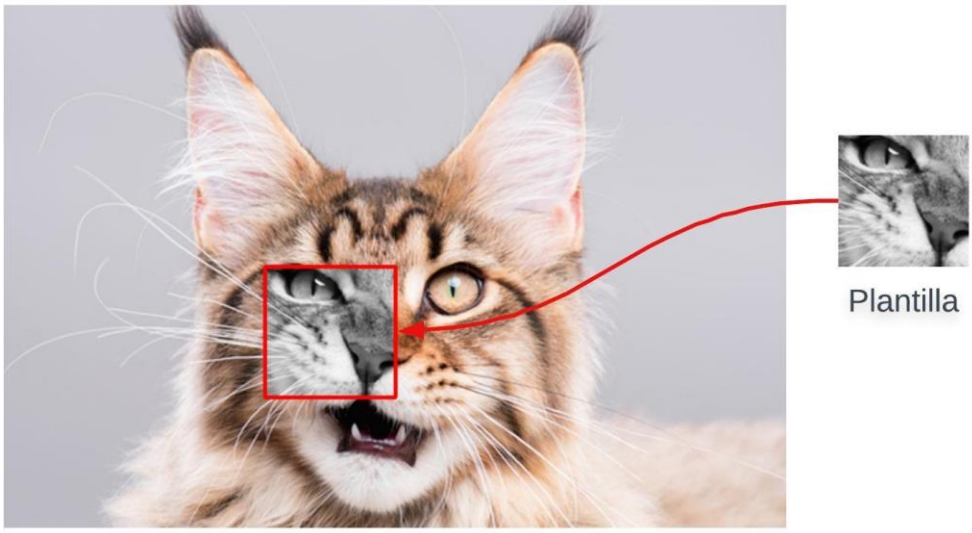
ACTIVIDAD_1 Template Matching

DESCRIPCIÓN: La **Template matching** (comparación de plantillas) es una técnica para encontrar áreas de una imagen que son similares a un patrón (plantilla). La suma de las diferencias absolutas proporciona una forma simple de automatizar la búsqueda de objetos dentro de una imagen, pero puede no ser confiable debido a los efectos de factores contextuales como cambios en la iluminación, el color, la dirección de visualización, el tamaño o la forma.

OBJETIVO: Implementar la búsqueda de patrones a través de la suma de diferencias absolutas comparando las intensidades de los píxeles de la plantilla y regiones de la imagen original.



Funciones a utilizar:

```
cv2.imread() -  cv2.merge() - cv2.rectangle() - cv2.absdiff()
cv2.imshow() -  cv2.waitKey() - cv2.destroyAllWindows() - np.sum()
```

Un patrón es una imagen pequeña con ciertas características. El objetivo de la comparación de plantillas (Template Matching) es encontrar el patrón/plantilla en una imagen.

Figura 1. Imagen plantilla (T).

Para encontrarlo, el usuario tiene que dar dos imágenes de entrada: **Imagen de origen (S)** – La imagen para encontrar la plantilla y la **Imagen de plantilla (T)** – La imagen con cierto atributo que se encuentra en la imagen de origen.

Figura 2. Imagen de origen (S).

Las ocurrencias de patrones tienen que conservar la orientación de la imagen de patrón de referencia (plantilla). Como resultado, no funciona para versiones rotadas o escaladas de la plantilla como un cambio en la forma/tamaño.

PROCEDIMENTO

Para Implementar la búsqueda de patrones a través de la suma de diferencias absolutas ejecutaremos los siguientes pasos:

- Realizar lectura de imágenes original (`original.jpg`) y plantilla (`1.jpg`) en escala de grises.

- Definir las dimensiones de la ventana de búsqueda (**ROI**) considerando las características de la imagen plantilla o patrón.

- Desplazar la ventana de búsqueda (recuadro color verde) sobre la imagen original a escala de grises a través de bucles **`for`** anidados:

```
for x in range(0,a):
        for y in range(0,b):
```

Los valores de `a` y `b` corresponden al número de desplazamientos de la ventana de búsqueda en la dirección de **x** y **y** respectivamente. Cada nueva posición de la ventana de búsqueda será comparada con la imagen plantilla. El recorrido puede realizarse de arriba hacia abajo como lo muestra la figura o de derecha a izquierda.

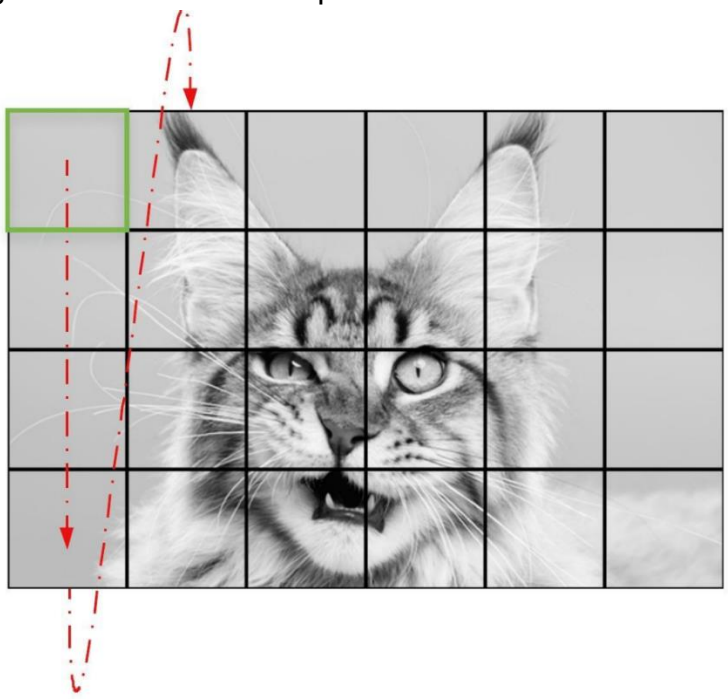


Figura 3. Desplazamiento de ventana de búsqueda.

- Definir la región de interés **ROI**. Recuerde que la región de interés debe tener las mismas dimensiones de la imagen plantilla para posterior operación aritmética.

- Realizar la resta absoluta entre imagen plantilla y ROI.

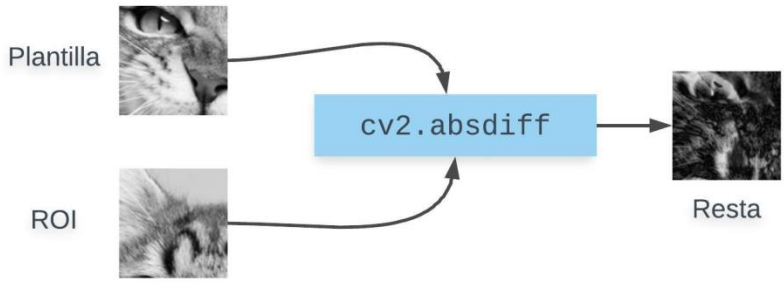


Figura 4. Resta absoluta de *Template* y región de interés ROI.

- Determinar la suma total de los pixeles de la imagen resultado de la operación de resta. Para esto haremos uso de la operación **`np.sum(imagen)`** del módulo *Numpy* que calcula la suma de los elementos de la matriz asociada a la resta.

- Imprime este resultado para establecer un umbral. Si el resultado es menor que el umbral, la región de interés será marcada como patrón detectado.

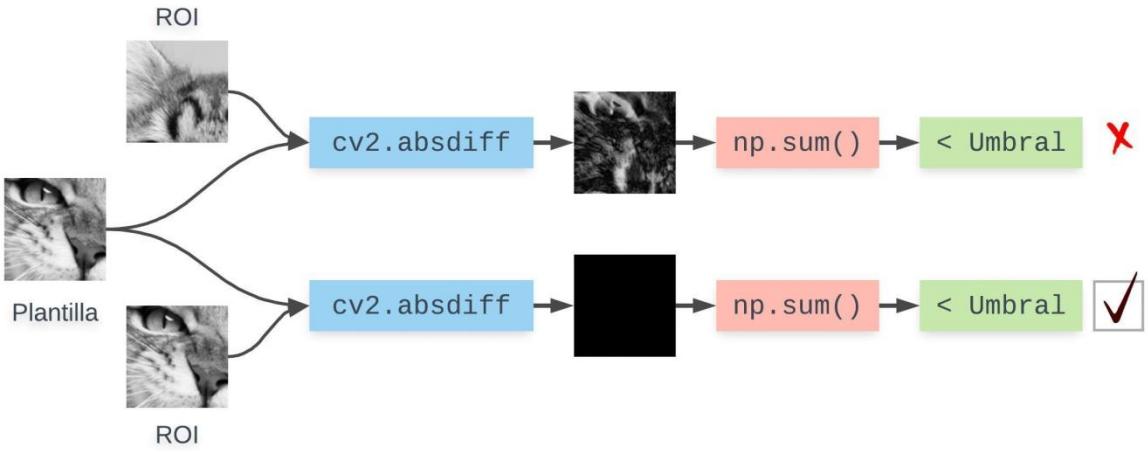
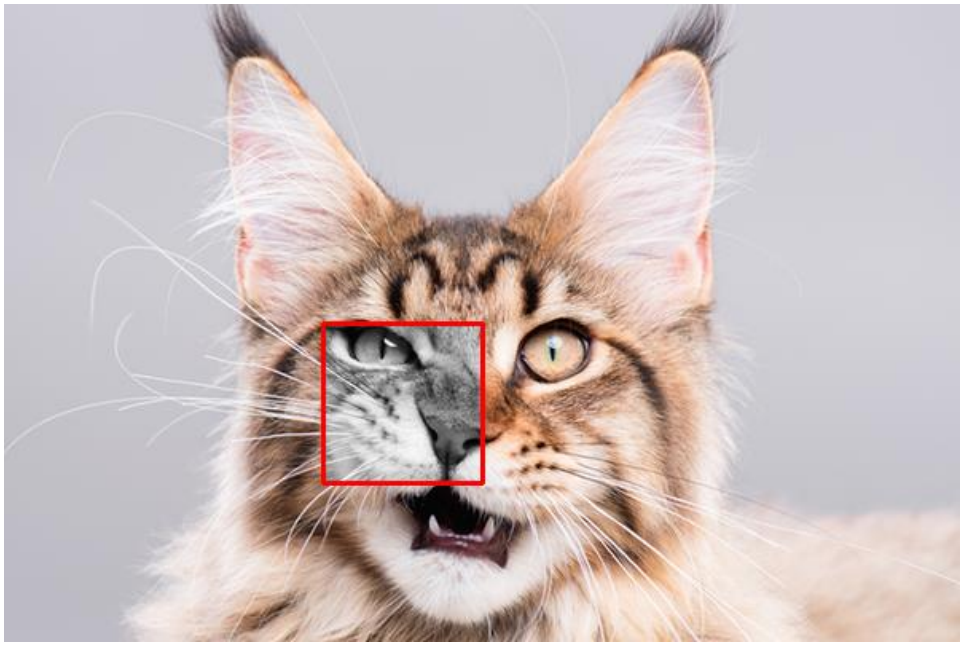


Figura 5. Localización de plantilla a través de suma de diferencias absolutas.

- Una vez localizado el patrón o plantilla se debe dibujar un rectángulo **`cv2.rectangle`** color rojo `(0,0,255)` y asignar a esta región de la imagen original a color la plantilla a escala de grises, para lograr el siguiente resultado:



Para hacer posible esta asignación, la imagen plantilla a escala de grises debe ser de 3 canales. Esto lo realizamos a través de la función **`cv2.merge`**.

- Finalmente ejecuta el algoritmo modificando la plantilla por cada una de las imágenes patrón: