentacion_umbralizacion_canicas4.jpg
  - salida/segmentacion/segmentacion_umbralizacion_canicas5.jpg

  - salida/segmentacion/segmentacion_crecimientoregiones_canicas1.jpg
  - salida/segmentacion/segmentacion_crecimientoregiones_canicas2.jpg
  - salida/segmentacion/segmentacion_crecimientoregiones_canicas3.jpg
  - salida/segmentacion/segmentacion_crecimientoregiones_canicas4.jpg
  - salida/segmentacion/segmentacion_crecimientoregiones_canicas5.jpg

  - salida/segmentacion/segmentacion_operacionesmorfologicas_canicas1.jpg
  - salida/segmentacion/segmentacion_operacionesmorfologicas_canicas2.jpg
  - salida/segmentacion/segmentacion_operacionesmorfologicas_canicas3.jpg
  - salida/segmentacion/segmentacion_operacionesmorfologicas_canicas4.jpg
  - salida/segmentacion/segmentacion_operacionesmorfologicas_canicas5.jpg

I)Extracción de Características (módulo 6)

Objetivos:

- Realizar una extracción de características para la selección de regiones.
- Mirar propiedades métricas, topológicas y de textura.

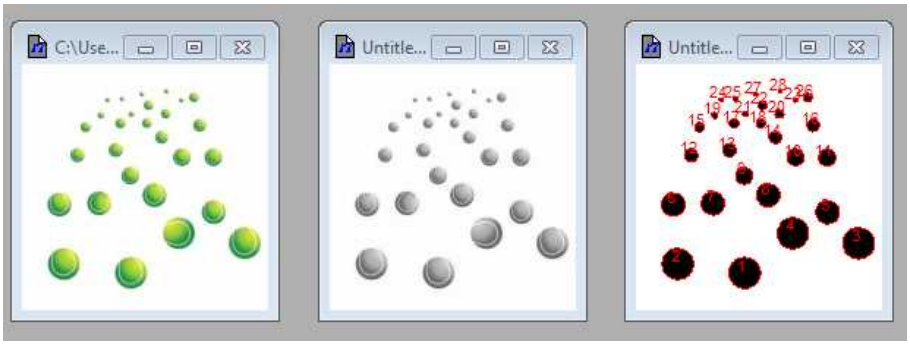
Procesos realizados:

Después de aplicar un algoritmo de segmentación, cada objeto de la imagen puede ser estudiado individualmente para extraer sus características. Dicha extracción de características consiste en obtener una serie de valores numéricos que describan las regiones segmentadas. Aunque existen muchas métricas como el área, perímetro, compacticidad, rectangularidad, centro de gravedad, orientación, momentos invariantes a la rotación, escala y traslación y signatura, nosotros nos hemos decantado por las siguientes métricas:

MÉTRICAS

- Área
- Perímetro
- Compactitud
- Centroide
- Elongación
- Redondez
- Longitud del eje principal
- Ángulo del eje principal
- Longitud del eje secundario
- Ángulo del eje secundario
- Diámetro de Ferret

En este caso hemos usado el software propuesto por el profesor (ImageTool) para facilitarnos el trabajo, que de otro modo sería mucho más complejo de implementar. El proceso a seguir va a ser el siguiente: primero se transforma la imagen a escala de grises. Luego se debe binarizar la imagen utilizando thresholding. Finalmente, una vez que se ha realizado el thresholding se han de detectar contornos y objetos.



A continuación se enseña una imagen de la tabla de resultados, aunque si se precisa revisarla con precisión se ha de consultar el anexo **Anexo I)** , donde se adjuntan todos los resultados en formato de hoja de cálculo.

Objeto	Área	Perímetro	Longitud del eje principal	Ángulo del eje principal	Longitud del eje secundario	Ángulo del eje secundario	Elongación	Redondez	Diámetro de Ferret	Compactitud	Centroide
#1	397	76,36	22,36	10,3	21,38	-79,22	1,05	0,86	22,48	1,01	(72,143)
#2	395	73,18	22,36	26,57	22,36	-63,43	1	0,93	22,43	1	(27,136)
#3	399	71,94	22,47	32,28	21,63	-56,31	1,04	0,97	22,54	1	(150,122)
#4	396	71,94	22,2	82,23	21,21	-8,13	1,05	0,96	22,45	1,01	(105,115)
#5	228	54,04	17,03	-40,24	16,28	47,49	1,05	0,98	17,04	1	(129,101)
#6	226	52,63	16,76	17,35	15,81	-71,57	1,06	1,03	16,96	1,01	(24,96)
#7	226	55,28	17,03	-86,63	16,03	3,58	1,06	0,93	16,96	1	(51,95)
#8	228	53,46	16,76	17,35	15,81	-71,57	1,06	1	17,04	1,02	(88,90)
#9	127	40,56	12,81	-51,34	12,04	41,63	1,06	0,97	12,72	0,99	(72,76)
#10	126	38,97	12,53	-28,61	11,66	59,04	1,07	1,04	12,67	1,01	(107,64)
#11	129	41,38	12,81	-38,66	12,04	48,37	1,06	0,95	12,82	1	(128,64)
#12	81	30,73	10	-36,87	9,22	49,4	1,08	1,08	10,16	1,02	(36,62)
#13	81	32,73	9,9	45	9,22	-40,6	1,07	0,95	10,16	1,03	(62,59)
#14	72	30,56	9,22	12,53	9,22	-77,47	1	0,97	9,57	1,04	(93,50)
#15	44	22,9	7,28	-74,05	6,32	18,43	1,15	1,05	7,48	1,03	(41,43)
#16	70	29,97	9,22	77,47	8,25	-14,04	1,12	0,98	9,44	1,02	(119,42)
#17	44	21,31	7,28	15,95	6,32	-71,57	1,15	1,22	7,48	1,03	(65,40)
#18	41	20,49	6,71	63,43	6,71	-26,57	1	1,23	7,23	1,08	(83,40)
#19	22	18,07	5,1	-78,69	4,12	14,04	1,24	0,85	5,29	1,04	(52,35)
#20	39	25,9	7,21	-56,31	5,83	30,96	1,24	0,73	7,05	0,98	(96,34)
#21	17	13,24	4,47	-26,57	3,61	56,31	1,24	1,22	4,65	1,04	(73,34)
#22	39	21,49	6,71	63,43	6,71	-26,57	1	1,06	7,05	1,05	(84,28)
#23	12	13,24	3,61	-33,69	2,83	45	1,27	0,86	3,91	1,08	(107,25)
#24	11	12,41	3,61	-33,69	2,24	63,43	1,61	0,9	3,74	1,04	(56,24)
#25	15	14,41	4,24	-45	2,83	45	1,5	0,91	4,37	1,03	(66,24)
#26	40	23,07	7,07	-45	7,07	45	1	0,94	7,14	1,01	(115,23)
#27	11	12,41	3,61	-33,69	2,24	63,43	1,61	0,9	3,74	1,04	(80,21)
#28	9	12	2,83	45	2,83	-45	1	0,79	3,39	1,2	(97,18)
Media	125,89	35,17	10,83	-7,15	10,06	-0,75	1,14	0,97	11	1,03	-
Desviación Estándar	132,71	20,76	6,44	48,97	6,46	51,36	0,17	0,12	6,39	0,04	-

Se puede comprobar a simple vista que los resultados son muy buenos, ya que la compactitud de los objetos tiene como media 1 y muy poca desviación. Lo mismo ocurre con la redondez.

Ficheros de imágenes:

- Entrada:
  - entrada/tenis.jpg

- Salida:
  - salida/caracteristicas/tenis_escala_grises.jpg
  - salida/caracteristicas/tenis_threshold.jpg
  - salida/caracteristicas/tenis_objetos.jpg