

**26 Tutorial Inspección Visual Automatizada: Coincidencia de Plantillas y Análisis de Manchas en Tiempo Real**

**Estudiantes Practicantes:**

Luis Pereira

**Profesor:**

Luis Vera

**Laboratorio CIMUBB**

2023-2

### **Introducción:**

El programa que se presenta a continuación se ha diseñado para desempeñar un papel esencial en un proceso de inspección visual automatizado, emulando de alguna manera el funcionamiento de una máquina de clasificación como las utilizadas en entornos industriales, como los supermercados.

#### **Reconocimiento de Plantillas:**

El programa utiliza el poderoso motor de coincidencia de plantillas de OpenCV para identificar y reconocer figuras específicas en las imágenes capturadas en tiempo real por una cámara. Estas plantillas actúan como referencias visuales, y cuando se encuentran coincidencias, se activan acciones correspondientes.

#### **Análisis de Manchas:**

Simultáneamente, el programa realiza un análisis detallado de las manchas presentes en las áreas de interés de las imágenes. Esta funcionalidad es crucial para evaluar la calidad y la integridad de las figuras identificadas. Se cuentan y cuantifican tanto las manchas blancas como las negras, proporcionando datos precisos sobre la limpieza y el estado general de cada figura.

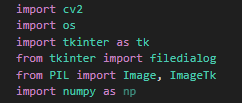
#### **Aplicación en un Proceso de Inspección:**

Imaginemos un escenario donde las imágenes capturadas representan productos o componentes que pasan por una línea de producción. El programa, actuando como una "máquina de inspección visual", verifica la presencia y la calidad de las figuras deseadas, al mismo tiempo que evalúa la presencia de manchas que podrían indicar defectos o condiciones no deseadas.

Este enfoque integrado no solo agiliza el proceso de inspección, sino que también mejora la confiabilidad al combinar la identificación visual con métricas cuantitativas sobre la limpieza de las figuras. El resultado es un sistema robusto que contribuye a la calidad y la eficiencia en la producción.

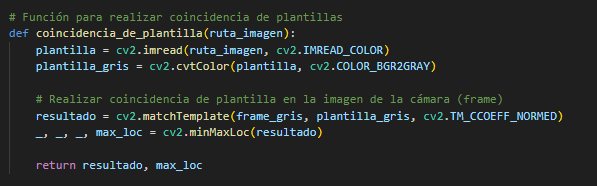
**Explicacion del codigo**

### **1. Importar las bibliotecas necesarias:**

****

### **2. Funciones utilizadas:**

* **coincidencia\_de\_plantilla(ruta\_imagen)**: Función para realizar coincidencia de plantillas en una imagen.



La función **coincidencia\_de\_plantillas(ruta\_imagen)** tiene como objetivo buscar coincidencias de una plantilla específica en el fotograma actual capturado por la cámara. Aquí hay una explicación paso a paso de lo que hace:

1. **Cargar la Plantilla**: Lee la imagen de la plantilla desde la ruta especificada (**ruta\_imagen**) utilizando OpenCV y la almacena en la variable plantilla.
2. Convertir a Escala de Grises: Convierte la plantilla a escala de grises utilizando **cv2.cvtColor()**, guardando el resultado en la variable plantilla\_gris.
3. **Realizar Coincidencia de Plantilla:** Utiliza la función **cv2.matchTemplate()** para realizar una comparación entre la plantilla y la imagen actual del fotograma (almacenada en **frame\_gris**). Esto genera un mapa de coincidencia, guardado en la variable resultado.
4. **Encontrar la Mejor Coincidencia:** Utiliza **cv2.minMaxLoc()** para encontrar las coordenadas del valor máximo en el mapa de coincidencia (**max\_loc**). Este valor máximo indica la región donde la plantilla coincide mejor con la imagen.
5. **Devolver Resultados**: La función devuelve el mapa de coincidencia (**resultado**) y las coordenadas de la mejor coincidencia (**max\_loc**), que se pueden utilizar para identificar la ubicación de la plantilla en la imagen.

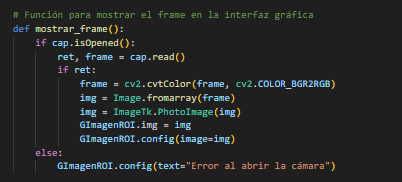
* **procesar\_frame()**: Función para procesar cada fotograma del flujo de la cámara.



La función **procesar\_frame()** realiza el procesamiento principal de cada fotograma capturado por la cámara. Aquí está una explicación paso a paso de lo que hace:

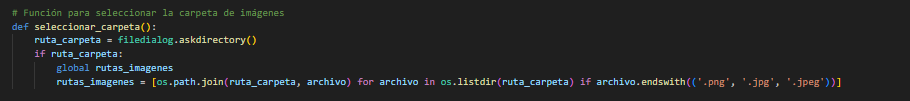
1. **Capturar Fotograma**: Utiliza **cap.read()** para capturar un fotograma de la cámara, y almacena el resultado en frame.
2. **Convertir a Escala de Grises**: Convierte el fotograma (frame) a escala de grises utilizando **cv2.cvtColor()**, y guarda el resultado en **frame\_gris**.
3. **Binarizar la Imagen**: Utiliza **cv2.threshold()** para binarizar la imagen en blanco y negro (**th**). Este paso facilita la detección de contornos y objetos en la imagen.
4. **Encontrar Contornos**: Utiliza **cv2.findContours()** para identificar los contornos en la imagen binarizada. Los contornos se almacenan en la variable contornos.
5. **Contar Objetos:**
   * Itera sobre los contornos encontrados.
   * Filtra los contornos basándose en el área.
   * Aproxima cada contorno a un polígono.
   * Si el polígono tiene 4 vértices, lo considera un rectángulo.
   * Dibuja un contorno alrededor del objeto y lo recorta.
   * Muestra la imagen recortada en un segundo label (**LImagen**).
   * Llama a la función **contar\_manchas()** si hay una imagen binarizada disponible.
6. **Coincidencia de Plantillas:**
   * Itera sobre las rutas de las imágenes de plantillas.
   * Llama a la función **coincidencia\_de\_plantilla()** para cada plantilla y verifica si hay coincidencias por encima de un umbral.
   * Muestra el resultado en la interfaz gráfica.
7. **Mostrar el Frame en la Interfaz Gráfica:** Utiliza **mostrar\_frame()** para mostrar el fotograma actualizado en la interfaz gráfica.
8. **Llamada Recursiva**: Llama a la función nuevamente después de un breve intervalo **(ventana.after(10, procesar\_frame))**, asegurando un procesamiento continuo y en tiempo real.

* **mostrar\_frame()**: Función para mostrar el fotograma actual en la interfaz gráfica.



La función **mostrar\_frame()** es esencial para visualizar el fotograma actual en la interfaz gráfica, permitiendo que la aplicación muestre el flujo en tiempo real proveniente de la cámara.

* **seleccionar\_carpeta():** Función para seleccionar una carpeta de imágenes.



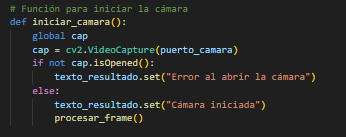
La función **seleccionar\_carpeta()** facilita al usuario la selección de una carpeta que contiene imágenes para ser utilizadas en el proceso de coincidencia de plantillas. Las rutas de estas imágenes se almacenan para su posterior procesamiento.

* **contar\_manchas()**: Función para contar manchas blancas y negras en una imagen.



La función **contar\_manchas()** realiza un análisis cuantitativo de las manchas en la imagen binarizada recortada y presenta los resultados en la interfaz gráfica. Esto proporciona información sobre la presencia y la extensión de manchas blancas y negras en la región de interés de la imagen.

* **iniciar\_camara():** Función para iniciar la cámara y procesar los fotogramas.



### **3. Crear la interfaz gráfica con Tkinter:**

* **Botones de Acción:**
  + **Seleccionar Carpeta:** Permite al usuario elegir una carpeta que contiene imágenes de plantillas para el proceso de coincidencia.
  + **Iniciar Cámara**: Inicia el flujo de video desde la cámara en tiempo real para la inspección visual continua.
* **Etiqueta de Resultado:**
  + Muestra mensajes informativos sobre el estado del programa, como si la cámara se ha iniciado correctamente o si hay algún error.
* **Área de Visualización de Imágenes:**
  + Dos cuadros de imágenes (**GImagenROI** y **LImagen**) que presentan el flujo de video en tiempo real desde la cámara y la imagen binarizada recortada respectivamente. Estas áreas ofrecen una visualización en tiempo real de los objetos detectados y las manchas analizadas.
* **Cuadros de Texto:**
  + Dos cuadros de texto (**CajaTextoBlanco** y **CajaTextoNegro**) que proporcionan información detallada sobre la cantidad de manchas blancas y negras detectadas, así como el porcentaje de área ocupada por estas manchas.
* **Controles de Umbral:**
  + Dos controles deslizantes (**slider\_umbral** y **slider\_umbral2**) para ajustar dinámicamente los umbrales utilizados en el proceso de coincidencia de plantillas y en la binarización de la imagen recortada.
* **Información Adicional:**
  + Etiquetas informativas (**etiqueta\_manchas\_blancas** y **etiqueta\_manchas\_negras**) que describen la información presentada en los
  + cuadros de texto.

La interfaz gráfica se ha diseñado para proporcionar una experiencia fácil de usar y comprender, permitiendo al usuario controlar y monitorear la inspección visual automatizada en tiempo real. La combinación de elementos visuales y datos cuantitativos mejora la eficacia del proceso de análisis, facilitando la identificación y evaluación de objetos específicos y la presencia de manchas.

****

**4. Resultado final**

****

Para garantizar un rendimiento óptimo del programa, es esencial seguir ciertos pasos clave. Aquí hay una guía que destaca las consideraciones importantes:

* **Selección de Carpeta con Plantillas:**
  + Antes de iniciar la inspección visual, asegúrate de seleccionar una carpeta que contenga las plantillas de manera clara y visible. Esto proporcionará al programa las referencias necesarias para realizar la coincidencia de patrones.
* **Uso del Programa Generador de Plantillas:**
  + Se recomienda utilizar el programa generador de plantillas adjunto. Este programa facilita el recorte de las imágenes de plantillas, lo que contribuye a una coincidencia más precisa y eficiente durante el proceso de inspección.
* **Ajuste del Umbral de Coincidencia:**
  + Selecciona cuidadosamente el umbral de coincidencia utilizando el control deslizante correspondiente. Ajustar este umbral de manera adecuada permitirá una identificación más precisa de las plantillas en el flujo de video de la cámara.
* **Umbral de la Imagen Recortada:**
  + El control deslizante destinado al umbral de la imagen recortada es fundamental para un análisis de manchas eficiente. Ajusta este umbral según sea necesario para obtener resultados precisos en la detección y cuantificación de manchas.

Estas consideraciones ayudarán a optimizar el rendimiento del programa y garantizarán una inspección visual eficaz. Recuerda que la selección adecuada de carpetas y umbrales es crucial para obtener resultados precisos y significativos durante la ejecución del programa.