Ejercicio 4 – Bordes y regiones

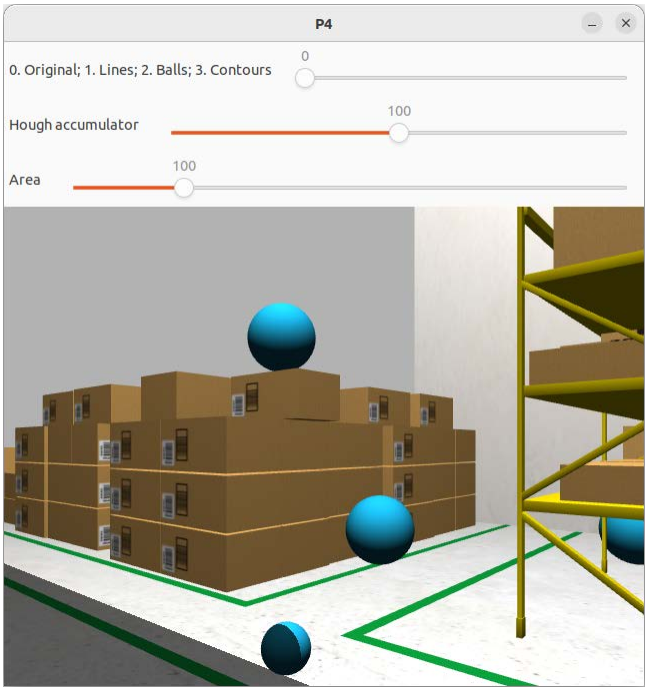
Este ejercicio tiene como objetivo aplicar los conceptos aprendidos en el Tema 5: Bordes y regiones.

La defensa del ejercicio se hará en clase, y hay que entregar un archivo **cv_node_p4.cpp** con el código generado que deberás subir al Aula Virtual.

Puntos totales posibles del ejercicio: 10

Instrucciones

Utilizando el simulador con Tiago, 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.



Para ello, utilizaremos las siguientes funciones:

```
// create Trackbar and add to a window
cv::createTrackbar("trackbar_text", "window_name", nullptr, max_value, 0);
// set Trackbar's value
cv::setTrackbarPos("trackbar_text", "window_name", value);
// get Trackbar's value
cv::getTrackbarPos("trackbar_text", "window_name");
```

donde **"trackbar text"** es el texto que queremos que aparezca en el slider; **"window name"** es el nombre de la ventana a la que se añade el slider; **max_value** es el valor máximo que puede alcanzar el slider; y **value** es el valor que se le asigna al slider (normalmente es el valor que tendrá inicialmente). Hay que tener cuidado de que los nombres sean los mismos en las distintas funciones.

Estos sliders se tienen que añadir una única vez. Para ello es posible crear la ventana inicialmente con la siguiente función:

```
// create a window
cv::namedWindow("window_name");
```

Se pide que en el escenario **aws_warehouse** en cada una de las 4 opciones se haga lo siguiente con la imagen **BGR** del Tiago:

- **Opción 0**: mostrar la imagen original sin procesar.
- **Opción 1**: Identificar utilizando la **transformada de Hough estándar**, las **cintas verdes** del suelo que aparecen, independientemente de su dirección. El **acumulador** de puntos estará controlado por un **segundo slider**, y permitirá elegir valores entre 0 y 200. Queda a vuestro criterio diseñar un algoritmo que detecte las cintas lo más fiable posible independientemente de su tamaño, longitud, del número de cintas, o de su oclusión con otros objetos. Hay que dibujar tanto el contorno como su centro.
- **Opción 2**: Identificar las **pelotas azules** 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 las pelotas lo más fiable posible independientemente de su tamaño, del número de pelotas, o de su oclusión con otros objetos. Hay que dibujar tanto el contorno como su centro.
- **Opción 3**: Se extraerán los **contornos** obtenidos al fusionar las dos imágenes filtradas de las opciones 1 y 2, es decir, la imagen donde extraer los contornos será aquella donde únicamente aparezcan las cintas verdes y las pelotas azules. De estos contornos, únicamente se mostrarán aquellos cuya **cantidad de puntos** sea mayor que el valor modificable a través de un **tercer slider**, que irá de 0 a 1000. 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** 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.

Notas

Se puede utilizar todo lo aprendido hasta ahora para la realización de la práctica, filtros, suavizados, detectores, etc.

Para cada función u operación que se desee utilizar, los valores serán aquellos que consideres mejores para la detección deseada (por ejemplo, los umbrales de histéresis de Canny o los radios de los círculos).

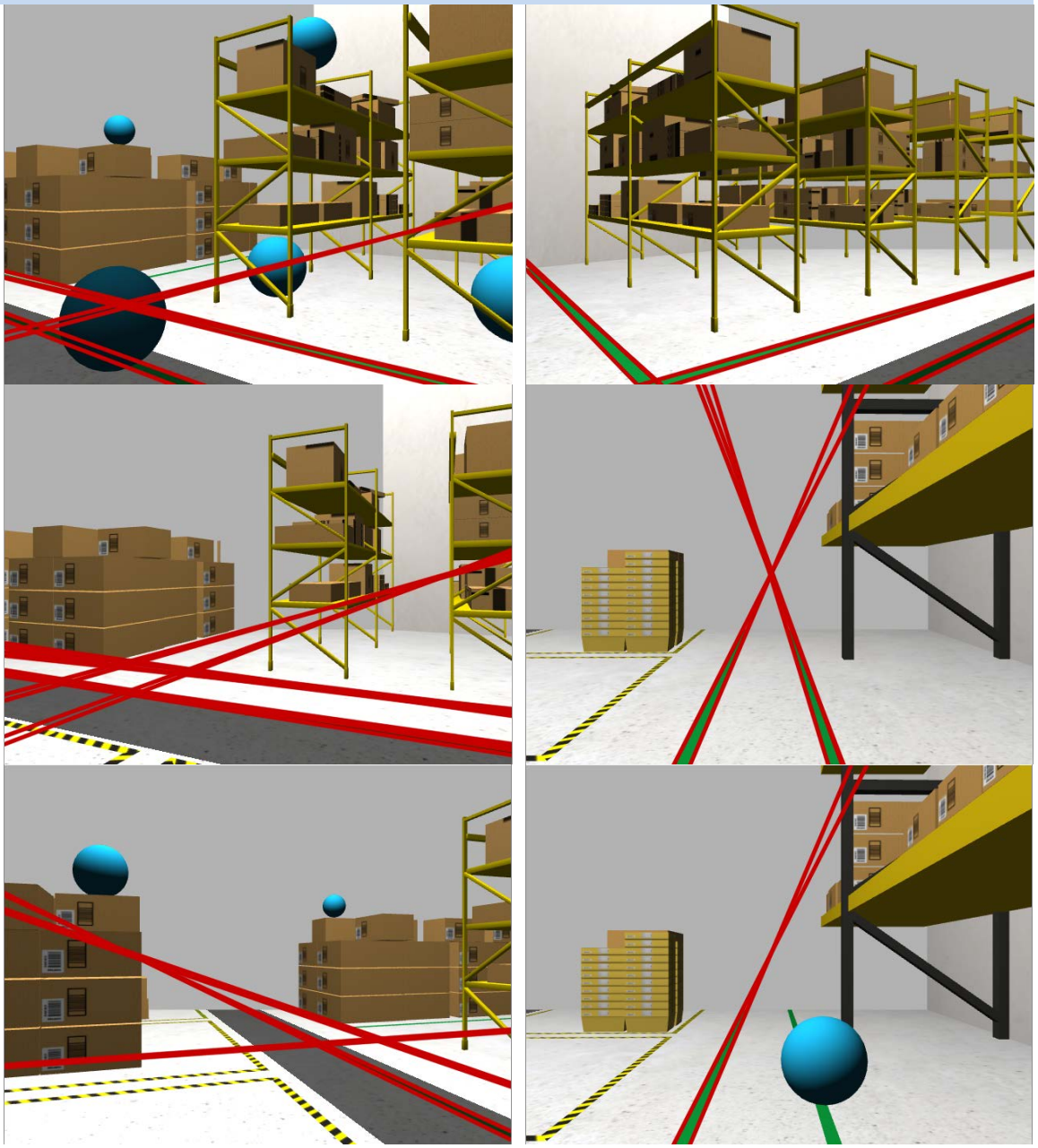
Únicamente se mostrará una ventana, con los sliders, y el resultado de cada procesamiento elegido en función de los valores de los sliders.

Las pelotas, se deben introducir desde Gazebo. Y no hay número fijo de pelotas o de cintas, hay que detectar todas de la mejor forma posible.

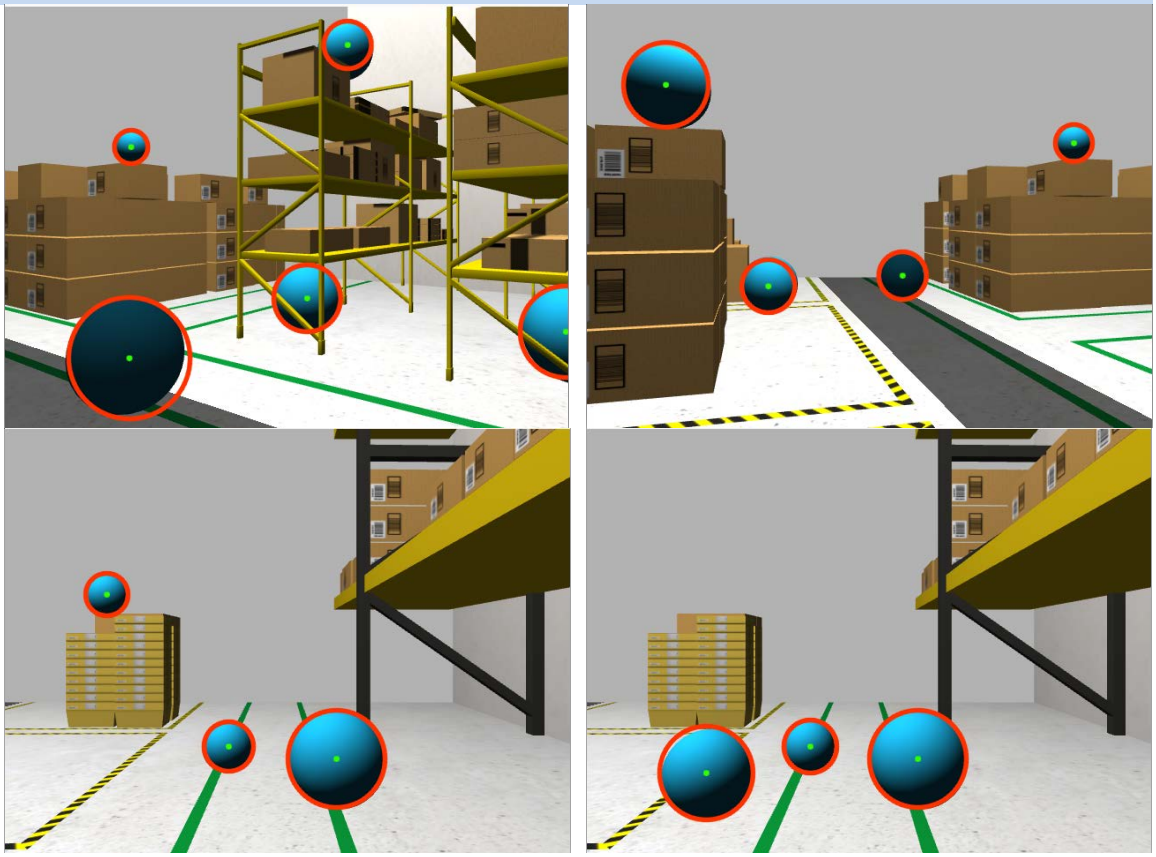
Para que los colores sean aleatorios pero tengan repetición, se puede utilizar la función **srand(0)** para generar números aleatorios, de manera que los colores utilizados para los contornos no varíen.

A continuación, se muestran varias imágenes de ejemplo para cada apartado:

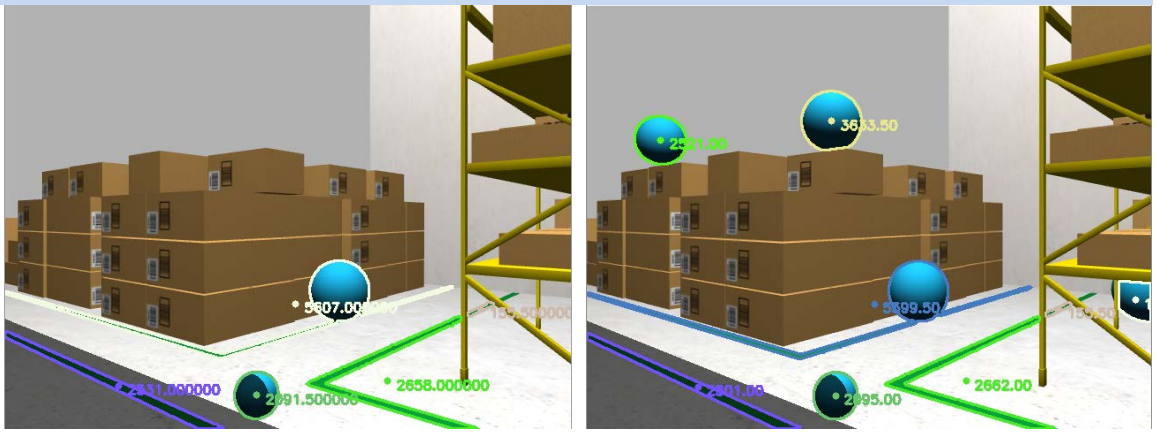
Opción 1



Opción 2



Opción 3



Ayuda

Funciones del Trackbar:
https://docs.opencv.org/3.4/d7/dfc/group__highgui.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