PRÁCTICA DE CALIBRACIÓN CAMARA

**LISTA DE MATERIALES**

*Equipo extra:*

* Camara web con cable USB.
* Trípode.
* Tablero de ajedrez impreso blanco y negro.
* Códigos ArUco
* Carrito del laboratorio

*Estructuras mecánicas:*

* Perfil BOSCH de 20 mm
* Paquete de piezas impresas en 3D

*Tornillería:*

* Tornillos M5 x 7.
* Rondanas M5 x 7.
* Tuercas para ranuras o tipo T x 7.
* Tornillo M6
* Tuerca M6 x 2

*Cables:*

* Extensión USB hembra – macho min. 3 metros.

**INTRODUCCIÓN**

Información sobre cómo funciona una cámara de manera física, distancia focal, proyección de la imagen, etc.

**CAPÍTULO 0: INSTALACIÓN DE PYTHON Y LIBRERIAS**

Este capítulo explica cómo instalar Python, el entorno de desarrollo (IDE) y las librerías necesarias para llevar a cabo la calibración de la cámara.

1. Instalación de Python:

Para este proyecto se utilizará Python como lenguaje de programación. Primero, necesitas descargar Python desde su página oficial.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Accede al [sitio de descarga de Python](https://www.python.org/downloads/) y selecciona la versión correspondiente a tu sistema operativo.

1. Instalación del editor de código (IDE):

En este proyecto utilizamos Visual Studio Code (VS Code) como editor de código, pero puedes elegir cualquier otro editor que prefieras.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Si decides usar VS Code, descárgalo desde el siguiente enlace: [Descargar Visual Studio Code](https://code.visualstudio.com/Download).

1. Instalación de librerías necesarias:

Una vez instalado Python y el editor de código, será necesario instalar las siguientes librerías para trabajar con la calibración de la cámara. Las librerías principales que vamos a utilizar son:

* Open CV (librería y módulos “contrib”)
* numpy

Para instalarlas, abre la terminal y escribe lo siguiente:

*pip install opencv-contrib-python numpy*

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

2.- Escribe la siguiente línea:

1.- Seleccionar “Terminal”

**CAPÍTULO 1: ARMADO DE LA ESTRUCTURA DE LA CAMARA.**

Demostración con pasos de como armar la estructura de la cámara y como colocarla en su lugar.

**CAPÍTULO 2: TOMA DE FOTOS DEL TABLERO DE AJEDREZ PARA SU CALIBRACIÓN**

Una vez instalado los programas necesarios y ensamblada la estructura para la cámara, podemos iniciar la primera etapa de la calibración de la cámara web.

Imprime el patrón de ajedrez en blanco y negro (puede ser en cualquier tamaño de hoja). Lo crucial es el alto contraste entre los cuados. Se recomiendo que el tablero sea minimo de 8x7 cuadros (es decir, 7 x 6 esquinas internas).

Patrón de fondo

El contenido generado por IA puede ser incorrecto.

Del patrón, cuenta las esquinas internas de cada lado (los puntos de intersección). Además, mide longitud del lado de un cuadro del patrón en metros. Como se muestra en la siguiente imagen:

Imagen que contiene Icono

El contenido generado por IA puede ser incorrecto.

Con el patrón de ajedrez impreso y los datos dimensionales listos, se procede al código para la captura de las imágenes. Estas fotos se utilizarán para calcular la matriz intrínseca de la cámara.

El código permite capturar imágenes desde la cámara web presionando el botón de espacio. Cada vez que se toma una foto, esta se guarda en una carpeta llamada “Fotos\_Calibracion”, ubicada en la misma dirección que el script.

Código

import cv2

import Funciones\_Codigos as fc

# Se especifica la resolucion de la camara (se recomineda el valor de 1280x720)

resolucion\_width = 1280

resolucion\_height = 720

fc.guardar\_Resolucion\_Camara(resolucion\_width, resolucion\_height)

# Se crea la carpeta para guardar las fotos del tablero de ajedrez,

# se guardará en la direccion donde esta guardado este código

# (Por default el nombre de la carpeta es: 'Foto\_Calibracion').

fc.crear\_Carpeta\_Fotos()

# Iniciamos la cámara y ajustamos la resolución.

camara\_web = cv2.VideoCapture(1)

camara\_web.set(cv2.CAP\_PROP\_FRAME\_WIDTH, resolucion\_width)

camara\_web.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, resolucion\_height)

# Se verifica si se abrió la cámara, si no se sale del programa.

if not camara\_web.isOpened():

    print("No se puede abrir la cámara")

    exit()

contador\_fotos = 0

print("Presiona 'SPACE' para tomar una foto, 'ESC' para salir.")

while True:

    ret, imagen = camara\_web.read()

    if not ret:

        print("No se puede recibir imagen. Saliendo ...")

        break

    # Mostrar el video en una ventana

    cv2.imshow('Presiona SPACE para tomar foto', imagen)

    key = cv2.waitKey(1)

    if key == 27:  # ESC para salir

        break

    elif key == 32:  # SPACE para tomar foto

        fc.guardar\_Foto\_Calibracion(imagen, contador\_fotos)

        contador\_fotos += 1

# Liberar la cámara y cerrar ventanas

camara\_web.release()

cv2.destroyAllWindows()

print(f"Se guardaron {contador\_fotos} fotos en la carpeta: {fc.obtener\_Nombre\_Carpeta\_Fotos()} con una resolucion de {resolucion\_width}x{resolucion\_height}.")

Explicación del código

Importamos la librería de Open CV y también el módulo de funciones con funciones secundarias que ayudan al código.

import cv2

import Funciones\_Codigos as fc

Se define la resolución de la cámara, esto es importante dado que la calibración depende de estos valore. **¡ADVERTENCIA!** Si se modifica la resolución, la calibración existente dejará de ser válida y se deberá repetir el proceso.

# Se especifica la resolucion de la camara (se recomineda el valor de 1280x720)

resolucion\_width = 1280

resolucion\_height = 720

fc.guardar\_Resolucion\_Camara(resolucion\_width, resolucion\_height)

En estos puntos se crea la carpeta donde se van a guardar las fotos que se tomen y esta carpeta se va a guardar en la dirección donde esta guardado el código.

# Se crea la carpeta para guardar las fotos del tablero de ajedrez,

# se guardará en la direccion donde esta guardado este código

# (Por default el nombre de la carpeta es: 'Foto\_Calibracion').

fc.crear\_Carpeta\_Fotos()

Se inicializa la cámara y se ajusta la resolución para que este en los valores establecidos por ustedes. Después se evalúa si se puede abrir y si no se puede se sale del programa.

# Iniciamos la cámara y ajustamos la resolución.

camara\_web = cv2.VideoCapture(1)

camara\_web.set(cv2.CAP\_PROP\_FRAME\_WIDTH, resolucion\_width)

camara\_web.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, resolucion\_height)

 # Se verifica si se abrió la cámara, si no se sale del programa.

if not camara\_web.isOpened():

    print("No se puede abrir la cámara")

    exit()

Esta la parte principal del código la cual hace que se muestre a pantalla la imagen que está viendo la cámara para observar donde poner el tablero de ajedrez. Una vez presionado el botón de espacio se guardará la foto en la carpeta designada, podrás tomar las fotos que creas necesario y una vez que ya no quieras tomar más fotos se debe de presionar el botón de ESC.

print("Presiona 'SPACE' para tomar una foto, 'ESC' para salir.")

while True:

    ret, imagen = camara\_web.read()

    if not ret:

        print("No se puede recibir imagen. Saliendo ...")

        break

    # Mostrar el video en una ventana

    cv2.imshow('Presiona SPACE para tomar foto', imagen)

    key = cv2.waitKey(1)

    if key == 27:  # ESC para salir

        break

    elif key == 32:  # SPACE para tomar foto

        fc.guardar\_Foto\_Calibracion(imagen, contador\_fotos)

        contador\_fotos += 1

Una vez dado el botón de ESC se liberará la cámara y se cerrara la ventana, lo siguiente es una impresión a consola sobre la cantidad de fotos tomadas, en donde se guardaron y la resolución con la cual se tomaron las fotos.

# Liberar la cámara y cerrar ventanas

camara\_web.release()

cv2.destroyAllWindows()

print(f"Se guardaron {contador\_fotos} fotos en la carpeta: {fc.obtener\_Nombre\_Carpeta\_Fotos()} con una resolucion de {resolucion\_width}x{resolucion\_height}.")

RECOMENDACIONES CLAVE PARA LA CAPRUTA DE IMÁGENES:

* Mover el tablero por **todo** el rango de visión de la cámara: centro: esquinas, bordes. **Es fundamental variar su posición y orientación.**
* Toma fotos desde distintos **ángulos de inclinación** (no solo de frente), así como **acerca y aleja** el patrón para obtener diversas perspectivas.
* Asegura que **todas las esquinas internas** del tablero sean **visibles y nítidas** en la imagen, ya que son los puntos clave que utiliza el algoritmo de calibración.
* Utilizar un patrón **bien impreso**, con alto contraste, montado sobre una **superficie plana y rígida** para evitar curvaturas o arrugas.
* Las fotos deben de estar **bien enfocadas**. Evitar sombras, reflejos o brillos que puedan oscurecer u ocultar las esquinas.
* Se recomienda tomar **al menos 15 fotos**. Cuantas más imágenes diversas y de alta calidad se capturen, más precisa resultara la calibración.

**CAPÍTULO 3: OBTECIÓN DE LAS MATRICES DE CALIBRACION INTRISECA DE LA CAMARA Y EXPLICACIÓN.**

Ya con las fotos en la carpeta se puede pasar a la calibración intrínseca de la cámara, para esto se utiliza el siguiente código de Python.

Código

import cv2

import numpy as np

import Funciones\_Codigos as fc

tamaño\_patron = (7, 7) # Tamaño del patrón: número de esquinas internas (no cuadros)

longitud\_cuadro = 0.024 # Tamaño real de cada cuadrado (en metros)

# Preparar coordenadas del patrón (puntos 3D en el mundo real)

objp = np.zeros((tamaño\_patron[0] \* tamaño\_patron[1], 3), np.float32)

objp[:, :2] = np.mgrid[0:tamaño\_patron[0], 0:tamaño\_patron[1]].T.reshape(-1, 2)

objp \*= longitud\_cuadro # Escalar según el tamaño real de los cuadros

objpoints = [] # Puntos 3D en el mundo real

imgpoints = [] # Puntos 2D detectados en la imagen

# Filtrar archivos .jpg dentro de esa carpeta

imagenes = fc.filtrar\_Archivos\_JPG()

for fname in imagenes:

    img = cv2.imread(fname)

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    # Encontrar esquinas del tablero

    ret, esquinas = cv2.findChessboardCorners(gray, tamaño\_patron, None)

    if ret:

        objpoints.append(objp)

        imgpoints.append(esquinas)

        # Dibujar las esquinas encontradas

        cv2.drawChessboardCorners(img, tamaño\_patron, esquinas, ret)

        cv2.imshow('Detección', img)

        cv2.waitKey(100)

cv2.destroyAllWindows()

# Calibrar la cámara

ret, matriz\_camara, dist\_coeffs, rvecs, tvecs = cv2.calibrateCamera(

    objpoints, imgpoints, gray.shape[::-1], None, None )

print(f"Matriz de cámara (intrínseca): \n {matriz\_camara} \n")

print(f"Coeficientes de distorsión:\n {dist\_coeffs}")

print(f"Reporjection error: {ret}")

fc.guardar\_Parametros\_Camara(matriz\_camara, dist\_coeffs)

print("Se guardaron los parametros en el archivo json")

Explicación del código

Importamos las librerías necesarias las cuales son: open cv, numpy y la Funciones\_Codigo para partes secundarías del código.

import cv2

import numpy as np

import Funciones\_Codigos as fc

En esta sección se tienen que establecer los parámetros necesarios para la calibración. Estos parámetros son el numero de esquinas internas del patrón impreso y la longitud de un lado de un cuadro.

tamaño\_patron = (7, 7) # Tamaño del patrón: número de esquinas internas (no cuadros)

longitud\_cuadro = 0.024 # Tamaño real de cada cuadrado (en metros)

Esta sección es para preparar los puntos del patrón de ajedrez, para esto se crea un array de ceros del tamaño del múltiplo de las esquinas internas (Por ejemplo, el tablero antes creado sería una multiplicación de 7x7). De este array creamos una malla con todas las combinaciones de coordenadas de las esquinas del tablero, el cual será ajustará para que solo considere los puntos y , no se considera dado que en una imagen 2D solo hay dos dimensiones. Al final es escalar las coordenadas por el tamaño real de los cuadros, creando así un array con los puntos físicos de las esquinas internas del patrón de ajedrez.

# Preparar coordenadas del patrón (puntos 3D en el mundo real)

objp = np.zeros((tamaño\_patron[0] \* tamaño\_patron[1], 3), np.float32)

objp[:, :2] = np.mgrid[0:tamaño\_patron[0], 0:tamaño\_patron[1]].T.reshape(-1, 2)

objp \*= longitud\_cuadro # Escalar según el tamaño real de los cuadros

Se crea dos variables para los puntos en 3D (mundo real) y uno para 2D (la imagen). También se llama a la función *filtrar\_Archivos\_JPG* para guardar en la variable imágenes de todas las fotos tomadas del tablero ajedrez visto en el capítulo anterior.

objpoints = [] # Puntos 3D en el mundo real

imgpoints = [] # Puntos 2D detectados en la imagen

# Filtrar archivos .jpg dentro de esa carpeta

imagenes = fc.filtrar\_Archivos\_JPG()

En este *for* es donde se procesan las imágenes y hacemos que Open CV detecte las esquinas internas del patrón y las guarde en la variable esquinas, esto se logra gracias a la función findChessboardCorners. Esta función requiere de dos parámetros, la primera es la imagen, el segundo es el tamaño del patrón (número de esquinas internas). Lo que retorna es una booleano siendo *True* si se identificaron y ordenaron de manera correcta las esquinas. Lo segundo que retorna es una matriz con las coordenadas 2D en pixeles de las esquinas.

for fname in imagenes:

    img = cv2.imread(fname)

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    # Encontrar esquinas del tablero

    ret, esquinas = cv2.findChessboardCorners(gray, tamaño\_patron, None)

Si la función logra identificar las esquinas, se guarda la matriz de coordenadas 2D en la variable *imgpoints* y en la variable de *objpoints* se guarda las coordenadas de los puntos reales. Luego se muestra a pantalla las esquinas que detecto Open CV para revisión del usuario y confirmar que se haya hecho de manera correcta.

    if ret:

        objpoints.append(objp)

        imgpoints.append(esquinas)

        # Dibujar las esquinas encontradas

        cv2.drawChessboardCorners(img, tamaño\_patron, esquinas, ret)

        cv2.imshow('Detección', img)

        cv2.waitKey(100)

cv2.destroyAllWindows()

Una vez obtenidas todos los puntos 3D y 2D de la imagen pasamos a la calibración de la cámara, lo cual se hace con la función *calibrateCamera*. Este requiere de la matriz de puntos 3D, matriz de puntos 2D, y el tamaño de la imagen en pixeles (se usa *gray.shape[::-1]* para devolver las dimensiones de la imagen en escala de grises, en formato alto, ancho). Lo que nos devuelve un error de re-proyeccion, la matriz intrínseca, los coeficientes de distorsión, el vector de rotación y vector de translación. De estos solo vamos a usar el error de re-proyeccion, la matriz intrínseca y los coeficientes de distorsión.

# Calibrar la cámara

ret, matriz\_camara, dist\_coeffs, rvecs, tvecs = cv2.calibrateCamera(

    objpoints, imgpoints, gray.shape[::-1], None, None )

Se manda la imprimir la matriz, los coeficientes y el error. Para después, guardar estos parámetros en un archivo tipo .json, todo esto se hace en la función *guardar\_Parametros\_Camara* lo cual pide dos variables, la matriz intrínseca y los coeficientes de distorsión.

print(f"Matriz de cámara (intrínseca): \n {matriz\_camara} \n")

print(f"Coeficientes de distorsión:\n {dist\_coeffs}")

print(f"Reporjection error: {ret}")

fc.guardar\_Parametros\_Camara(matriz\_camara, dist\_coeffs)

print("Se guardaron los parametros en el archivo json")

Explicación de la matriz intrínseca, coeficientes de distorsión y re-proyeccion.

gfgdfgdfgdfgdfgdfgdfgdf

**CAPÍTULO 4: IMPRIMIR Y COLOCAR LOS CÓDIGOS ArUco Y SU DETECCIÓN.**

dfgdfgdfgd

**CAPÍTULO 5: OBTENCIÓN DE LA MATRIZ DE ROTACION Y VECTOR DE TRASLACION DEL WORLD A LA CAMARA.**