Tarea 5: Detección de objetos usando SIFT

Isaac Salas Carmona *(Universidad de Guanajuato, División de ingenierías campus Irapuato – Salamanca, i.salascarmona@ugto.mx, Visión por Computadora)*

I. INTRODUCCIÓN

El algoritmo **Scale-Invariant Feature Transform (SIFT)** es bastante conocido en el campo de la Visión por Computadora, dado que es una herramienta poderosa con su capacidad para detectar y describir puntos clave invariantes a cambios de escala, rotación, e iluminación. Este algoritmo está actualmente patentado y su uso ya no es libre, pero, existe una implementación de código abierto en OpenCV llamada **ORB** que utilizaremos para llevar a cabo nuestra tarea**:**

*"Realizar un programa usando openCV y Python que sea capaz de detectar 10 diferentes objetos de la base de datos COIL-100. Utiliza el algoritmo **SIFT**"*

II. DESARROLLO

Para lograr resolver este problema, utilice la técnica de **Brute-Force Matching with ORB Descriptors** que se encuentra descrita en la documentación de OpenCV y en el archivo README de mi repositorio.
Esta técnica, por lo que estuve investigando se utiliza en su mayoría para comparar 1 sola imagen de referencia contra 1 sola imagen de comparación para obtener las coincidencias entre ambas. Es la más sencilla de utilizar y la que encontré primero durante mi investigación así que decidí adaptarla para que utilizara **10 imágenes de referencia** y las compare con la imagen de comparación, a fin de detectar el objeto deseado.

Mi algoritmo sigue los siguientes pasos:

- **Entrenar al programa para que detecte objetos:**

    1. Definimos un rango de imágenes (en nuestro caso **10 imágenes**) de vistas del objeto que queremos poder reconocer.
    2. Guardamos esas 10 imágenes, junto a sus **keypoints (puntos clave)** y **descriptors (descriptores)** calculados con el detector **ORB**. Esto pasa dentro de la función "Training" y se repite las veces que sean necesarias para reconocer el número de objetos deseados, **10 en nuestro caso**.
    3. Definimos una lista con keypoints, descriptores, imágenes y **etiquetas (el nombre del objeto)** de los 10 objetos que tenemos en nuestra base de datos.
    4. Con todos estos datos podemos iniciar la comparación

- **Comparar los datos de los objetos detectables contra el objeto de la imagen a comparar:**

    1. Definimos la imagen que queremos utilizar (su número de objeto y su vista).
    2. Guardamos esa imagen junto a sus keypoints y descriptores calculados con el detector **ORB**.
       Esto ocurre dentro de la función "Automate", la cual también llama a la función de "Compare" y "DrawMatches".
    3. Comparamos cada descriptor de la lista de objetos con el descriptor del objeto que queremos reconocer utilizando el **Brute Force Matcher ("BFMatcher")** de OpenCv y guardamos la lista de **matches (coincidencias)** TOTALES que resultan de este cálculo.
       Esto ocurre dentro de la función "Compare".
    4. Dibujamos todas las coincidencias en memoria con la función **"drawMatches"** de OpenCV y hacemos un cálculo que promedia el número de coincidencias totales resultantes sobre el número de keypoints totales de la base de datos, el resultado va a ser el **porcentaje de seguridad o de coincidencia** de la imagen de comparación con las imágenes de entrenamiento, el porcentaje más alto va a decidir que objeto de la base de datos es el mostrado dentro de la imagen de comparación.
       Esto ocurre dentro de la función "DrawMatches".

- **Dar un resultado de la comparación:**

  Aquí tenemos 2 opciones:

    1. Mostramos en la consola de salida todos los porcentajes de coincidencia, el resultado de la comparación **(nombre del objeto detectado)** y las imágenes que se estuvieron comparando junto a sus coincidencias.

       o

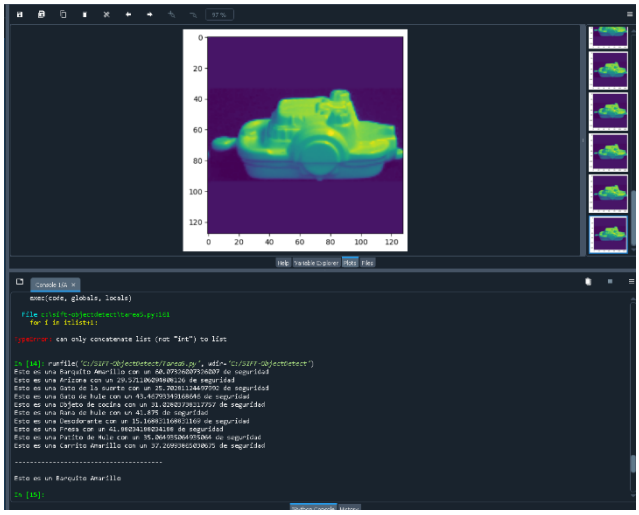
    2. Mostramos en la consola de salida todos los porcentajes de coincidencia, el resultado de la comparación **(nombre del objeto detectado)** y la imagen original de comparación.

III. PRUEBAS Y RESULTADOS

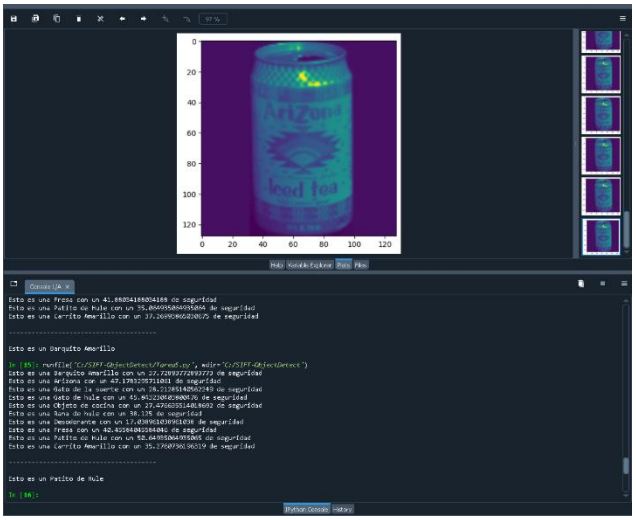
Para las pruebas aquí documentadas usaremos la opción **2** de desplegar los resultados y vamos a utilizar **10 imágenes** para entrenar cada conjunto de datos para cada objeto como había mencionado antes.
La opción de desplegar los resultados así es por que de esta manera tarda menos tiempo en calcular y desplegar los resultados, y la decisión de utilizar 10 imágenes para cada objeto fue porque por prueba y error encontré que esa me daba mayor precisión al momento de hacer la comparación.

- Hice entonces 10 pruebas, cada una con 1 imagen que corresponde a cada objeto que el programa esta entrenado para reconocer:

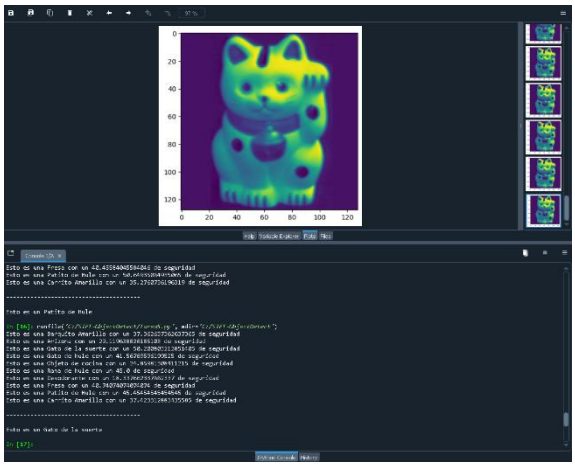
1. Barquito amarillo



2. Arizona

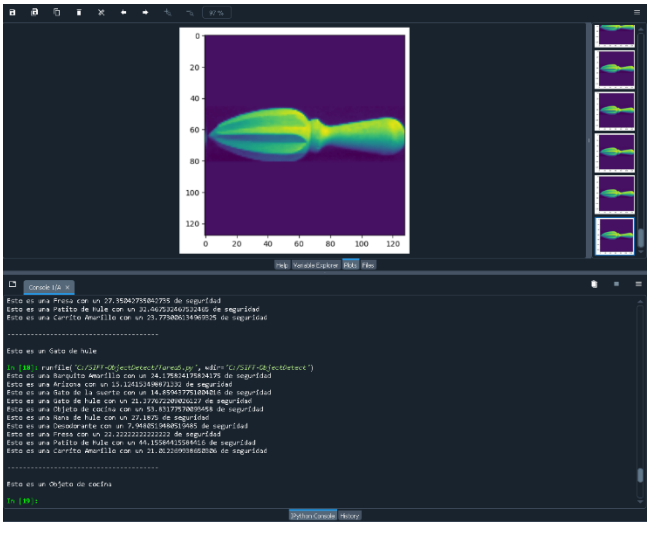


3. Gatito de la suerte

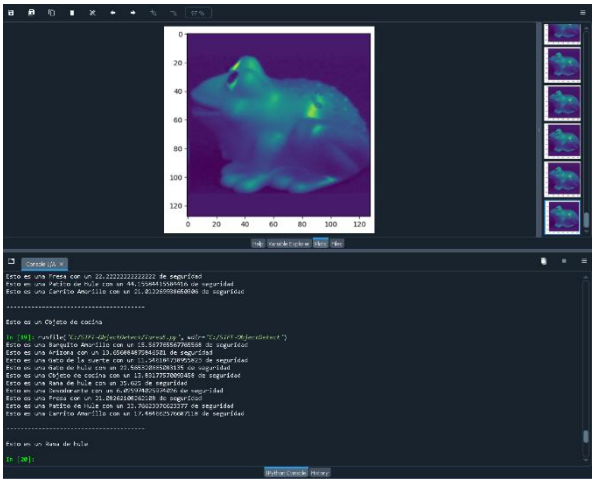


4. Gato de hule

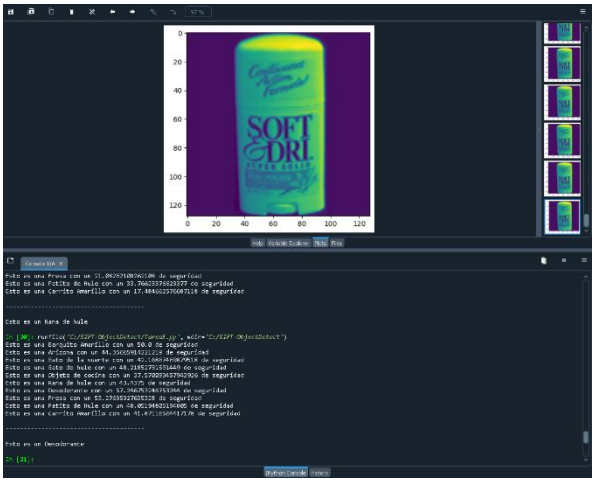
5. Objeto de cocina



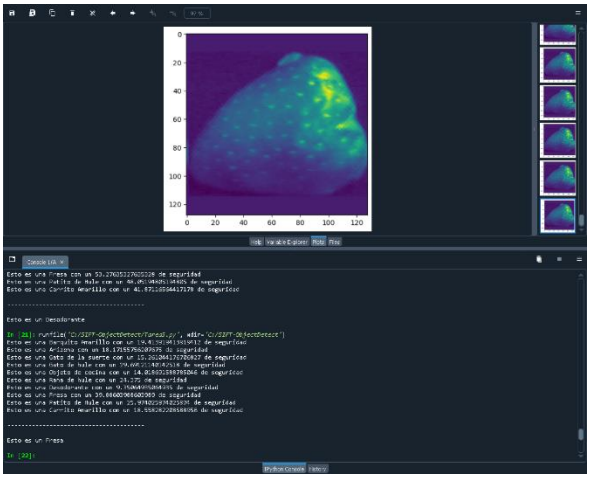
6. Rana de hule



7. Desodorante



8. Fresa



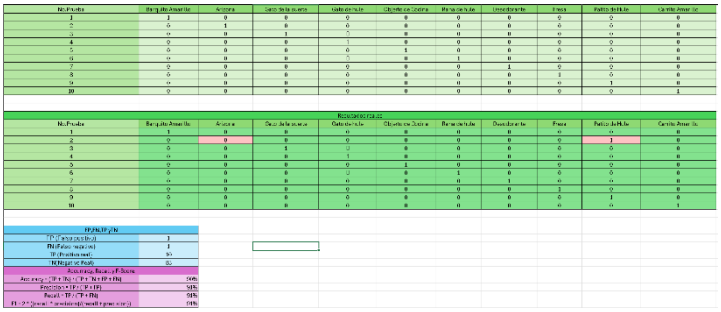
9. Patito de hule

10. Carro amarillo

- Y como podemos ver, se detectan 9 de 10 objetos de manera satisfactoria.

IV. CONCLUSIONES

Vamos a ver los valores que debíamos de calcular de acorde a la asignación:



Anexo el archivo de Excel junto a todo el demás material si se ocupan mas detalles, pero, en breve tenemos:

- Accuracy de 90%
- Precision de 91%
- Recall de 91%
- Y un F1 Score de 91%

Con estos resultados creo yo que podemos decir que la actividad se completo de manera exitosa y que, aunque aún podríamos intentar jugar más con los valores de cuantas imágenes se usan en el entrenamiento, intentar llegar a valores más altos de precisión o exactitud, o cambiar la técnica utilizada para procesar las imágenes, de cualquier manera, me siento muy satisfecho de todo lo que logré y aprendí durante la elaboración de este programa.

Para ver a más detalle todo, la versión mas reciente de mi código se encuentra en: https://github.com/Isaac-Salas/SIFT-ObjectDetect

V. REFERENCIAS

- OpenCV: Feature matching. (n.d.). https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html
- ORB (Oriented FAST and Rotated BRIEF) — OpenCV 3.0.0-dev documentation. (n.d.). https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html
- GeeksforGeeks. (2023, September 21). SIFT Interest Point Detector using Python OpenCV. GeeksforGeeks. https://www.geeksforgeeks.org/sift-interest-point-detector-using-python-opencv/
- Wikipedia contributors. (2024, July 18). Oriented FAST and rotated BRIEF. Wikipedia. https://en.wikipedia.org/wiki/Oriented_FAST_and_rotated_BRIEF
- Wikipedia contributors. (2024b, September 21). Scale-invariant feature transform. Wikipedia. https://en.wikipedia.org/wiki/Scale-invariant_feature_transform