

# Laboratorio

## Extracción de características y bolsa de palabras visuales

Visión por Ordenador I  
Ingeniería Matemática

Universidad Pontificia Comillas, ICAI



**COMILLAS**  
UNIVERSIDAD PONTIFICIA

ICAI

# Laboratorio

## Extracción de características y bolsa de palabras visuales

**Profesor:**

Erik Velasco      [evelasco@icai.comillas.edu](mailto:evelasco@icai.comillas.edu)  
Lionel Güitta      [lglopez@icai.comillas.edu](mailto:lglopez@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:

Flying aboard Voyagers 1 and 2 are identical "golden" records, carrying the story of Earth far into deep space. This gold aluminum cover was designed to protect the gold-plated records from micrometeorite bombardment, but also serves a double purpose in providing the finder a key to playing the record. Image Credit: NASA/JPL-Caltech

# Contents

<b>1 Sesión 3: Extracción de características y bolsa de palabras visuales</b>	<b>1</b>
1.1 Materiales . . . . .	1
1.2 Apartados de la práctica . . . . .	1
1.3 Observaciones . . . . .	1
1.4 Qué va a aprender . . . . .	2
1.5 Evaluación . . . . .	2
<b>2 Apartado A: Detección de esquinas</b>	<b>3</b>
Tarea A.1: Creación de carpeta de resultados . . . . .	3
Tarea A.2: Carga de imágenes . . . . .	3
Tarea A.3: Definición del método Harris para detección de esquinas . . . . .	3
Tarea A.4: Aplicación de Harris . . . . .	3
Tarea A.5: Definición del método Shi-Tomasi para detección de esquinas . . . . .	4
Tarea A.6: Aplicación de Shi-Tomasi . . . . .	4
Preguntas . . . . .	4
<b>3 Apartado B: Detección de líneas rectas</b>	<b>5</b>
Tarea B.1: Aplicación de Canny . . . . .	5
Tarea B.2: Visualización de líneas . . . . .	5
Tarea B.3: Aplicación de transformada de Hough . . . . .	5
Preguntas . . . . .	6
<b>4 Apartado C: Detección de puntos de interés y Bolsa de palabras</b>	<b>7</b>
C.1: Detección de puntos de interés . . . . .	7
Tarea C.1.1: Filtro gaussiano con OpenCV . . . . .	7
Tarea C.1.2: Generación de espacio de escalas con imágenes gaussianas . . . . .	8
Tarea C.1.3: Generación de diferencias de gaussianas . . . . .	8
Tarea C.1.4: Evaluación de extremos . . . . .	8
Tarea C.1.5: Localización de puntos de interés y orientación de los mismos . . . . .	9
Tarea C.1.6: <i>Pipeline</i> de generación de puntos de interés y descriptores . . . . .	9
Tarea C.1.7: Correspondencia de características entre imágenes . . . . .	10
Preguntas . . . . .	10
C.2: Bolsa de palabras . . . . .	11
Tarea C.2.1: Carga de <i>datasets</i> de entrenamiento y validación . . . . .	11
Tarea C.2.2: Extracción de los descriptores . . . . .	11
Tarea C.2.3: Creación del vocabulario . . . . .	11
Tarea C.2.4: Entrenamiento del clasificador . . . . .	11
Tarea C.2.5: Inferencia en el dataset de entrenamiento . . . . .	11
Tarea C.2.6: Inferencia en el dataset de validación . . . . .	11
Preguntas . . . . .	12

# Sesión 3: Extracción de características y bolsa de palabras visuales

## 1.1. Materiales

En esta práctica se trabajará con los siguientes recursos (puede encontrarlos en la sección de Moodle *Laboratorio/Sesión 3*):

- **partA\_to-do.ipynb**: notebook con el código que deberá desarrollar en el apartado A.
- **partB\_to-do.ipynb**: notebook con el código que deberá desarrollar en el apartado B.
- **partC\_to-do.ipynb**: notebook con el código que deberá desarrollar en el apartado C.
- **helpers**: Archivos con métodos que le ayudarán a obtener los resultados.
  - `image_classifier.py`
  - `bow.py`
  - `dataset.py`
  - `results.py`
  - `utils.py`
- **data**: carpeta con imágenes para trabajar durante la práctica. Tenga en cuenta que, además, deberá descargar de Moodle el archivo `dataset.zip` (que deberá descomprimir) para trabajar en el Apartado C.

## 1.2. Apartados de la práctica

La Sesión 3 del laboratorio está dividida en los siguientes apartados:

- Librerías: Importación de las librerías que se utilizan en la sesión. Se recomienda realizar la importación en una celda inicial para mantener la organización del Notebook.
- Apartado A: Detección de esquinas.
- Apartado B: Detección de líneas rectas.
- Apartado C: Detección de puntos de interés (C1) y Bolsa de palabras (C2).

## 1.3. 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> para familiarizarse de cara al examen. Tenga en cuenta que en los exámenes no podrá utilizar herramientas de ayuda como Copilot.

## 1.4. Qué va a aprender

Al finalizar esta práctica, además de detectar características como esquinas y líneas rectas, el alumno aprenderá a construir un diccionario visual a partir de características de varias imágenes, a representar imágenes mediante histogramas de palabras visuales y a clasificar y comparar imágenes basándose en sus características.

## 1.5. 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 1.1. Tenga en cuenta que en esta práctica la pareja con el mejor valor de *accuracy* en el apartado *Bolsa de palabras* obtendrá 1 punto extra.

Tarea	Valor	Resultado
Pregunta A.1	1.5	
Pregunta A.2	1.5	
Pregunta B.1	2.0	
Pregunta C.1	1.5	
Pregunta C.2.A	1.5	
Pregunta C.2.B	2.0	
Pregunta Extra C.2.C	1.0	
<b>Total</b>	<b>11.0</b>	

**Table 1.1:** Valoración de los apartados de la práctica.

---

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

# 2

## Apartado A: Detección de esquinas

### Tarea A.1: Creación de carpeta de resultados

Cree una nueva carpeta llamada `partA`, dentro de la carpeta `data`, con el objetivo de presentar en ella los resultados de esta parte de la práctica.

### Tarea A.2: Carga de imágenes

Defina y ejecute los dos métodos propuestos para cargar imágenes `imageio_load_images()` y `opencv_load_images()`. Observe lo que ocurre al guardar ambas imágenes usando la misma función `cv2.imwrite()`.

### Tarea A.3: Definición del método Harris para detección de esquinas

Defina la función `harris_corner_detector()`, que servirá para detectar las esquinas de las imágenes de trabajo. Siga los siguientes pasos:

- Convertir la imagen de entrada a escala de grises usando `cv2.cvtColor()`.
- Transformar la imagen en escala de grises al tipo `float32` con `np.float32`.
- Aplicar el método `cv2.cornerHarris()`<sup>1</sup>.
- Dilatar la imagen resultante con `cv2.dilate` para mejorar la visibilidad de las esquinas detectadas.
- Establecer un umbral para resaltar las esquinas detectadas.

### Tarea A.4: Aplicación de Harris

Aplique la función `harris_corner_detector()` sobre las imágenes de trabajo. Asegúrese de que las imágenes quedan guardadas como se especifica en los comentarios. La Figura 2.1 muestra un ejemplo con el resultado que debería obtener al aplicar el método Harris.



Figure 2.1: Ejemplo de detección de esquinas con el método Harris.

<sup>1</sup>Documentación del método Harris en OpenCV:  
[https://docs.opencv.org/3.4/dd/d1a/group\\_\\_imgproc\\_\\_feature.html#gac1fc3598018010880e370e2f709b4345](https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#gac1fc3598018010880e370e2f709b4345)

## Tarea A.5: Definición del método Shi-Tomasi para detección de esquinas

Defina la función `shi_tomasi_corner_detection()`, que servirá para la detección de esquinas sobre las imágenes de trabajo. Siga los siguientes pasos:

- Convertir la imagen de entrada a escala de grises utilizando `cv2.cvtColor()`.
- Aplicar el método `cv2.goodFeaturesToTrack()`<sup>2</sup>.
- Convertir las coordenadas de las esquinas detectadas a enteros utilizando `np.intp`.
- Para cada esquina detectada, extraer las coordenadas `x` e `y` y dibujar un círculo en la imagen usando `cv2.circle()`.

## Tarea A.6: Aplicación de Shi-Tomasi

Aplique la función `shi_tomasi_corner_detection()` sobre las imágenes de trabajo. Asegúrese de que las imágenes quedan guardadas como se especifica en los comentarios.

### Preguntas

#### Pregunta A.1

Realice la detección de esquinas en las otras dos imágenes de la carpeta `data/source` (cuyos nombres de guardado han de ser `sudoku` y `tennis`) con el método de Harris.

#### Pregunta A.2

Realice la detección de esquinas en las otras dos imágenes de la carpeta `data/source` (cuyos nombres de guardado han de ser `sudoku` y `tennis`) con el método de Shi-Tomasi.

---

<sup>2</sup>Documentación del método `goodFeaturesToTrack()` en OpenCV:  
[https://docs.opencv.org/3.4/dd/d1a/group\\_\\_imgproc\\_\\_feature.html#ga1d6bb77486c8f92d79c8793ad995d541](https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga1d6bb77486c8f92d79c8793ad995d541)

# 3

## Apartado B: Detección de líneas rectas

### Tarea B.1: Aplicación de Canny

Para obtener los bordes de las imágenes, aplique el método `cv2.Canny()`<sup>1</sup> de OpenCV a las imágenes de trabajo ajustando los hiperparámetros.

Recuerde que para aplicar Canny, primero deberá pasar su imagen a escala de grises con el método `cvtColor()`. Por otro lado, observe que el método Canny devuelve una imagen binaria (blanco y negro) con los bordes detectados.

### Tarea B.2: Visualización de líneas

Implemente la función `draw_lines()` para pintar las líneas detectadas sobre las imágenes. Para ello, puede utilizar el método `cv2.line()`<sup>2</sup> de OpenCV.

### Tarea B.3: Aplicación de transformada de Hough

Aplique el método `cv2.HoughLinesP()`<sup>3</sup> de OpenCV a las imágenes de trabajo y afine los hiperparámetros para extraer las líneas de interés. La Figura 3.1 muestra el resultado esperado al aplicar la transformada de Hough sobre una de las imágenes.



Figure 3.1: Ejemplo de detección de líneas con el método Hough.

<sup>1</sup>Documentación del método `cv2.Canny()` en OpenCV:  
[https://docs.opencv.org/3.4/dd/d1a/group\\_imgproc\\_feature.html#ga04723e007ed888ddf11d9ba04e2232de](https://docs.opencv.org/3.4/dd/d1a/group_imgproc_feature.html#ga04723e007ed888ddf11d9ba04e2232de)

<sup>2</sup>Documentación del método `cv2.line()` en OpenCV:  
[https://docs.opencv.org/3.4/d6/d6e/group\\_imgproc\\_draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2](https://docs.opencv.org/3.4/d6/d6e/group_imgproc_draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2)

<sup>3</sup>Documentación del método `cv2.HoughLinesP()` en OpenCV:  
[https://docs.opencv.org/3.4/dd/d1a/group\\_imgproc\\_feature.html#ga8618180a5948286384e3b7ca02f6feeb](https://docs.opencv.org/3.4/dd/d1a/group_imgproc_feature.html#ga8618180a5948286384e3b7ca02f6feeb)

## Preguntas

### Pregunta B.1

Repita el procedimiento para extraer las líneas de las dos imágenes restantes.

# 4

## Apartado C: Detección de puntos de interés y Bolsa de palabras

En este apartado deberá extraer los puntos de interés de una de las imágenes de la carpeta data. Una vez hecho, podrá comprobar la correspondencia con esa imagen alterada y la relación entre las mismas (Figura 4.1). Tras entender la funcionalidad de este método, pasará a obtener una bolsa de palabras a partir de un *dataset* de entrenamiento y generar predicciones para un *dataset* de evaluación.

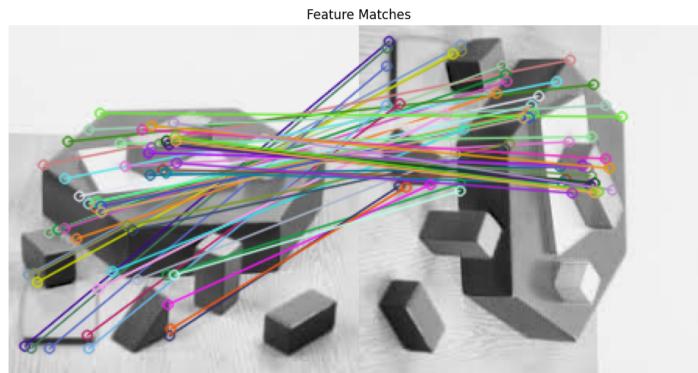


Figure 4.1: Correspondencia de características entre una imagen y la misma alterada con OpenCV.

### C.1: Detección de puntos de interés

En esta sección se generarán las funciones necesarias que cubran la detección de puntos de interés y los descriptores para las imágenes empleando la transformación SIFT. La transformación SIFT consta de cuatro pasos principales:

1. Detección de extremos en espacio de escalas.
2. Localización de keypoints.
3. Asignación de orientaciones.
4. Generación de los descriptores de keypoints.

Cada uno se irá desarrollando con las funciones necesarias para completarlos. De esta forma se comenzará generando el espacio de escalas en el que se evaluarán los extremos.

#### Tarea C.1.1: Filtro gaussiano con OpenCV

Para generar el espacio de escalas que se desea analizar, se debe aplicar un filtrado gaussiano a la imagen original hasta obtener un número de imágenes gaussianas relevante. Para ello, debe desarrollar

la función `generateGaussianImages()` empleando el método `cv2.GaussianBlur()`<sup>1</sup> de la librería OpenCV.

Como entrada, se recibe una imagen, y una lista de sigmas que deberá aplicar en el filtrado. Deberá generar tantas imágenes gaussianas como sigmas haya en la lista aplicando el filtro al resultado del filtrado anterior o a la imagen original en primera instancia.

### Tarea C.1.2: Generación de espacio de escalas con imágenes gaussianas

Deberá cargar la imagen base de la carpeta de data con `cv2` y en escala de grises sin emplear la función `cv2.cvtColor()`. A continuación, se definen los parámetros iniciales del espacio de escalas, la *sigma* inicial y el número de diferencias gaussianas que se desea generar. Cabe destacar que aunque el mínimo de gaussianas necesarias son 3 para detectar los máximos y mínimos en las ventanas de  $3 \times 3 \times 3$  como se verá más adelante, es posible que no sea suficiente este valor para detectar esos puntos de interés. Por último, emplee el método `generateGaussianImages()` que ha desarrollado en el apartado anterior.

Como puede observar en el código, se le proporciona el método `generateGaussianSigmas()` que recibe la *sigma* inicial y el número de diferencias gaussianas que desea generar. Este método devuelve una lista de valores de *sigma* que deberá aplicar en los filtrados consecutivos para la generación de las imágenes gaussianas. Si quiere generar  $n$  diferencias de gaussianas, el método le devolverá  $n+1$  valores de sigma. La Figura 4.2 muestra un ejemplo del resultado esperado de este apartado.

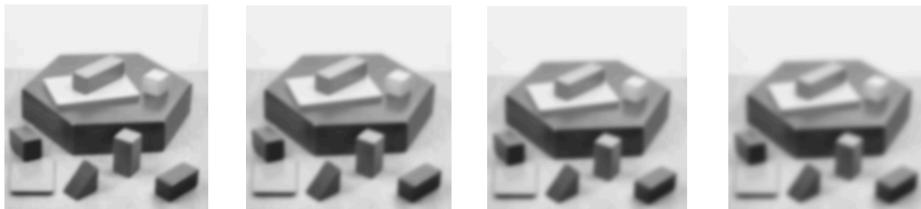


Figure 4.2: Resultados esperados de la generación de imágenes gaussianas

### Tarea C.1.3: Generación de diferencias de gaussianas

A partir de la lista de imágenes anteriores, se deberá generar una nueva lista con las diferencias entre las imágenes consecutivas empleando el método `subtract()`<sup>2</sup> de la librería OpenCV. La Figura 4.3 muestra un ejemplo del resultado esperado de este apartado.

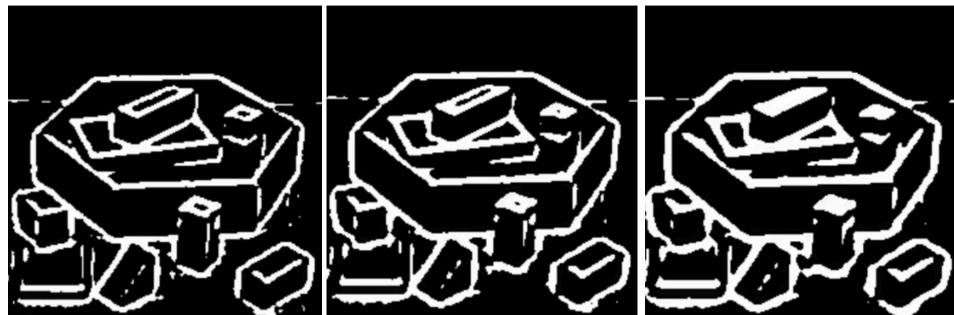


Figure 4.3: Resultados esperados de la generación de diferencias gaussianas

### Tarea C.1.4: Evaluación de extremos

En esta tarea debe desarrollar la función `isPixelAnExtremum()` que evalúa si el pixel central de una ventana de  $3 \times 3 \times 3$  es un máximo o mínimo entre sus vecinos y si es superior a un umbral mínimo en

<sup>1</sup> Documentación del método `cv2.GaussianBlur()` en OpenCV:  
[https://docs.opencv.org/3.4/d4/d86/group\\_\\_imgproc\\_\\_filter.html#gaabe8c836e97159a9193fb0b11ac52cf1](https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1)

<sup>2</sup> Documentación del método `cv2.subtract()` en OpenCV:  
[https://docs.opencv.org/3.4/d2/de8/group\\_\\_core\\_\\_array.html#gaa0f00d98b4b5edeaeb7b8333b2de353b](https://docs.opencv.org/3.4/d2/de8/group__core__array.html#gaa0f00d98b4b5edeaeb7b8333b2de353b)

valor absoluto. La ventana está compuesta por 3 regiones de 3x3 píxeles cada una que corresponden a regiones de 3 diferencias de gaussianas consecutivas. La función debe devolver *True* en caso de ser un máximo o mínimo y *False* en cualquier otro caso. **Importante:** Tras el filtrado gaussiano, los píxeles pueden tener valores negativos.

### Tarea C.1.5: Localización de puntos de interés y orientación de los mismos

Empleando la función del apartado anterior deberá completar el desarrollo de la función `findScaleSpaceExtrema()`.

Primero tendrá que completar el bucle `for` para obtener en cada iteración 3 diferencias de gaussianas consecutivas y enumerarlas. Tras esto, iterar sobre las coordenadas de estas teniendo en cuenta que deberá mover la ventana de 3x3 a través de toda la imagen y no puede quedar ningún punto de esta ventana fuera de la misma. Complete la llamada a la función `isPixelAnExtremum()` con los argumentos pertinentes para evaluar si el pixel en las coordenadas de esa iteración es un punto extremo.

Si los píxeles son máximos o mínimos, se tiene una localización aproximada de los puntos de interés de esta imagen. La función `localizeExtremumViaQuadraticFit()` (ya proporcionada) refina esta localización teniendo en cuenta el gradiente y el hessiano; descartando los valores que no tengan el suficiente contraste. Con la localización final se obtiene la orientación del punto de interés gracias a la función `computeKeypointsWithOrientations()` (de nuevo, proporcionada). Del resultado de esta función se tiene el punto de interés definido completamente.

Se recomienda el uso de la función de borrado de duplicados (`removeDuplicateKeypoints()`, también proporcionada) ya que existen puntos de interés que se repiten en el resultado del proceso anterior. Se proporciona al alumno la función `visualizeKp()` que le permitirá visualizar los puntos de interés en la imagen original. La Figura 4.4 muestra el resultado que debería obtener.

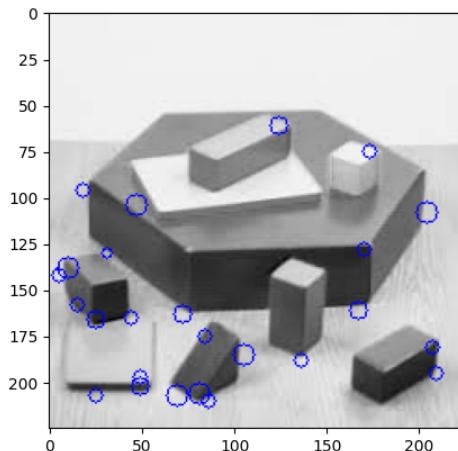


Figure 4.4: Resultados esperados de la detección de puntos de interés y orientación

Una vez obtenidas las imágenes gaussianas y los puntos de interés con sus orientaciones, se pueden generar los descriptores de las regiones vecinas a estos, que son invariantes a los cambios en la iluminación y el punto de vista 3D. En esta sesión, esto se proporciona al alumno con la función `generateDescriptors()`.

### Tarea C.1.6: Pipeline de generación de puntos de interés y descriptores

En esta tarea el alumno deberá completar la función `computeKeypointsAndDescriptors()`. Esta función recibe la imagen original, el valor inicial de *sigma* y el número de diferencias de gaussianas que se deseé generar. La tarea del alumno es completar la función con los métodos desarrollados en apartados anteriores para obtener el *pipeline* completo de localización de puntos de interés y generación de descriptores.

### Tarea C.1.7: Correspondencia de características entre imágenes

En esta tarea, el alumno comprobará la capacidad de sistema de localizar las mismas características en una imagen alterada para luego establecer la correspondencia entre las características de la imagen original y la de la alterada. Esto se conseguirá cargando la imagen original y la alterada en escala de grises con la librería cv2 de OpenCV. Después tendrá que hacer las llamadas pertinentes al método desarrollado en el apartado anterior para obtener los puntos de interés y descriptores de ambas imágenes.

Con estos datos, empleando la función `matchFeatures()` comprobará la capacidad de establecer la correspondencia entre los puntos de interés de ambas imágenes. La Figura 4.5 muestra los resultados de la correspondencia de características entre imágenes transformadas.

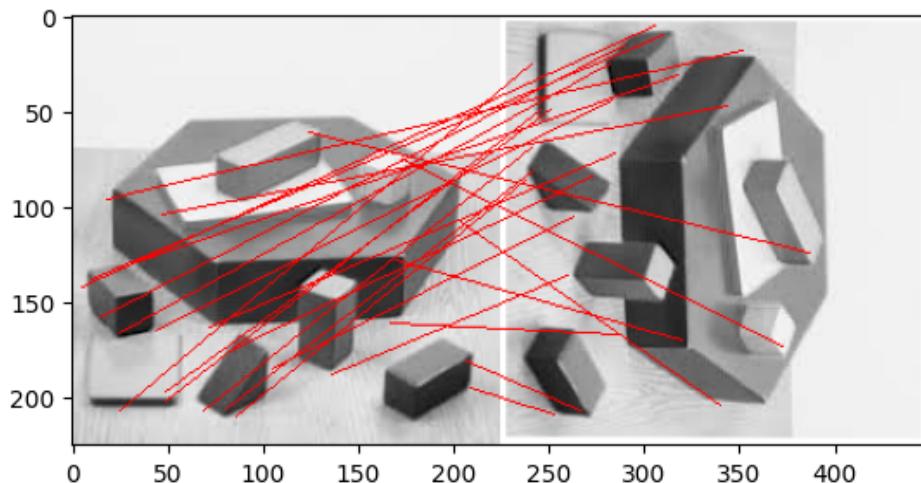


Figure 4.5: Resultados esperados de la correspondencia de características

Una vez comprendido el funcionamiento de un detector de puntos de interés y descriptores se anima al alumno a emplear el método propio de la librería OpenCV como se proporciona en el cuaderno.

### Preguntas

#### Pregunta C.1: Correspondencia de imágenes propias y evaluación

Con los métodos desarrollados para la extracción de características, haga uso del *pipeline* que ha desarrollado y del método propio de OpenCV con sus propias imágenes. Adjunte los resultados en ambos casos y compárelos extrayendo sus propias conclusiones.

## C.2: Bolsa de palabras

En este apartado se va a generar una bolsa de palabras a partir de un dataset de entrenamiento y se evaluará con un dataset de evaluación. Para generar la bolsa de palabras, se seguirán los pasos descritos en las sesiones de teoría:

1. Extracción de descriptores de puntos de interés de las imágenes del dataset.
2. Construcción de diccionario de palabras visuales (K-Means).
3. Para cada imagen del dataset, generar histograma de ocurrencias de palabras del diccionario.
4. Obtener un clasificador utilizando los histogramas de las imágenes del dataset (SVM).

De esta forma, se comenzará con la carga de las imágenes que compondrán los distintos *datasets* para ser procesadas posteriormente.

### Tarea C.2.1: Carga de datasets de entrenamiento y validación

Como ayuda, se le proporciona una clase de Python llamada `Dataset`. Empleando el método `load()` de esta clase, cargue los *datasets* de entrenamiento y validación.

### Tarea C.2.2: Extracción de los descriptores

Una vez cargados los *datasets* se puede proceder a extraer los descriptores de las imágenes que poblarán nuestra bolsa de palabras. Estos descriptores conformarán las ocurrencias de cada característica visual en la bolsa de palabras. Para ello, primero deberá cargar las imágenes del *dataset* de entrenamiento con el path de cada una de ellas y hacerlo en escala de grises. Despues, empleando los métodos necesarios de OpenCV que ha visto durante la sesión, genere los descriptores y guárdelos en una lista.

### Tarea C.2.3: Creación del vocabulario

Con la bolsa de palabras llena con los descriptores, se puede proceder a agrupar los mismos en las distintas palabras del vocabulario que se va a generar. Para ello, se va a ejecutar el algoritmo de clusterización de K-Means que agrupará estos descriptores. En esta tarea debe añadir los descriptores de la lista que obtuvo como resultado en la tarea anterior a la bolsa de palabras `words` que se inicializa en el código y sobre la que ejecutaremos el algoritmo. Fíjese en que se determina el número de palabras deseadas en el vocabulario final con la variable `vocabulary_size` y que esto afectará al desempeño del clasificador más adelante.

### Tarea C.2.4: Entrenamiento del clasificador

Una vez tenemos las características de las imágenes agrupadas en las palabras, se va a entrenar un clasificador que sea capaz de diferenciar entre estos grupos. Este será un modelo basado en una *Support Vector Machine*(SVM). Con este clasificador podremos recibir imágenes externas al *dataset* de entrenamiento y clasificarlas en nuestro grupo de palabras.

Para entrenar el clasificador se aporta la clase `BoW` como ayuda. Añada los argumentos necesarios a los métodos especificados en la celda de código. Compruebe qué argumentos tiene que añadir accediendo al código de la clase proporcionada y analizando los métodos declarados. El modelo final se guardará en el ordenador para su uso más adelante.

### Tarea C.2.5: Inferencia en el dataset de entrenamiento

Con el modelo entrenado, es hora de ver cómo trabaja y comprobar su desempeño. En este caso la métrica empleada será la precisión y podrá observar la matriz de confusión generada para las predicciones.

Al igual que en la tarea anterior, añada los argumentos necesarios a los métodos especificados en la celda de código.

### Tarea C.2.6: Inferencia en el dataset de validación

Después de extraer los resultados con el dataset de entrenamiento, se tienen que comparar los mismos con los del dataset de validación donde se debería obtener un desempeño inferior.

Al igual que en la tarea anterior, añada los argumentos necesarios.

## Preguntas

### Pregunta C.2.A: Cambio de SIFT por KAZE

Ejecute el proceso completo de nuevo cambiando el extractor de características SIFT por KAZE incluido también en la librería de OpenCV y en las clases de ayuda proporcionadas. Incluya el código y los resultados de inferencia en entrenamiento y validación.

### Pregunta C.2.B: ¿Cuantas palabras uso?

Como ha podido comprobar, a la hora de agrupar los descriptores y generar el vocabulario, se indica el tamaño del vocabulario y número de palabras que se desea. Itere sobre este valor y fíjese en los efectos de este sobre el desempeño final con el dataset de entrenamiento y validación. Incluya como resultados los valores empleados, la precisión obtenida y las conclusiones sobre el número adecuado de palabras a emplear.

### **EXTRA** - Pregunta C.2.C: En busca de los mejores parámetros

Al igual que ha iterado en la pregunta anterior con el número de palabras, itere con el resto de parámetros como el número de iteraciones para el K-Means, el número de iteraciones para el SVM, el extractor de características... Proporcione el código final con los parámetros de entrenamiento y los resultados obtenidos. El grupo que obtenga mayor precisión en validación recibirá un punto extra en la práctica.