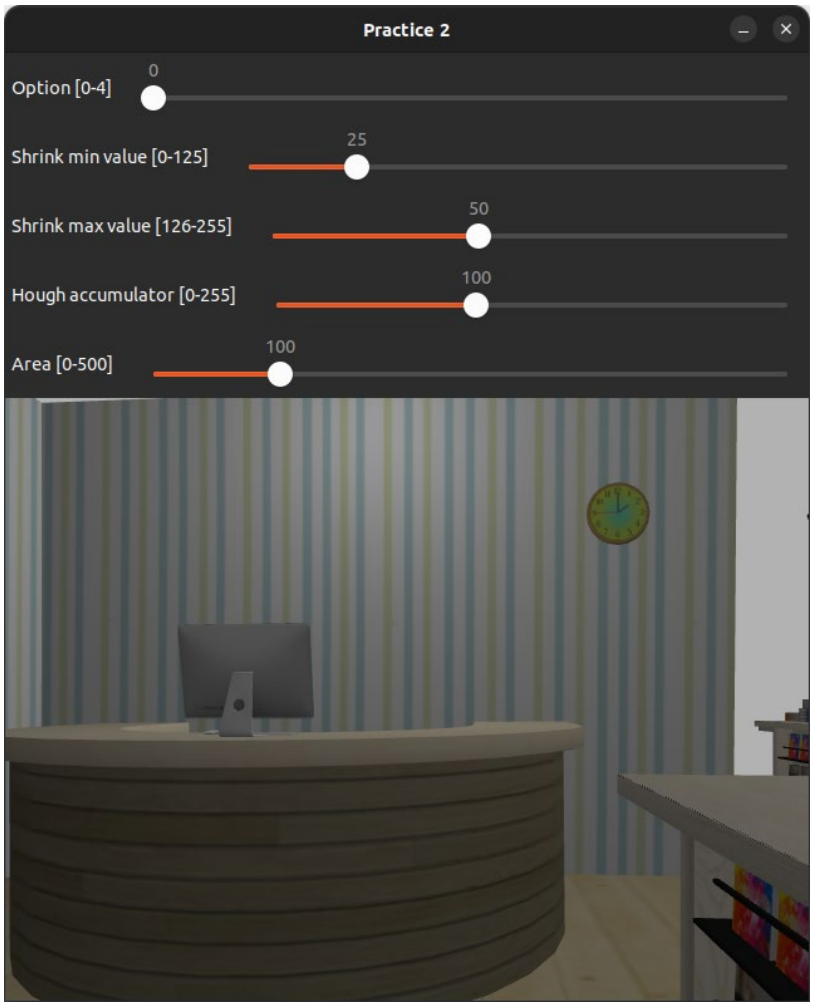
Práctica 2 – Histogramas, Bordes y Regiones

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

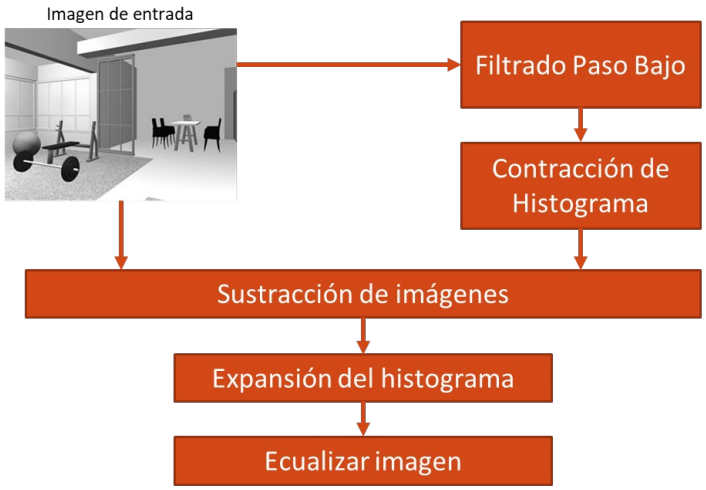
Utilizando el simulador con Tiago y el escenario de **aws_bookstore**, se pide crear un programa que trabaje con la imagen y muestre en la parte superior varios controles deslizantes (sliders) como los de la figura:



Se pide que en cada una de las **5 opciones** se haga lo siguiente con la imagen **BGR** del Tiago:

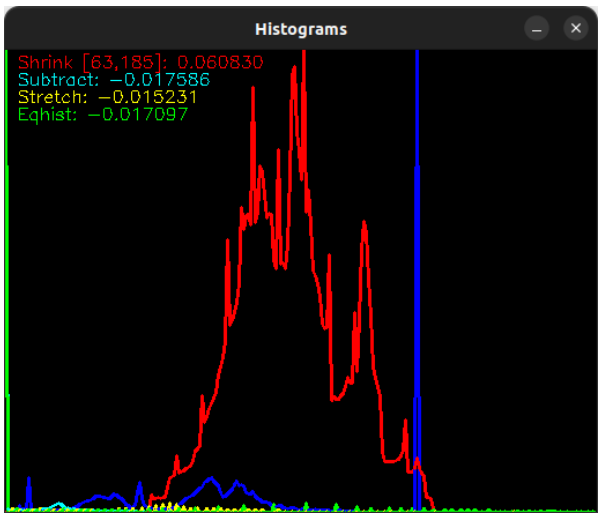
- Opción 0: Mostrar la imagen en formato de color **RGB**.

- Opción 1: Mostrar la imagen en escala de grises resultante del aplicar el siguiente proceso de **realzado**:
    1. Convertir la imagen a escala de grises.
    2. Aplicar un filtrado **paso bajo** sobre la imagen del punto 1.
    3. Contraer el histograma, usando los valores de **shrink min** y **max** modificables a través de **dos sliders**, que irán de 0 a 125 para min, y de 126 a 255 para max (aunque el slider vaya de 0 a 125).
    4. Restar píxel a píxel la imagen generada en el punto 3 a la imagen del punto 1.
    5. Expandir el histograma de la imagen resultante del punto 4 entre [0, 255].
    6. Ecualizar la imagen resultado del punto 5.



Además de la **imagen resultado** que se mostrará en la **ventana principal** de la aplicación, se debe mostrar otra **ventana auxiliar** que incluirá los **histogramas de la imagen original (azul) junto con los histogramas de las imágenes obtenidas en los puntos 3 (rojo), 4 (cyan), 5 (amarillo) y 6 (verde)**.

Dentro de este histograma se tiene que **incluir un texto** en el que se muestre el **valor obtenido al comparar cada histograma con el de la imagen original en escala de grises** aplicando la técnica de **correlación**. Además, se mostrará los valores entre los que se está haciendo la contracción.



- **Opción 2:** Identificar los **marcos del ventanal** utilizando la **transformada de Hough** estándar, independientemente de su dirección. El **acumulador** de puntos estará controlado por un **cuarto slider**, y permitirá elegir valores entre 0 y 255. Queda a vuestro criterio diseñar un algoritmo que detecte la ventana lo más fiable posible, independientemente de su tamaño, longitud, del número de marcos que se vean, o de su oclusión con otros objetos.

- **Opción 3:** Identificar el **reloj** 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 el reloj lo más fiable posible independientemente de su tamaño, del número de relojes, la distancia, o de su oclusión con otros objetos. Hay que dibujar tanto el contorno como su centro.

- **Opción 4:** Identificar las **líneas azules de la pared**, y extraer los **contornos** obtenidos al **fusionar** esta nueva imagen con los marcos de la **opción 2**, es decir, la imagen donde se extraerán los contornos será aquella donde únicamente aparezcan los marcos del ventanal y las líneas azules de la pared. De estos contornos, únicamente se mostrarán aquellos cuya **cantidad de puntos** sea mayor que el valor modificable a través de un **quinto slider**, que irá de 0 a 500. Además, se obtendrán los **centroides** de los contornos, para ello es necesario **calcular los momentos** de los contornos y utilizar las siguientes fórmulas:

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

Para cada **contorno** y su **centroide** se le asignará un **color diferente**, y serán representados mediante un **punto con radio 4** junto al que deberá aparecer el **número de píxeles** que forman ese contorno. Además, se mostrará la **cantidad de contornos** detectados en la parte superior.

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));
```

Ayuda

Funciones del Trackbar:
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

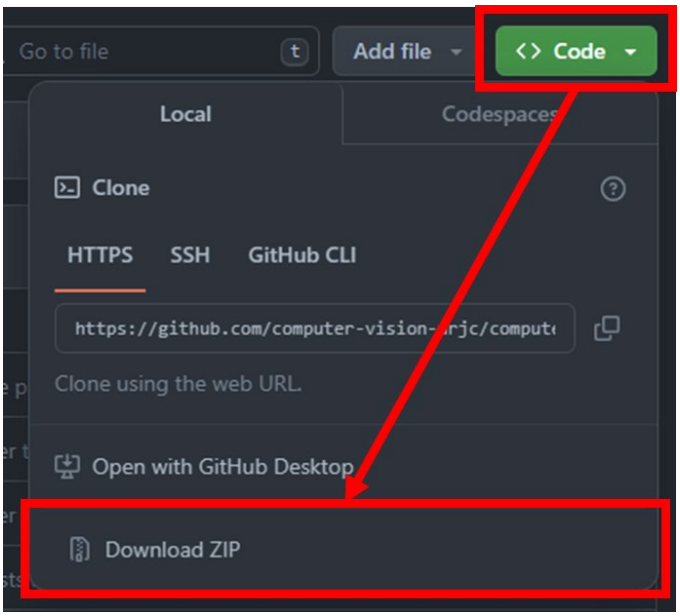
Memoria

Deberá modificarse el README y añadir un apartado para responder los siguientes puntos:

1. Adjunta una captura de los histogramas obtenidos en la opción 1 cuando los valores mínimo y máximo de la contracción son 125 y 126 respectivamente, y explica brevemente el comportamiento de cada histograma en dicha imagen.
2. ¿Es posible acotar la dirección de las líneas detectadas en la transformada de Hough? En caso afirmativo, ¿cómo? Justifique la/s respuesta/s.

Entrega

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



Para ello, será necesario que cada integrante del grupo esté dentro del curso, si no lo está, tiene que entrar en el siguiente enlace, y asociarse con el usuario creado a partir de su email de la URJC. Una vez hecho esto, deberá entrar en el grupo de la asignatura, si no existe, uno de los miembros tendrá que crearlo, siendo el nombre del grupo el mismo que figura en el Aula Virtual.

La plantilla con el nodo ROS 2 proporcionada, deberá ser modificada para que el **nombre del paquete** sea **practica1-grupoX**, donde X es el número de grupo del Aula Virtual. Es importante **no modificar el archivo cabecera (.hpp)** de la plantilla que se proporciona, todo el **código** necesario deberá estar **incluido** únicamente en su **archivo fuente (.cpp)**.

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