UNIVERSIDAD DEL BÍO-BÍO

25 Template Matching

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Introducción al template matching:

El template matching, o coincidencia de plantillas, es una técnica fundamental en el campo de la visión artificial que se utiliza para encontrar la ubicación de una plantilla o patrón específico dentro de una imagen más grande. La idea central es comparar una plantilla, que es una pequeña imagen o patrón, con regiones superpuestas de una imagen de entrada y encontrar la posición donde la plantilla coincide mejor.

A continuación, se proporciona una introducción a algunos aspectos clave del template matching y sus utilidades en visión artificial:

Proceso de Template Matching:

Selección de la Plantilla:

- Se elige una pequeña imagen llamada plantilla, que representa el patrón que se busca en la imagen de entrada.

Comparación de la Plantilla:

- La plantilla se desliza o superpone sobre la imagen de entrada en diferentes ubicaciones y se compara la similitud entre la plantilla y la región correspondiente de la imagen.

Función de Similitud:

- Se utiliza una función de similitud para medir qué tan bien coincide la plantilla con la región actual. Una métrica común es la correlación cruzada o la suma de los cuadrados de las diferencias (SSD).

Ubicación de la Coincidencia:

- La posición donde la similitud es máxima indica la ubicación donde se encuentra el patrón en la imagen.

Utilidades en Visión Artificial:

Detección de Objetos:

- El template matching se utiliza para detectar objetos específicos dentro de una imagen, ya que puede encontrar regiones donde un objeto coincide con una plantilla predefinida.

Seguimiento de Objetos en Video:

- Es útil para realizar seguimiento de objetos a lo largo del tiempo en secuencias de video al buscar la plantilla en cada fotograma.

Reconocimiento de Patrones:

- Se utiliza en reconocimiento de patrones para identificar regiones específicas con características particulares en una imagen.

Calibración de Cámaras:

- Puede aplicarse en la calibración de cámaras para encontrar patrones conocidos en las imágenes capturadas y calcular parámetros de la cámara.

Automatización Industrial:

- En entornos industriales, el template matching se utiliza para la inspección de calidad al buscar defectos o patrones específicos en productos.

Aunque el template matching es efectivo en ciertos escenarios, también tiene limitaciones, como la sensibilidad a la variabilidad en la apariencia y el tamaño de los objetos. En consecuencia, en aplicaciones más complejas, se pueden utilizar enfoques más avanzados, como el uso de características locales y algoritmos de aprendizaje profundo.

Se mostrará código de Python implementa una interfaz gráfica simple para realizar coincidencia de plantillas en imágenes o secuencias de video provenientes de una cámara. Aquí hay una explicación detallada del código:

Librerías Utilizadas:

```python
import cv2
import os
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
```

- **cv2**: OpenCV, una biblioteca de visión por computadora.
- **os**: Operaciones de sistema operativo, utilizada para manipulación de
- rutas y archivos.
- **tkinter**: Interfaz gráfica de usuario estándar de Python.
- **PIL**: Python Imaging Library, utilizada para manejar imágenes.

Variables Globales:

```python
# Variables globales
cap = None
frame_gris = None
rutas_imagenes = []
imagen_seleccionada = None
puerto_camara = 0
```

- **puerto_camara**: Número del puerto de la cámara (por defecto 0 para la cámara principal).
- cap: Objeto para acceder a la cámara.
- **frame_gris**: Almacena el frame actual convertido a escala de grises.
- **rutas_imagenes**: Lista que contendrá las rutas de las imágenes en una carpeta seleccionada.
- **imagen_seleccionada**: Ruta de la imagen actualmente seleccionada.

Funciones Principales:

1. **coincidencia_de_plantilla**(): Realiza la coincidencia de plantilla entre la imagen seleccionada y el frame actual de la cámara. Utiliza la técnica cv2.matchTemplate y devuelve el resultado y la posición de la coincidencia.

```
def coincidencia_de_plantilla():
    if imagen_seleccionada is not None:
        plantilla = cv2.imread(imagen_seleccionada, cv2.IMREAD_COLOR)
        plantilla_gris = cv2.cvtColor(plantilla, cv2.COLOR_BGR2GRAY)

        # Realizar coincidencia de plantilla en el frame actual
        resultado = cv2.matchTemplate(frame_gris, plantilla_gris, cv2.TM_CCOEFF_NORMED)
        _, _, _, max_loc = cv2.minMaxLoc(resultado)

        return resultado, max_loc
    else:
        return None, None
```

procesar_frame(): Captura y procesa continuamente los frames de la cámara. Convierte cada frame a escala de grises y los muestra en la interfaz gráfica.

```
def procesar_frame():
    ret, frame = cap.read()

    if ret:
        global frame_gris
        frame_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Mostrar el frame en la interfaz gráfica
        mostrar_frame()

        # Llamar a esta función nuevamente después de un breve intervalo
        root.after(10, procesar_frame)
```

mostrar_frame(): Muestra el frame actual en la interfaz gráfica utilizando la librería PIL.

```python
def mostrar_frame():
    if cap.isOpened():
        ret, frame = cap.read()
        if ret:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img = Image.fromarray(frame)
            img = ImageTk.PhotoImage(img)
            panel.img = img
            panel.config(image=img)
    else:
        panel.config(text="Error al abrir la cámara")
```

comparar_imagenes(): Llama a **coincidencia_de_plantilla**() y compara la similitud entre la plantilla y la región actual. Actualiza una etiqueta en la interfaz gráfica indicando si hay o no coincidencia.

```python
def comparar_imagenes():
    # Realizar coincidencia de plantilla solo con la imagen seleccionada
    resultado, max_loc = coincidencia_de_plantilla()

    # Si el valor máximo está por encima del umbral, mostrar el resultado
    if resultado is not None and umbral_var.get() < resultado[max_loc[1], max_loc[0]]:
        texto_resultado.set("SI hay coincidencia")
    else:
        texto_resultado.set("NO hay coincidencia")
```

seleccionar_carpeta(): Abre un cuadro de diálogo para seleccionar una carpeta que contiene imágenes. Habilita varios botones después de seleccionar la carpeta.

```python
def seleccionar_carpeta():
    ruta_carpeta = filedialog.askdirectory()
    if ruta_carpeta:
        global rutas_imagenes
        rutas_imagenes = [os.path.join(ruta_carpeta, archivo) for archivo in os.listdir(ruta_carpeta) if archivo.endswith(('.png', '.jpg', '.jpeg'))]

        # Activar botones después de seleccionar la carpeta
        boton_seleccionar_imagen.config(state=tk.NORMAL)
        boton_siguiente_imagen.config(state=tk.NORMAL)
        boton_iniciar_camara.config(state=tk.NORMAL)
        boton_comparar.config(state=tk.NORMAL)
```

iniciar_camara(): Inicia la cámara en el puerto especificado y comienza el procesamiento continuo de frames.

```python
def iniciar_camara():
    global cap
    cap = cv2.VideoCapture(puerto_camara)
    if not cap.isOpened():
        texto_resultado.set("Error al abrir la cámara")
    else:
        texto_resultado.set("Cámara iniciada")
        procesar_frame()
```

seleccionar_imagen(): Abre un cuadro de diálogo para seleccionar una imagen específica y actualiza la etiqueta correspondiente en la interfaz gráfica.

```python
def seleccionar_imagen():
    global imagen_seleccionada
    imagen_seleccionada = filedialog.askopenfilename(filetypes=[("Imagen", "*.png;*.jpg;*.jpeg")])
    etiqueta_nombre_imagen.config(text="Imagen seleccionada: " + os.path.basename(imagen_seleccionada))
    boton_comparar.config(state=tk.NORMAL)
```

siguiente_imagen(): Permite navegar a la siguiente imagen en la carpeta de imágenes seleccionada.

```python
def siguiente_imagen():
    global imagen_seleccionada
    if rutas_imagenes:
        index_actual = rutas_imagenes.index(imagen_seleccionada) if imagen_seleccionada in rutas_imagenes else -1
        siguiente_index = (index_actual + 1) % len(rutas_imagenes)
        imagen_seleccionada = rutas_imagenes[siguiente_index]
        etiqueta_nombre_imagen.config(text=os.path.basename(imagen_seleccionada))
```

Interfaz Gráfica (tkinter):

```python
# Crear la interfaz gráfica
root = tk.Tk()
root.title("Coincidencia de Plantillas")
root.geometry("600x800")
```

- Se crean botones para seleccionar carpeta, seleccionar imagen, avanzar a la siguiente imagen, iniciar la cámara y comparar imágenes.

```
# Botones y etiquetas
boton_seleccionar_carpeta = tk.Button(root, text="Seleccionar Carpeta", command=seleccionar_carpeta)
boton_seleccionar_carpeta.pack(pady=10)

boton_seleccionar_imagen = tk.Button(root, text="Seleccionar Imagen", command=seleccionar_imagen, state=tk.DISABLED)
boton_seleccionar_imagen.pack(pady=10)

boton_siguiente_imagen = tk.Button(root, text="Siguiente Imagen", command=siguiente_imagen, state=tk.DISABLED)
boton_siguiente_imagen.pack(pady=10)

boton_iniciar_camara = tk.Button(root, text="Iniciar Cámara", command=iniciar_camara, state=tk.DISABLED)
boton_iniciar_camara.pack(pady=10)
```

- Se incluyen etiquetas para mostrar el nombre de la imagen seleccionada y el resultado de la comparación.

```
# Etiqueta de nombre de imagen
etiqueta_nombre_imagen = tk.Label(root, text="Aqui se mostrara la imagen seleccionada")
etiqueta_nombre_imagen.pack(pady=10)
```

- Se utiliza un slider para ajustar el umbral de coincidencia.

```
slider_umbral = tk.Scale(root, from_=0, to=1, resolution=0.01, orient=tk.HORIZONTAL, variable=umbral_var)
slider_umbral.pack(pady=5)
```

Se sugiere usar 7 o 8 de umbral para un buen resultado.

Bucle Principal:

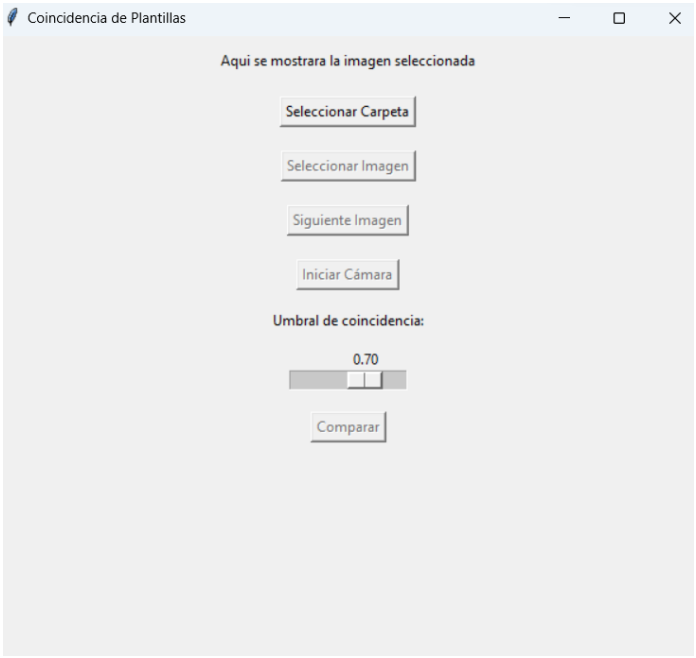
```
# Iniciar el bucle de la interfaz gráfica
root.mainloop()
```

- **root.mainloop()**: Inicia el bucle principal de la interfaz gráfica.

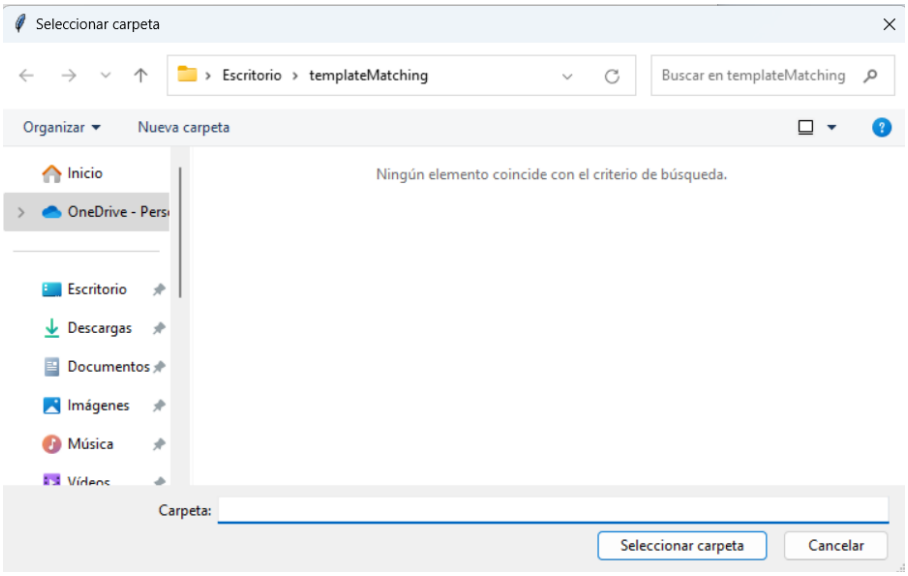
Este código proporciona una interfaz sencilla para explorar y comparar imágenes utilizando la técnica de coincidencia de plantillas en tiempo real desde una cámara o imágenes almacenadas en una carpeta.

Tutorial de uso

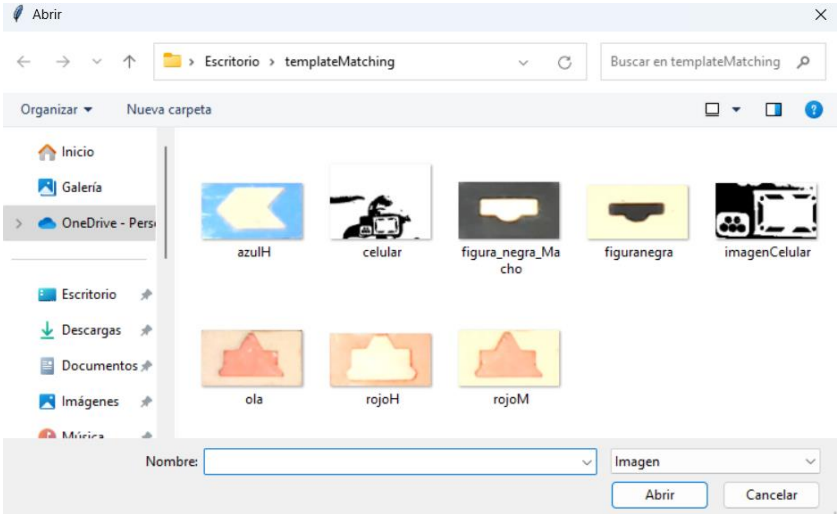
El resultado de la interfaz gráfica es el siguiente:



Primero debemos seleccionar la carpeta donde se encuentran las imágenes que guardamos desde el generador de templates.



Una vez seleccionada la carpeta procedemos a cargar las imágenes que están dentro de la carpeta.



Luego podemos iniciar la cámara para comparar con el template que se esté mostrando en tiempo real, para realizar la comparacion tendremos que presionar el boton Comparar y tambien podemos seguir avanzando con la siguiente imagen para ver si sigue un patron

similar, recuerda ajustar el umbral, como observacion funciona mejor para figuras en blanco y negro.