Práctica 2 – Histogramas, bordes y regiones

Este ejercicio tiene como objetivo trabajar en OpenCV con los histogramas, extracción de bordes y detección de regiones. Todo ello visto en el **Tema 4: Transformaciones y correcciones** y **Tema 5: Bordes, regiones y puntos de interés**.

Puntos totales posibles del ejercicio: 10

Instrucciones

Cada integrante del grupo debe acceder al siguiente enlace.

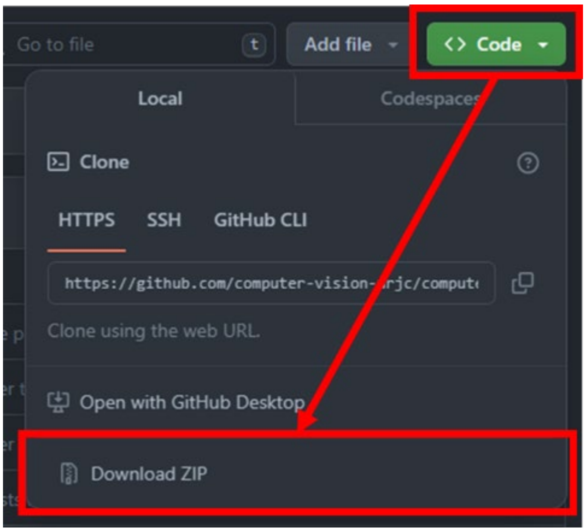
Si no lo ha hecho anteriormente, deberá asociarse con el usuario creado a partir de su correo electrónico de la **URJC**. Una vez hecho esto, deberá unirse al grupo de la asignatura. Si dicho grupo no existe, uno de los miembros deberá crearlo, utilizando el **mismo nombre** que figura en el **Aula Virtual**.

📌 La plantilla con el nodo **ROS 2** proporcionada deberá modificarse para que el nombre del paquete sea: **practica2-grupoX**, donde **X** es el número de grupo asignado en el **Aula Virtual**.

⚠️ **Importante:** No se debe modificar el archivo de cabecera (.hpp) de la plantilla proporcionada. Todo el código necesario deberá implementarse únicamente en el archivo fuente (.cpp).

Entrega

La **entrega** consistirá en subir al **Aula Virtual** el **archivo .zip** generado a través del repositorio de GitHub Classroom.



Si no se cumplen los criterios anteriores, o se entrega un paquete que no compila, la calificación será 0

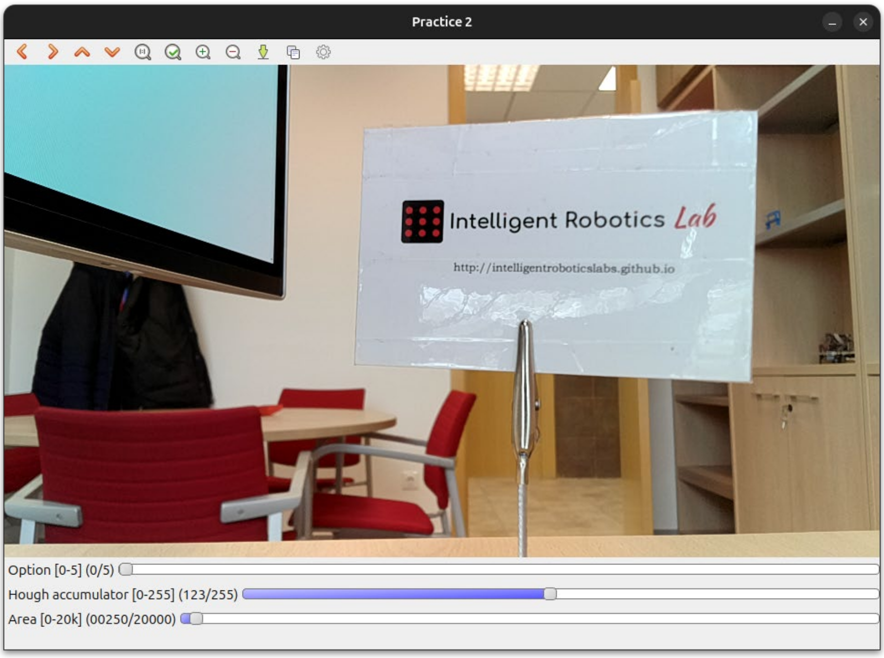
Enunciado

Para lanzar la cámara, se utilizará el siguiente comando a fin de mejorar el rendimiento:

```
// Camera launcher
ros2 launch oak_d_camera rgbd_stereo.launch.py only_rgb:=true use_rviz:=false
```

Se pide crear un programa que trabaje con la imagen y muestre varios controles deslizantes (*sliders*) como los de la figura:

1. **Option**: que irá de 0 a 5.
2. **Hough accumulator**: que irá de 0 a 255.
3. **Área**: que irá de 0 a 20000.



Option [0-5] (0/5)
Hough accumulator [0-255] (123/255)
Area [0-20k] (00250/20000)

Para cada una de las **6 opciones**, se debe realizar lo siguiente con la imagen **BGR**:

- **Opción 0:** mostrar la imagen en formato de color **BGR**.
- **Opción 1:** sobre la imagen, se podrá realizar clic con el **botón izquierdo del ratón** en diferentes coordenadas. En las tres primeras, se mostrará el texto con número 1, 2, 3 según corresponda con el clic realizado. Al obtener una cuarta pulsación, se utilizarán esos cuatro puntos para **rectificar la imagen**, de manera que el clic 1 tenga su posición en la esquina superior izquierda de la imagen original, el clic 2 en la esquina superior derecha, el 3 en la esquina inferior izquierda y el 4 en la esquina inferior derecha (zoom). Al realizar el siguiente clic sobre la imagen, se volverá a mostrar la imagen original, y se podrá pulsar nuevamente sobre ella.

- **Opción 2:** se creará una **ventana nueva** donde se mostrarán los **histogramas de cada canal BGR** de la imagen original (en rojo para el canal R, verde para el G y azul para el B). Además, se realizará un **ecualizado** sobre cada canal de la imagen original de forma independiente, mostrando estos histogramas en la ventana de histogramas anterior (mismo color, pero menor intensidad) y como resultado, la combinación de estos tres canales ecualizados en la ventana de la práctica. En la ventana de histogramas se deberá mostrar en color rojo, el valor obtenido al **comparar** con correlación el canal R de la imagen original y el ecualizado de la imagen resultante, en verde la comparación con el canal G, y en azul el del canal B. Así mismo, en base al histograma de la imagen original, se deberá indicar en amarillo si la imagen está **subexpuesta**, **bien expuesta**, o **sobreexpuesta**, teniendo en cuenta que el rango en el que se considera una buena exposición está entre 65 y 190.
- **Opción 3:** detectar las **líneas de la hoja** utilizando la **transformada de Hough** estándar, independientemente de su dirección. El **acumulador** de puntos estará controlado por un **segundo *slider***, y permitirá elegir valores entre 0 y 255. Queda a vuestro criterio diseñar un algoritmo que detecte las líneas lo más fiable posible, independientemente de su tamaño y longitud, y evitar que haya solape. Incluye sobre la imagen un **texto** con la cantidad de **líneas detectadas**.
- **Opción 4:** identificar solo la **pelota rosa** utilizando la **transformada circular de Hough**. No hay límites en cuanto a la cantidad de píxeles, tamaño del círculo, etc. Queda a vuestro criterio diseñar un algoritmo que detecte la pelota lo más fiable posible independientemente de su tamaño, del número de pelotas, la distancia, o de su oclusión con otros objetos. Hay que **dibujar el contorno y su centro**.
- **Opción 5:** detectar únicamente los **rectángulos** que aparecen en la hoja. Para ello, se extraerán los contornos, y solo se tienen en cuenta aquellos que tienen una **cantidad de píxeles** superior al umbral establecido por el **tercer *slider***. Se mostrará tanto el **borde** como el **centroide** del contorno, para ello es necesario calcular los momentos de los contornos y utilizar las siguientes fórmulas:

  $$C_x = \frac{M_{10}}{M_{00}}, \qquad C_y = \frac{M_{01}}{M_{00}}$$

  Utilizar todo lo que creas necesario para detectar únicamente los rectángulos.

Para incluir un texto en una imagen se proporciona la siguiente función:

```
// Write text in an image
cv::putText(image, text, cv::Point(10, 20),
        cv::FONT_HERSHEY_SIMPLEX, 0.5, cv::Scalar(0, 0, 255));
```

Para añadir un evento que controle los clics de ratón, es necesario crear un Callback:

```
// create mouse callback
cv::setMouseCallback( "window_name", on_mouse, 0 );
```

donde “window_name” es el nombre de la ventana que controlará este evento, y la función on_mouse es la función a la que se llama cuando ocurra una pulsación, la cual se define como sigue:

```
// create mouse callback
void on_mouse(int event, int x, int y, int, void*)
{
    switch (event){
      case cv::EVENT_LBUTTONDOWN:
        std::cout << "Left button" << std::endl;
        break;
      case cv::EVENT_RBUTTONDOWN:
        std::cout << "Right button" << std::endl;
        break;
      default:
        std::cout << "No button" << std::endl;
        break;
    }
}
```

Ayuda

Funciones del Trackbar, waitKey, pollKey:
https://docs.opencv.org/3.4/d7/dfc/group__highgui.html

Dibujar líneas, círculos y texto:
https://docs.opencv.org/4.x/d6/d6e/group__imgproc__draw.html

Histogramas:
https://docs.opencv.org/3.4/d8/dbc/tutorial_histogram_calculation.html
https://docs.opencv.org/3.4/d4/d1b/tutorial_histogram_equalization.html

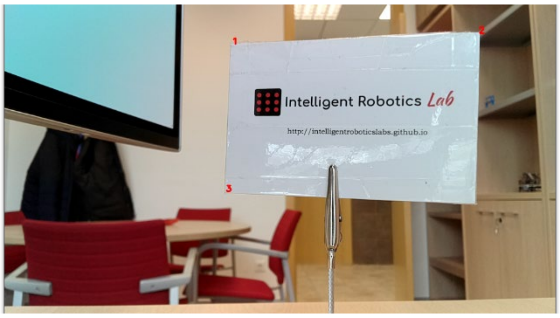
Hough Lines:
https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

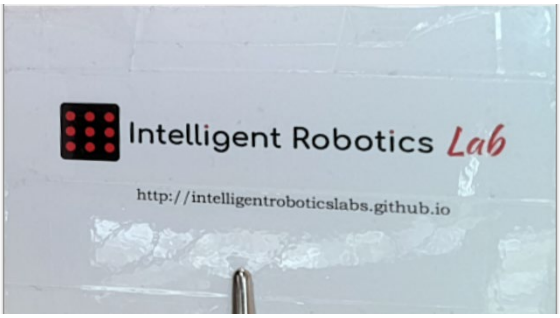
Hough Circles:
https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html

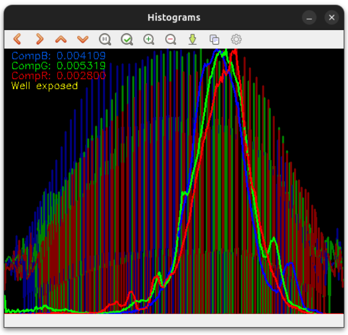
Propiedades de los contornos:
https://docs.opencv.org/3.4/d1/d32/tutorial_py_contour_properties.html

Características de los contornos:
https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html

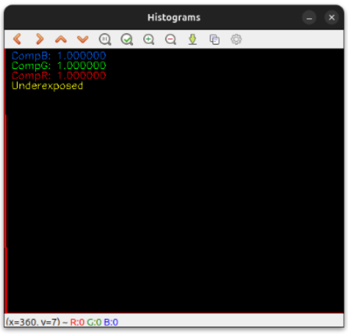
Imágenes



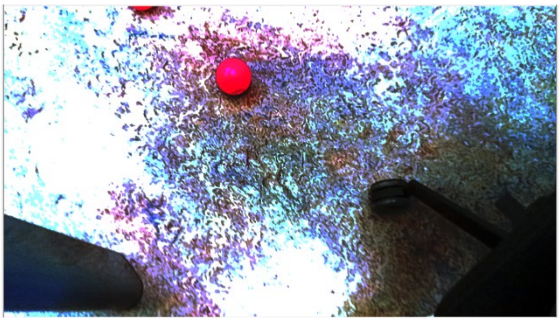


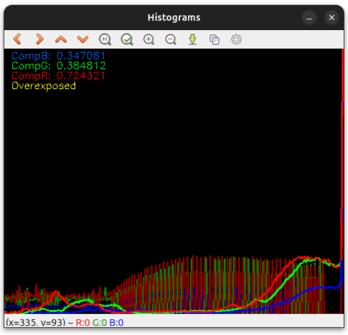


```
CompB: 0.004109
CompG: 0.005319
CompR: 0.002800
Well exposed
```



```
CompB: 1.000000
CompG: 1.000000
CompR: 1.000000
Underexposed
```





```
CompB: 0.347081
CompG: 0.384812
CompR: 0.724321
Overexposed
```