

# Laboratorio

# Calibración de Cámara

Visión por Ordenador I  
Ingeniería Matemática



# Laboratorio

## Calibración de Cámara

**Profesor:**

Erik Velasco      [evelasco@icai.comillas.edu](mailto:evelasco@icai.comillas.edu)  
Lionel Güitta      [lgllopez@icai.comillas.edu](mailto:lgllopez@icai.comillas.edu)  
Daniel Pinilla      [dpinilla@icai.comillas.edu](mailto:dpinilla@icai.comillas.edu)  
Luis Arias      [learias@icai.comillas.edu](mailto:learias@icai.comillas.edu)  
Rodrigo Sánchez      [rsmolina@icai.comillas.edu](mailto:rsmolina@icai.comillas.edu)  
Ignacio de Rodrigo      [iderodrigo@comillas.edu](mailto:iderodrigo@comillas.edu)

**Email**

Cover: Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA  
under CC BY-NC 2.0 (Modified)

# Contents

<b>1</b>	<b>Introducción al laboratorio</b>	<b>1</b>
1.1	Requisitos . . . . .	1
1.2	Metodología . . . . .	1
1.3	Materiales . . . . .	1
1.4	Entregas . . . . .	2
1.5	Evaluaciones . . . . .	2
1.6	Faltas al laboratorio . . . . .	2
<b>2</b>	<b>Sesión 1: Calibración de Cámara</b>	<b>3</b>
2.1	Materiales . . . . .	3
2.2	Apartados de la práctica . . . . .	3
2.3	Observaciones . . . . .	3
2.4	Qué va a aprender . . . . .	3
2.5	Evaluación . . . . .	4
<b>3</b>	<b>Apartado A</b>	<b>5</b>
	Tarea A.1: Carga de Imágenes . . . . .	5
	Tarea A.2: Detección de Esquinas . . . . .	5
	Tarea A.3: Comprobación de Detecciones . . . . .	6
	Tarea A.4: Obtención de Coordenadas 3D de las Esquinas del Patrón . . . . .	6
	Tarea A.5: Calibración de la Cámara Izquierda . . . . .	6
	Preguntas . . . . .	7
<b>4</b>	<b>Apartado B</b>	<b>8</b>
	Tarea B.1: Carga de Imágenes . . . . .	8
	Tarea B.2: Detección de esquinas de las imágenes . . . . .	9
	Tarea B.3: Obtención de coordenadas del tablero . . . . .	9
	Tarea B.4: Definición de parámetros para calibración . . . . .	9
	Tarea B.5: Calibración . . . . .	9
	Preguntas . . . . .	10

# Introducción al laboratorio

El laboratorio de *Visión por Ordenador I* es una oportunidad para poner en práctica los conocimientos teóricos de la asignatura. En este laboratorio se trabajará con técnicas de visión por ordenador clásica y se abarcarán los siguientes bloques:

- Sesión 1: Calibración de cámaras (2 horas).
- Sesión 2: Procesamiento de imagen (2 horas).
- Sesión 3: Extracción de características y bolsa de palabras visuales (2 horas).
- Sesión 4: Seguimiento de objetos (2 horas).

Además, las últimas 3 sesiones de laboratorio se dedicarán a la elaboración de un proyecto. Este contendrá los módulos trabajados durante el curso. El proyecto le ayudará a reforzar los conceptos estudiados en el aula y las sesiones de laboratorio. Puede encontrar inspiración en proyectos del curso anterior<sup>1</sup>.

## 1.1. Requisitos

Es fundamental que, antes de iniciar la sesión, cada persona disponga del lenguaje de programación Python instalado en su equipo. Además, debe tener disponible un IDE para trabajar de forma eficiente<sup>2</sup>. Si tiene problemas a la hora de realizar las instalaciones, por favor, comuníquese a uno de los profesores.

## 1.2. Metodología

El laboratorio se realiza en parejas. Se recomienda que ambas personas realicen la práctica (de forma colaborativa pero en ordenadores diferentes). Tenga en cuenta que los exámenes siempre contienen apartados dedicados a la elaboración de código, por lo que es importante que tenga soltura programando y consultando la documentación de las librerías.

## 1.3. Materiales

En algunas sesiones se trabajará con un Jupyter Notebooks (.ipynb), mientras que en otras se trabajará con scripts de Python (.py). En las prácticas finales se introducirá la Raspberry Pi junto con una cámara para obtener y procesar sus propias imágenes.

Los materiales relacionados con software podrá encontrarlos en la sección de Moodle dedicada al laboratorio. Se facilitarán los materiales antes de cada práctica. Por otro lado su profesor le facilitará el hardware necesario para trabajar con la RPi durante las sesiones finales.

<sup>1</sup>Proyecto Tangible UI en Github: <https://github.com/winoo19/wandavision>

<sup>2</sup>Un IDE muy extendido y recomendado es: VScode: <https://code.visualstudio.com/>

## **1.4. Entregas**

Las entregas se realizarán a través de la sección habilitada en Moodle para cada sesión de laboratorio. La entrega deberá realizarse, como tarde, una semana después de la sesión trabajada.

## **1.5. Evaluaciones**

En el guion de cada práctica encontrará una sección donde se especifique el peso de cada apartado. Así podrá saber cómo se ha evaluado su trabajo.

## **1.6. Faltas al laboratorio**

La persona que se ausente de una sesión de laboratorio de forma injustificada obtendrá un 0 como calificación de esa sesión.

# 2

## Sesión 1: Calibración de Cámara

### 2.1. Materiales

En esta práctica se trabajará con los siguientes recursos (puede encontrarlos en la sección de Moodle *Laboratorio/Sesión 1*):

- **lab\_1.ipynb**: notebook con el código que deberá completar.
- **left**: carpeta con imágenes de la cámara situada a la izquierda.
- **right**: carpeta con imágenes de la cámara situada a la derecha.
- **fisheye**: carpeta con imágenes de una cámara con lente ojo de pez.

### 2.2. Apartados de la práctica

La Sesión 1 del laboratorio está dividida en los siguientes apartados:

- Instalaciones: Instalación de librerías necesarias.
- Librerías: Importación de las librerías que se utilizan en el resto. Puede realizar la importación en cualquier celda de código, pero tenerlas concentradas en la celda inicial puede ayudar a mantener la organización del Notebook.
- Apartado A: Calibración de cámara (izquierda y derecha).
- Apartado B: Corrección de distorsión (ojo de pez).

### 2.3. Observaciones

Aunque el guion de la práctica y los comentarios en Markdown del Notebook estén escritos en español, observe que todo aquello que aparece en las celdas de código está escrito en inglés. Es una buena práctica que todo su código esté escrito en inglés.

Aquellas partes del código que deberá completar están marcadas con la etiqueta **TODO**.

Es muy importante que trabaje consultando la documentación de OpenCV<sup>1</sup> de Python para familiarizarse de cara al examen. Tenga en cuenta que en los exámenes no podrá utilizar herramientas de ayuda como Copilot.

### 2.4. Qué va a aprender

Al finalizar esta práctica, sabrá cómo obtener los parámetros intrínsecos y extrínsecos de una cámara, obteniendo así un modelo. También aprenderá a aplicar un modelo de cámara en un problema de distorsión de imagen. Además, se habrá familiarizado con la librería OpenCV<sup>2</sup> de Python.

---

<sup>1</sup>Documentación de OpenCV: <https://docs.opencv.org/4.x/index.html>

<sup>2</sup>OpenCV: <https://opencv.org/>

## 2.5. Evaluación

La nota que obtenga en esta sesión de laboratorio será la misma que obtenga su pareja. Los apartados de la práctica serán evaluados como refleja la Tabla 2.1

Tarea	Valor	Resultado
Pregunta A.1	2.5	
Pregunta A.2	2.5	
Pregunta A.3	2.5	
Pregunta B.1	2.5	
<b>Total</b>	<b>10.0</b>	

**Table 2.1:** Valoración de los apartados de la práctica.

# 3

## Apartado A

El objetivo del *Apartado A* es conseguir los parámetros intrínsecos y extrínsecos de las cámaras situadas a la derecha e izquierda de un sistema estereoscópico. En la Figura 3.1 se muestra un ejemplo de cámara estereoscópica<sup>1</sup> (no digital). Con estos parámetros se obtiene un modelo de cámara para cada subsistema.



Figure 3.1: Cámara estereoscópica.

En este apartado trabajará con el notebook y las imágenes de las carpetas *right* y *left*. En primer lugar, fíjese en la documentación de OpenCV del método `cv2.calibrateCamera()`<sup>2</sup>. Intente detectar los argumentos que requiere este método y cómo podrían derivarse de las imágenes.

Una vez hecho ese ejercicio mental, deberá seguir los pasos de este enunciado para conseguir un modelo de cámara a partir de las imágenes de cada carpeta:

### Tarea A.1: Carga de Imágenes

Defina y ejecute el método para cargar imágenes `load_images()`. Deberá pasar una lista que contenga las rutas de todas las imágenes de la carpeta *left*.

### Tarea A.2: Detección de Esquinas

Detecte las esquinas de los patrones usando `cv2.findChessboardCorners()`. Refine las detecciones con `cv2.cornerSubPix()`.

**Detección.** La función `cv2.findChessboardCorners()` de OpenCV busca el patrón de calibración en cada imagen y devuelve una tupla con dos elementos. El primer elemento es `False` si la detección del patrón falla y `True` si tiene éxito. El segundo elemento contiene las coordenadas de las esquinas del patrón, que solo son válidas si la detección fue exitosa (es decir, si el primer elemento de la tupla es `True`).

<sup>1</sup>De John Alan Elson - <http://www.3dham.com/>, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=8076697>  
<sup>2</sup>Doc. `cv2.calibrateCamera()` [https://docs.opencv.org/4.x/d9/d0c/group\\_\\_calib3d.html#ga687a1ab946686f0d85ae0363b5af1d7b](https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga687a1ab946686f0d85ae0363b5af1d7b)

Tenga en cuenta que los resultados de la llamada a esta función se almacenan en una lista, de manera que el elemento  $i$  de esa lista corresponde al resultado de procesar la imagen  $i$  cargada previamente.

**Refinamiento.** Se utiliza la función `cv2.cornerSubPix()` para refinar las detecciones de las esquinas. Tenga en cuenta que esta función necesita las imágenes de entrada en escala de grises.

**Nota:** Inspeccione las imágenes de forma manual y compruebe que el patrón está formado por 8 filas y 6 columnas de esquinas.

### Tarea A.3: Comprobación de Detecciones

Compruebe que las detecciones anteriores (tanto las iniciales como las refinadas) son correctas dibujando los resultados con `cv2.drawChessboardCorners()`.

Dibuje en las imágenes los puntos detectados por `cv2.findChessboardCorners()`. Por razones de eficiencia, la función utilizada modifica directamente las imágenes pasadas por parámetro en lugar de hacer una copia. Para evitar perder las imágenes originales, es mejor hacer una copia de ellas de antemano. Utilice la copia para realizar el ejercicio.

Una vez dibujadas las detecciones, deberá diseñar una función de visualización de imágenes con OpenCV. Para ello, podrá apoyarse en los métodos `cv2.imshow()`, `cv2.waitKey()` y `cv2.imwrite()`.

### Tarea A.4: Obtención de Coordenadas 3D de las Esquinas del Patrón

Defina y ejecute el método `get_chessboard_points(chessboard_shape, dx, dy)` que proporcione las coordenadas 3D de las esquinas del patrón.

Para ello, el punto de coordenadas  $[0, 0, 0]^\top$  es la primera esquina detectada en el patrón de calibración. El eje X corresponde al lado corto del patrón, mientras que el eje Y corresponde al lado largo. La función devolverá un array de numpy de tamaño  $N \times 3$  que contenga las coordenadas  $(x, y, z)$ , donde N es el número de esquinas.

Para diseñar la función, tenga en cuenta que `chessboard_shape` es una tupla que contiene el número de esquinas por fila y columna del patrón. Por otro lado, `dx` y `dy` corresponden al ancho y alto de cada cuadrado del patrón de calibración, que en esta práctica es de 30mm.

Tenga en cuenta que deberá proporcionar una lista que contenga los puntos para todas las imágenes con detecciones correctas.

### Tarea A.5: Calibración de la Cámara Izquierda

Utilice `cv2.calibrateCamera()` para obtener los parámetros de calibración para la cámara situada a la izquierda.

Deberá utilizar los resultados de `cv2.findChessboardCorners()` y el conjunto de puntos obtenido con `get_chessboard_points()` del ejercicio anterior.

Filtre aquellas detecciones de esquinas que hayan sido procesadas correctamente. Recuerde que `cv2.findChessboardCorners()` devuelve una tupla donde el primer elemento es un Booleano. Este elemento indica si la detección es correcta o no.

## Preguntas

### Pregunta A.1

Repita el proceso (carga de imágenes, detección, comprobación de esquinas, etc.) para la cámara situada a la derecha. Deberá adjuntar:

- Al menos 2 imágenes con la detección de esquinas.
- Los parámetros intrínsecos y extrínsecos de la cámara.
- Los coeficientes de distorsión y el error de calibración.

### Pregunta A.2

¿Observa diferencias entre las detecciones de esquinas realizadas con `cv2.findChessboardCorners()` y las refinadas con `cv2.cornerSubPix()`? Apoye su respuesta adjuntando imágenes.

### Pregunta A.3

Por ahora, ha asumido que el número de imágenes que le hemos ofrecido es adecuado para obtener una calibración adecuada. Sin embargo, en su proyecto final, ¿cuántas imágenes cree que debería tomar con su cámara para calibrarla? Responda a esta pregunta y obtenga un orden de magnitud dibujando un diagrama de Pareto. En este, aparecerá el número de imágenes consideradas (eje horizontal) vs. el error de calibración (eje vertical).

# 4

## Apartado B

En este apartado se trabajará en la corrección de la distorsión debido a lentes de ojo de pez. Primero se calibrará una cámara con este tipo de lente. Posteriormente se utilizarán estos parámetros de calibración para corregir la distorsión de una de las imágenes de calibración. La Figura 4.1 muestra un ejemplo (no digital) de este tipo de dispositivos<sup>1</sup>.



Figure 4.1: Lente ojo de pez.

Para este ejercicio, las imágenes de calibración se encuentran en la carpeta *fisheye*. De nuevo, se recomienda consultar la documentación de la función utilizada para calibrar y sus argumentos: `cv2.fisheye.calibrate()`<sup>2</sup>.

Una vez hecho ese ejercicio mental, deberá seguir los pasos de este enunciado para conseguir un modelo de cámara y corregir la distorsión:

### Tarea B.1: Carga de Imágenes

Reutilice el método `load_images()` para cargar las imágenes de la carpeta *fisheye*. En esta ocasión, puede apoyarse en el método `glob.glob('path_to_folder/*.jpg')` para obtener una lista con la ruta de todas las imágenes de interés.

<sup>1</sup>Morio, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons  
[https://commons.wikimedia.org/wiki/File:Fisheye-Nikkor\\_Auto\\_6mm\\_f2.8\\_lens\\_2015\\_Nikon\\_Museum.jpg](https://commons.wikimedia.org/wiki/File:Fisheye-Nikkor_Auto_6mm_f2.8_lens_2015_Nikon_Museum.jpg)

<sup>2</sup>Doc. `cv2.fisheye.calibrate()` [https://docs.opencv.org/4.x/db/d58/group\\_\\_calib3d\\_\\_fisheye.html#gad626a78de2b1dae7489e152a5a5a89e1](https://docs.opencv.org/4.x/db/d58/group__calib3d__fisheye.html#gad626a78de2b1dae7489e152a5a5a89e1)

## Tarea B.2: Detección de esquinas de las imágenes

Utilice los métodos `cv2.findChessboardCorners()` y `cv2.cornerSubPix()` para obtener las esquinas del tablero de todas las imágenes. Recuerde que para utilizar `cv2.cornerSubPix()` deberá pasar las imágenes a escala de grises y definir `subpix_criteria`.

Asegúrese de que el output de esta tarea contenga solo los casos en los que las detecciones son adecuadas.

## Tarea B.3: Obtención de coordenadas del tablero

Reutilice el método `get_chessboard_points()` para obtener las coordenadas del nuevo tablero. Para obtener el número de filas y columnas, abra de forma manual una de las imágenes. La longitud de cada cuadrado en los patrones es de 30 mm.

De nuevo, tenga en cuenta que deberá proporcionar una lista que contenga los puntos para todas las imágenes con detecciones correctas.

**Nota:** La función de calibración espera que cada vector asociado a cada imagen tenga el siguiente formato:  $(n, 1, 3)$ , siendo  $n$  el número de puntos detectados.

## Tarea B.4: Definición de parámetros para calibración

Defina los parámetros necesarios para llamar a la función `cv2.fisheye.calibrate()`:

- `calibration_flags`: Configura el método de calibración.
- `intrinsics`: Matriz de intrínsecos.
- `distortion`: Vector con coeficientes de distorsión.
- `rotations`: Vectores de rotación.
- `traslations`: Vectores de translación.

## Tarea B.5: Calibración

Utilice el método `cv2.fisheye.calibrate()` y visualice el valor de `intrinsics`, `distortion`.

## Preguntas

### Pregunta B.1

Una vez calibrada la cámara con lente de ojo de pez, tiene acceso a la matriz de intrínsecos y los coeficientes de distorsión. Con estos valores y con ayuda de las funciones `cv2.fisheye.initUndistortRectifyMap()` y `cv2.remap()` podrá eliminar la distorsión de las imágenes tomadas por este tipo de cámaras. En el notebook encontrará ayuda para trabajar con la función `cv2.fisheye.initUndistortRectifyMap()`. Sin embargo, en el caso de `cv2.remap()`<sup>3</sup> deberá utilizar la documentación para obtener la imagen solicitada. Para obtener toda la puntuación de este apartado, facilite el valor de la matriz de intrínsecos y los coeficientes de distorsión de la cámara. Además, adjunte las imágenes sin distorsión de las 2 primeras imágenes de la carpeta *fisheye*.

<sup>3</sup>Doc. `cv2.remap()` [https://docs.opencv.org/4.x/da/d54/group\\_\\_imgproc\\_\\_transform.html#gab75ef31ce5cdfb5c44b6da5f3b908ea4](https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#gab75ef31ce5cdfb5c44b6da5f3b908ea4)