Práctica 3 – Geometría, Calibración y 3D

Este ejercicio tiene como objetivo trabajar en OpenCV con operaciones morfológicas, geometría, calibración y visión 3D. Todo ello visto en el **Tema 6: Operaciones morfológicas, Tema 7: Geometría y calibración** y **Tema 8: Visión 3D**.

Puntos totales posibles del ejercicio: 10

Instrucciones

Cada integrante del grupo debe acceder al siguiente enlace.

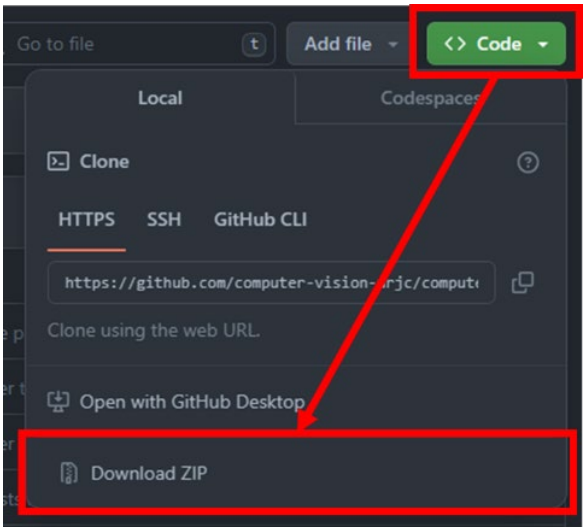
Si no lo ha hecho anteriormente, deberá asociarse con el usuario creado a partir de su correo electrónico de la **URJC**. Una vez hecho esto, deberá unirse al grupo de la asignatura. Si dicho grupo no existe, uno de los miembros deberá crearlo, utilizando el **mismo nombre** que figura en el **Aula Virtual**.

📌 La plantilla con el nodo **ROS 2** proporcionada deberá modificarse para que el nombre del paquete sea: **practica3-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

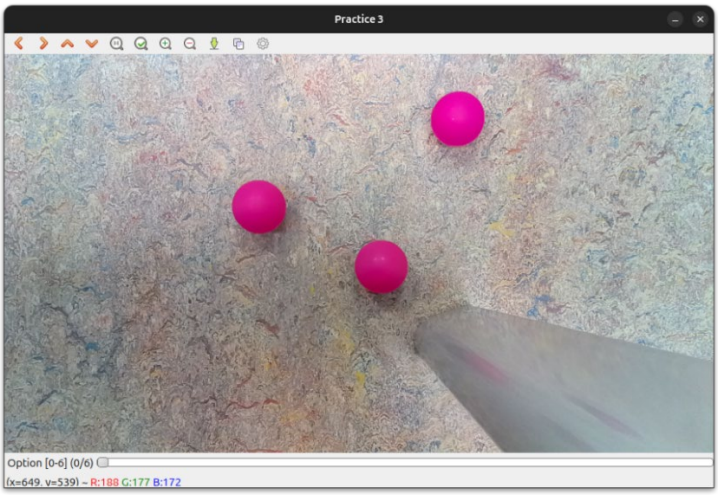
Enunciado

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 use_pointcloud:=false
```

Se pide crear un programa que trabaje con la imagen y muestre un control deslizante (*slider*) como el de la figura:

1. **Option**: que irá de 0 a 6.



Para cada una de las **7 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 **pulsaciones de teclado:**
  - Si la tecla pulsada es una **'c'**, se buscará el patrón del tablero de ajedrez para **guardar los puntos y calibrar la cámara**. Con cada pulsación, se acumularán los nuevos puntos a los ya existentes y se **calibrará** de nuevo.
  - Si se pulsa la tecla **'s'**, **se guardarán** tanto la matriz de **parámetros intrínsecos** como los **coeficientes de distorsión** en un **fichero**.
  - En caso de pulsar la tecla **'a'**, se comprobará si existe el fichero de calibración, y en caso afirmativo se aplicará la **corrección** correspondiente sobre la imagen. Al pulsar de nuevo la tecla 'a', se quitará la corrección y se mostrará la imagen original.

  En todos los casos, se tiene que mostrar la detección del tablero.
- **Opción 2:** se comprobará si existe el patrón con el tablero de ajedrez, y en caso afirmativo se utilizará la **función solvePnP para obtener las matrices de rotación y traslación** de la cámara con respecto al tablero. Utilizando estas matrices de **parámetros extrínsecos** junto a los **parámetros intrínsecos** de la cámara, utilizar la función **projectPoints** para proyectar un **cubo** en el **centro del tablero** y mostrar el resultado de la proyección.

- **Opción 3:** utilizando las **imágenes estéreo** en escala de grises **rectificadas** de la cámara, utilizar la función **stereoBM** para calcular el **mapa de disparidad** correspondiente y mostrarlo.
- **Opción 4:** utiliza las **imágenes a color y de profundidad** de la cámara para **generar la nube de puntos con PCL** mediante una **proyección de 2D a 3D**, y utiliza esa nube de puntos para realizar otra **proyección de 3D a 2D** que será la que se muestre como **imagen** en la ventana.
- **Opción 5: filtra** en la nube de puntos de **PCL** obtenida en el apartado 4 **los puntos** correspondientes a las **pelotas de color rosa**. Luego, **muestra** la **nube de puntos filtrada** en **RViz** y crea una **imagen** a partir de la **proyección de 3D a 2D** de la nube de puntos filtrada.
- **Opción 6: elimina** de la nube de puntos obtenida en el apartado 4, el **plano recto más grande** que exista mediante **PCL**, manteniendo únicamente los puntos que no pertenecen a dicho plano. Se deberá visualizar tanto la **nube de puntos filtrada en RViz** como la **proyección de 3D a 2D** de la nube filtrada en la imagen.

  Para llevar esto acabo, se recomienda **utilizar RANSAC** como en el siguiente ejemplo:

  https://pcl.readthedocs.io/projects/tutorials/en/latest/extract_indices.html

Para leer la tecla que se ha pulsado sobre la imagen, se debe usar una de las siguientes funciones:

```
// Read key
int key = cv::waitKey(int delay); // waiting time for a key pressed
int key = cv::pollKey(); // Polls for a pressed key.
```

Nota: Se valorará **hasta 1 punto extra**, la creación de una **figura más compleja** que la del cubo (Opción 2).

Ayuda

Funciones del Trackbar, waitKey, pollKey: enlace

PinHole model Camera: enlace

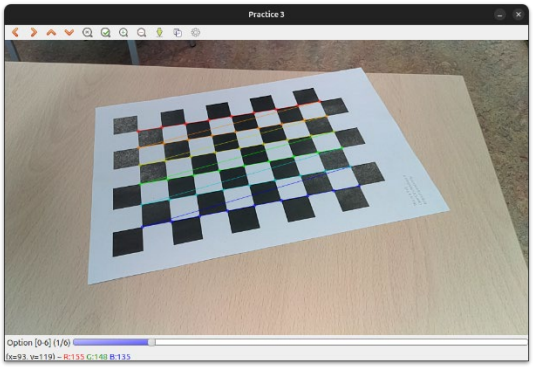
OpenCV stereoBM: enlace

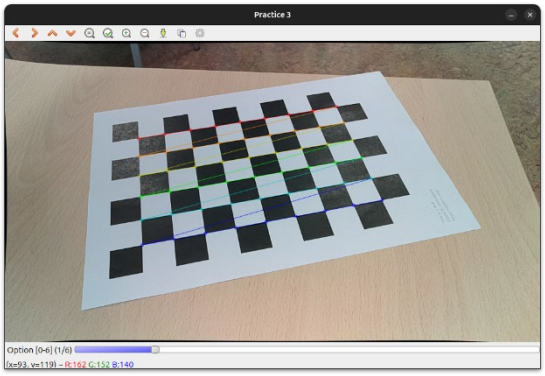
OpenCV projectPoints: enlace

OpenCV solvePNP: enlace1, enlace2

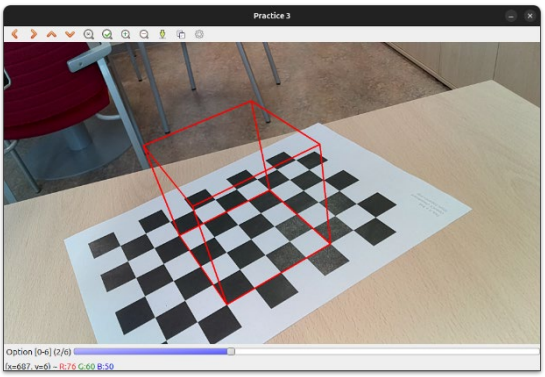
Imágenes

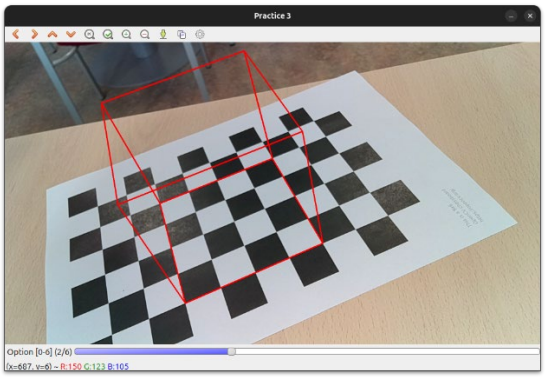
Opción 1



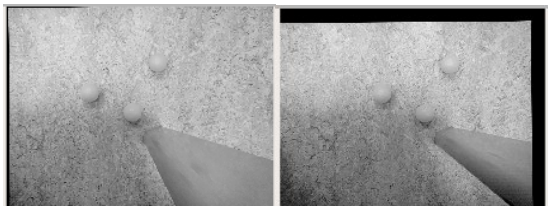


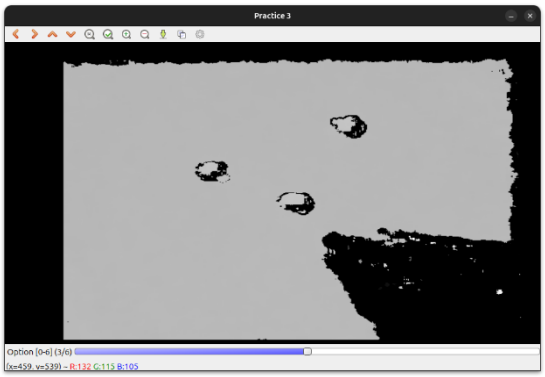
Opción 2



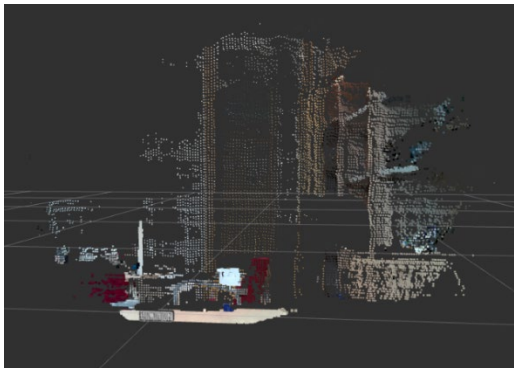


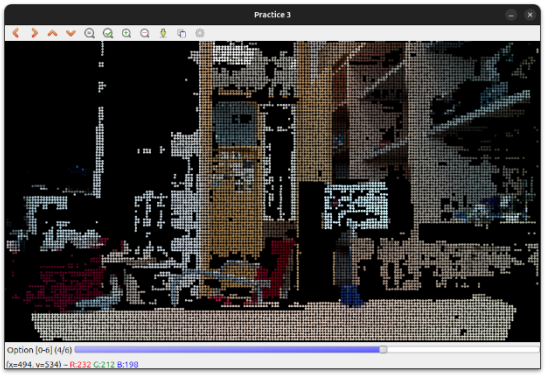
Opción 3



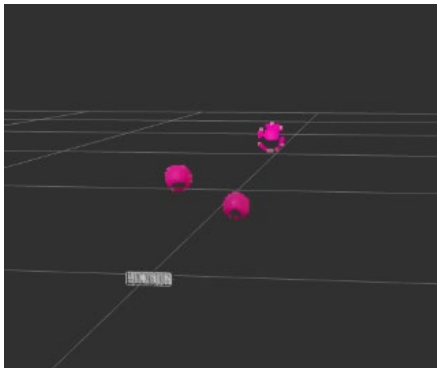


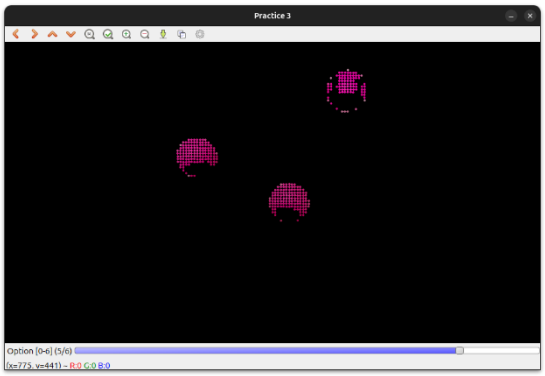
Opción 4



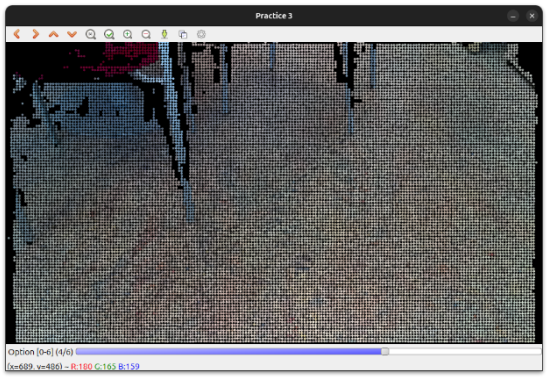


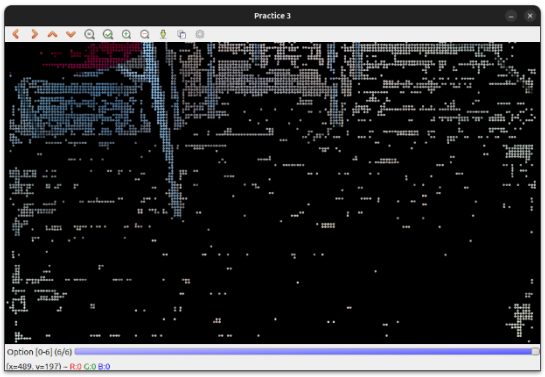
Opción 5

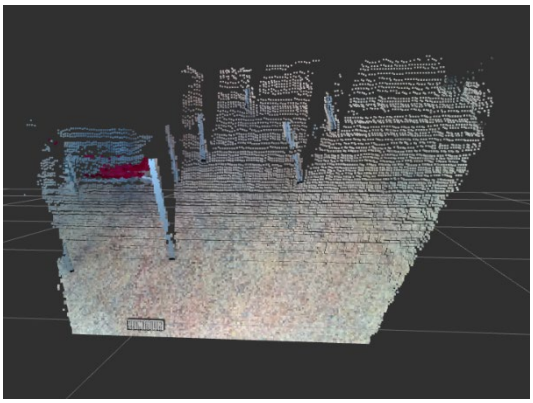


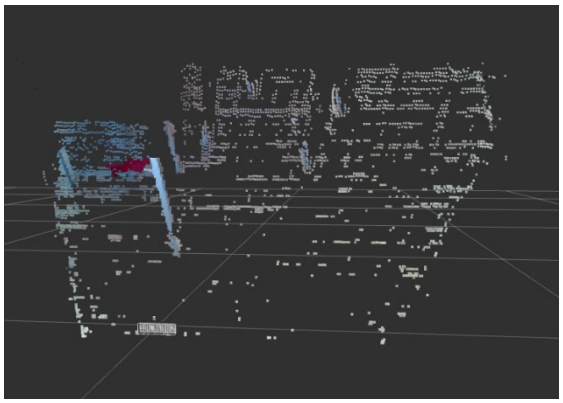


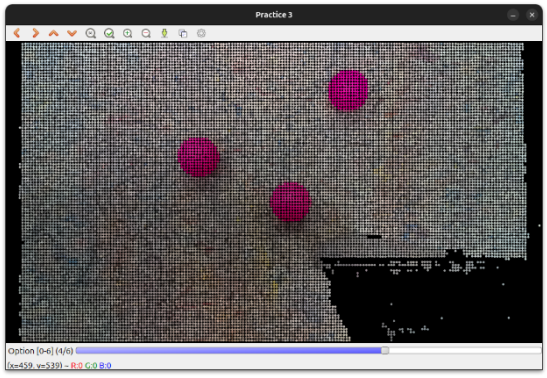
Opción 6

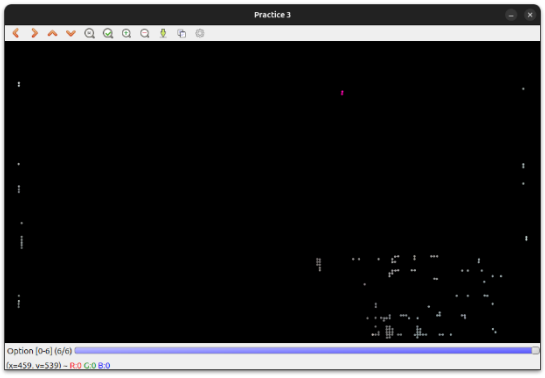


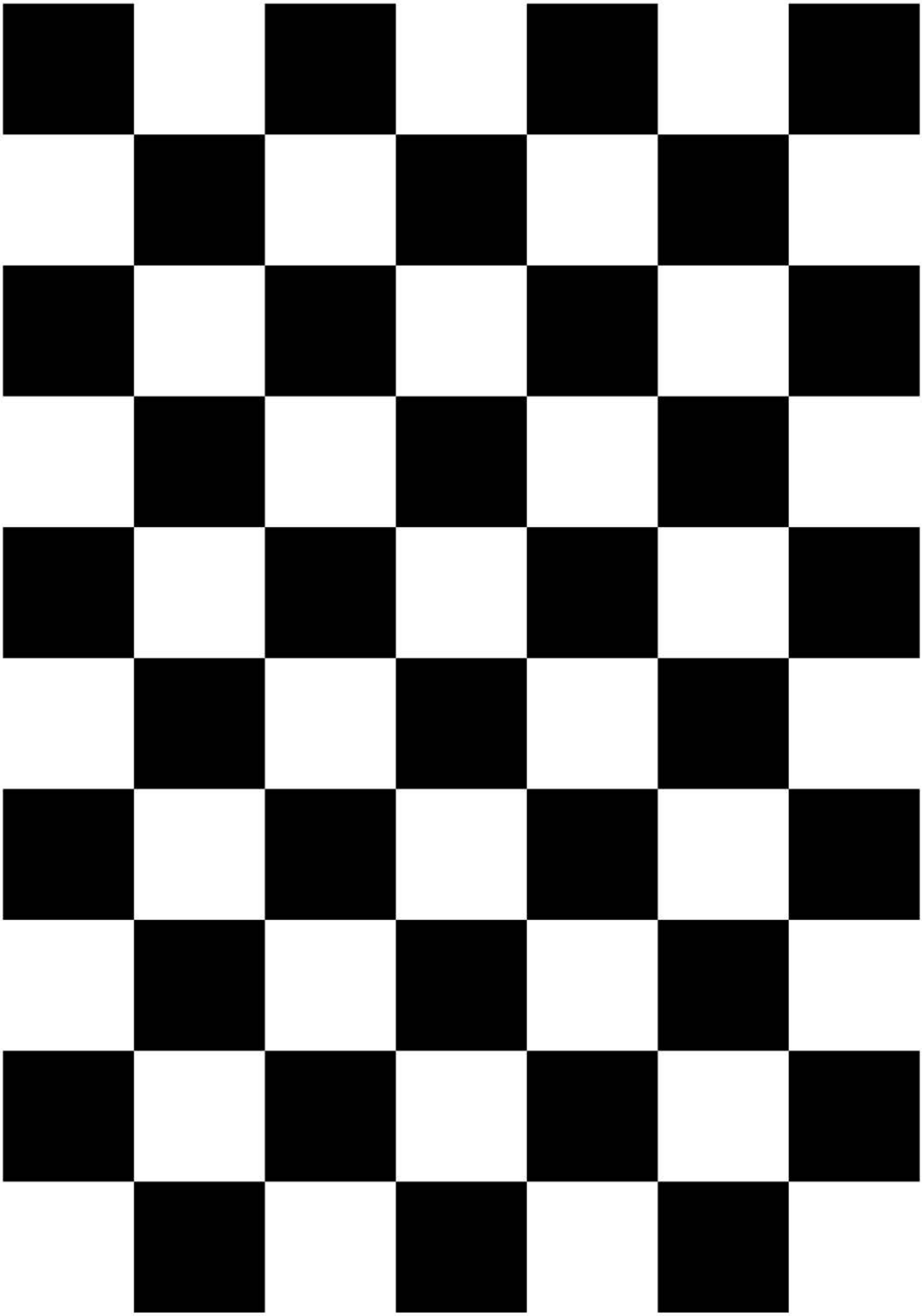












This is a 9x6
OpenCV chessboard
https://opencv.org/