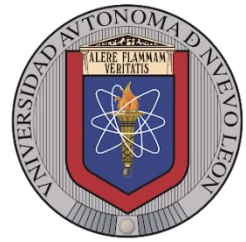


UANL

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

FIME

Nombre: Roberto Erick Aguilar Morales

Matricula: 1871004

Carrera: Ingeniero en Tecnologías del Software

9.– Polígonos y Agujeros

Materia: VISION COMPUTACIONAL LABORATORIO

Docente: RAYMUNDO SAID ZAMORA PEQUEÑO

Hora: N1-N2 **Días: Miércoles**

Fecha: 24/11/24

Objetivo

Detectar y analizar características geométricas en una imagen, identificando esquinas, agrupándolas en polígonos y localizando posibles agujeros dentro de estos, para visualizar y exportar los resultados en formatos gráficos y numéricos.

Marco teórico

Procesamiento de Imágenes

El procesamiento de imágenes es una disciplina que se ocupa de la manipulación y análisis de imágenes digitales a través de computadoras. Se emplea en diversas áreas, desde la medicina y la astronomía, hasta la inteligencia artificial y el control de calidad industrial. Dentro de esta área, el propósito principal es extraer información útil de las imágenes o transformarlas en representaciones más adecuadas para su análisis.

Detección de Esquinas

Las esquinas representan puntos clave en una imagen donde los gradientes de intensidad cambian significativamente en múltiples direcciones. En este código se utiliza:

Detector de esquinas Harris: Método que evalúa cambios de intensidad en ventanas de la imagen para identificar esquinas. Este método es rápido y efectivo para detectar características clave en una imagen.

Agrupación y Representación Geométrica

Una vez detectadas las esquinas, se agrupan para formar polígonos:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Es un algoritmo de agrupamiento basado en densidad que identifica grupos de puntos cercanos y trata los puntos aislados como ruido. Es ideal para trabajar con conjuntos de datos espaciales, como las coordenadas de las esquinas detectadas.

Convex Hull:
Es una envolvente convexa que representa la forma geométrica más simple que puede contener un conjunto de puntos. Este método se usa para aproximar las esquinas detectadas como polígonos.

Procesamiento Digital de Imágenes

El procesamiento digital de imágenes consiste en transformar y analizar imágenes utilizando algoritmos computacionales. En este contexto, los conceptos principales son:

- **Escala de grises:** Permite simplificar el análisis al reducir la imagen de un espacio de color RGB a intensidades de un solo canal, facilitando la detección de características.
- **Transformaciones geométricas y análisis:** Estas técnicas permiten identificar puntos clave, contornos y estructuras en una imagen.

Detección de Esquinas

Las esquinas representan puntos clave en una imagen donde los gradientes de intensidad cambian significativamente en múltiples direcciones. En este código se utiliza:

- **Detector de esquinas Harris:** Método que evalúa cambios de intensidad en ventanas de la imagen para identificar esquinas. Este método es rápido y efectivo para detectar características clave en una imagen.

Agrupación y Representación Geométrica

Una vez detectadas las esquinas, se agrupan para formar polígonos:

DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Es un algoritmo de agrupamiento basado en densidad que identifica grupos de puntos cercanos y trata los puntos aislados como ruido. Es ideal para trabajar con conjuntos de datos espaciales, como las coordenadas de las esquinas detectadas.

Convex Hull:

Es una envolvente convexa que representa la forma geométrica más simple que puede contener un conjunto de puntos. Este método se usa para aproximar las esquinas detectadas como polígonos.

Aproximación de polígonos:

Con el método cv2.approxPolyDP, se simplifican contornos para reducir su complejidad, eliminando puntos redundantes y conservando la forma general.

Detección de Agujeros

Un agujero dentro de un polígono es una región que presenta una intensidad significativamente diferente al resto del área del polígono. Para detectarlos:

- **Máscaras de intensidad:**
- Se evalúan los valores de intensidad dentro del polígono en relación con el promedio, identificando desviaciones como posibles agujeros.
- **Contornos:**
  Se identifican los bordes de estas áreas mediante algoritmos de detección de contornos, como cv2.findContours.

Introducción

El código combina técnicas de procesamiento de imágenes, como el detector de esquinas Harris y el método de agrupamiento DBSCAN, para identificar estructuras geométricas y analizar irregularidades internas, como agujeros. Estas herramientas son ampliamente utilizadas en áreas como el control de calidad, la inspección de superficies, y la detección de patrones y características clave.

Desarrollo

El programa diseñado emplea un enfoque estructurado para procesar imágenes y detectar características geométricas relevantes como esquinas, polígonos y agujeros. Este desarrollo consta de diversas etapas que combinan técnicas de procesamiento digital de imágenes y agrupamiento espacial.

Cálculos y Resultados

P09 poligonos y agujeros.py

La primera etapa consiste en cargar la imagen de entrada y convertirla a escala de grises, reduciendo su complejidad al trabajar con un solo canal de intensidad. Esta transformación es fundamental para realizar cálculos de gradientes y detectar cambios de intensidad que definen características como esquinas.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

# Cargar la imagen
image_path = "imagen.png"  # Cambia esto por la ruta de tu imagen
image = cv2.imread(image_path)

if image is None:
    raise ValueError("No se pudo cargar la imagen. Verifica la ruta.")
```

Este fragmento de código realiza la detección de esquinas en una imagen utilizando el algoritmo Harris Corner Detection, transformando primero la imagen a escala de grises para simplificar el análisis. Una vez detectadas, las esquinas se resaltan visualmente dibujando círculos sobre su ubicación en la imagen original y creando una versión binaria que contiene únicamente las esquinas. Posteriormente, los puntos correspondientes a las esquinas se procesan mediante el algoritmo de agrupamiento DBSCAN, que clasifica las esquinas en grupos basados en la proximidad espacial, lo que permite identificar regiones compactas de puntos que pueden corresponder a estructuras como polígonos. Los resultados se visualizan y se almacenan para análisis posterior.

```
# Convertir la imagen a escala de grises
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Aplicar el detector de esquinas Harris
gray_image = np.float32(gray_image)
dst = cv2.cornerHarris(gray_image, blockSize=2, ksize=3, k=0.04)

# Dilatar las esquinas para visualizarlas
dst = cv2.dilate(dst, None)

# Umbral relativo para considerar esquinas
threshold = 0.02 * dst.max()
esquinas = np.argwhere(dst > threshold)

# Dibujar círculos en las esquinas detectadas
imagen_con_esquinas = image.copy()
solo_esquinas = np.zeros_like(gray_image)

for esquina in esquinas:
    y, x = esquina  # Coordenadas (y, x)
    cv2.circle(imagen_con_esquinas, (x, y), 5, (0, 0, 255), 2)  # Esquinas
en rojo
    cv2.circle(solo_esquinas, (x, y), 5, (255), 2)

cv2.imshow('Solo Esquinas Detectadas', solo_esquinas)
# Mostrar la imagen con las esquinas detectadas
plt.figure(figsize=(10, 5))
plt.imshow(cv2.cvtColor(imagen_con_esquinas, cv2.COLOR_BGR2RGB))
plt.title('Esquinas Detectadas')
plt.axis('off')
plt.show()

# Guardar la imagen resultante
cv2.imwrite('Imagen Esquinas Detectadas.png', imagen_con_esquinas)
```

```
# Convertir las esquinas a formato adecuado para agrupar
puntos = np.array([p[::-1] for p in esquinas], dtype=np.float32)  # Cambiar
a (x, y)

clustering = DBSCAN(eps=10, min_samples=4).fit(puntos)  # Ajusta 'eps' según
la distancia esperada entre puntos
labels = clustering.labels_
```

Este segmento de código identifica y dibuja polígonos en una imagen utilizando los grupos de esquinas previamente clasificados por DBSCAN. Para cada grupo de puntos (esquinas) asociado a una etiqueta válida, se calcula su envolvente convexa (convex hull) para delimitar su contorno externo. Luego, el contorno se simplifica mediante aproximación poligonal (cv2.approxPolyDP), que reduce el número de vértices preservando la forma. Los polígonos resultantes se dibujan tanto en la imagen original como en una versión binaria dedicada a resaltar únicamente los polígonos detectados. Finalmente, se visualizan los resultados y se guarda la imagen procesada, junto con una lista que almacena los polígonos para un análisis posterior.

```
# Dibujar polígonos detectados
imagen_con_poligonos = image.copy()
solo_poligonos = np.zeros_like(gray_image)
poligonos = []

for label in set(labels):
    if label == -1:  # Ignorar ruido
        continue

    # Extraer puntos de un grupo
    grupo_puntos = puntos[labels == label]

    # Encontrar el contorno aproximado del polígono
    hull = cv2.convexHull(grupo_puntos.astype(np.int32))
    epsilon = 0.02 * cv2.arcLength(hull, True)
    aproximado = cv2.approxPolyDP(hull, epsilon, True)

    # Dibujar el polígono
    cv2.polylines(imagen_con_poligonos, [aproximado], True, (0, 255, 0), 2)
    cv2.polylines(solo_poligonos, [aproximado], True, (255), 2)
    poligonos.append(aproximado)

cv2.imshow('Solo poligonos detectadas', solo_poligonos)
# Mostrar la imagen con polígonos detectados
plt.figure(figsize=(10, 5))
```

```
plt.imshow(cv2.cvtColor(imagen_con_poligonos, cv2.COLOR_BGR2RGB))
plt.title(f'Polígonos Detectados: {len(poligonos)}')
plt.axis('off')
plt.show()

# Guardar la imagen resultante
cv2.imwrite('Imagen poligonos detectados.png', imagen_con_poligonos)
```

Este fragmento de código detecta y resalta los agujeros dentro de los polígonos previamente identificados en una imagen. Para cada polígono, se genera una máscara que lo delimita, y se calculan las intensidades promedio de los píxeles dentro de este. Basándose en un rango de intensidad definido alrededor del promedio, se crea otra máscara que identifica regiones dentro del polígono con intensidades significativamente diferentes, consideradas como posibles agujeros. Los contornos de estos agujeros se detectan y se dibujan en rojo, mientras que los polígonos se resaltan en verde sobre una copia de la imagen original. Además, se cuenta el número total de agujeros detectados, y los resultados se visualizan y almacenan.

```
# Detectar agujeros
imagen_con_resultados = image.copy()
solo_agujeros = np.zeros_like(gray_image)
agujeros = 0

for poligono in poligonos:
    # Crear una máscara para el polígono
    mask = np.zeros_like(gray_image, dtype=np.uint8)
    cv2.drawContours(mask, [poligono], -1, 255, thickness=cv2.FILLED)

    # Extraer intensidades dentro del polígono
    intensidades = gray_image[mask == 255]
    promedio = np.mean(intensidades)

    # Crear máscara para agujeros basados en la diferencia de intensidad
    lower_bound = promedio - 50
    upper_bound = promedio + 50
    mask_agujeros = ((gray_image < lower_bound) | (gray_image >
upper_bound)) & (mask == 255)

    # Detectar contornos de los agujeros
    contornos, _ = cv2.findContours(mask_agujeros.astype(np.uint8),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contorno in contornos:
```

```
        # Dibujar el agujero en rojo
        cv2.drawContours(imagen_con_resultados, [contorno], -1, (0, 0, 255),
2)
        cv2.drawContours(solo_agujeros, [contorno], -1, (255), 2)
        agujeros += 1

    # Dibujar el polígono en verde
    cv2.drawContours(imagen_con_resultados, [poligono], -1, (0, 255, 0), 2)

cv2.imshow('Imagen poligonos y agujeros', solo_agujeros)
```

Este código muestra visualmente los resultados de la detección de polígonos y agujeros en una imagen, incluyendo el conteo total de cada uno. Los resultados se guardan como una imagen procesada con polígonos resaltados en verde y agujeros en rojo. Adicionalmente, se generan y guardan matrices en formato CSV que representan las imágenes binarizadas de las esquinas, los polígonos y los agujeros detectados, permitiendo su análisis o uso posterior en otros procesos. Estos archivos se acompañan de mensajes de confirmación para garantizar que los datos se han almacenado correctamente.

```
# Mostrar la imagen con polígonos y agujeros detectados
plt.figure(figsize=(10, 5))
plt.imshow(cv2.cvtColor(imagen_con_resultados, cv2.COLOR_BGR2RGB))
plt.title(f'Polígonos: {len(poligonos)}, Agujeros: {agujeros}')
plt.axis('off')
plt.show()

# Guardar la imagen resultante
cv2.imwrite('Imagen poligonos y agujeros.png', imagen_con_resultados)

# Guardar la matriz de la imagen en escala de grises en un archivo CSV
np.savetxt('Matriz Solo esquinas.csv', solo_esquinas, delimiter=',',
fmt='%d')
print("La matriz de la imagen binarizada se ha guardado en 'Matriz Solo
esquinas.csv'.")

# Guardar la matriz de la imagen en escala de grises en un archivo CSV
np.savetxt('Matriz Solo poligonos.csv', solo_poligonos, delimiter=',',
fmt='%d')
print("La matriz de la imagen binarizada se ha guardado en 'Matriz Solo
poligonos.csv'.")

# Guardar la matriz de la imagen en escala de grises en un archivo CSV
np.savetxt('Matriz Solo agujeros.csv', solo_agujeros, delimiter=',',
fmt='%d')
```

```
print("La matriz de la imagen binarizada se ha guardado en 'Matriz Solo
agujeros.csv'.")
```

imagen original

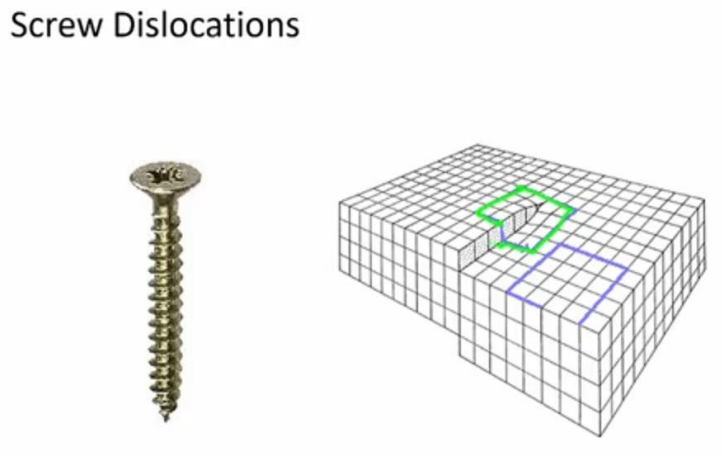
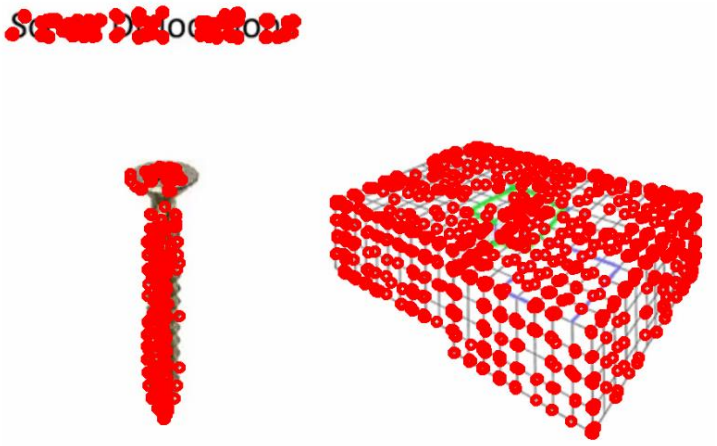


Imagen esquinas detectadas.png



imágenes polígonos detectados.png

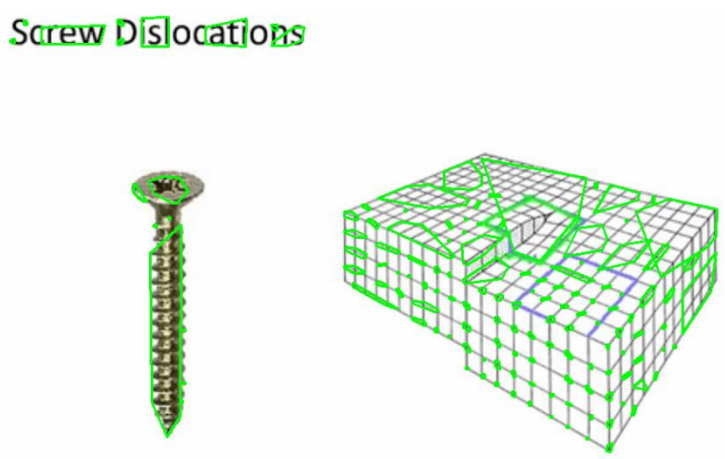
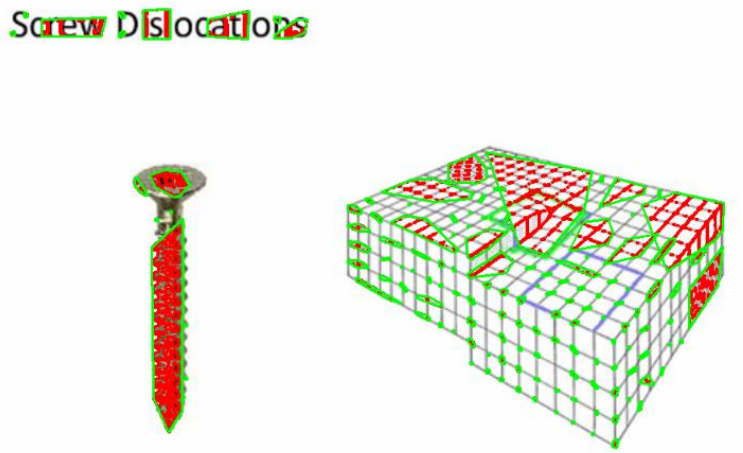


Imagen polígonos y agujeros.png



Conclusiones

En resumen, el código desarrollado demuestra ser efectivo para identificar esquinas, polígonos y agujeros en imágenes, implementando técnicas avanzadas como el algoritmo de Harris para la detección de esquinas y el método de agrupamiento DBSCAN para localizar regiones de interés.
La representación de los polígonos se logra mediante el cálculo de la envolvente convexa, mientras que la detección de agujeros se basa en el análisis de las variaciones de intensidad de los píxeles, permitiendo un estudio estructural detallado de las formas presentes en la imagen.
Los resultados se presentan de manera clara y se almacenan tanto en imágenes como en matrices binarizadas, lo que facilita su integración en aplicaciones futuras. Este enfoque constituye una base robusta para el desarrollo de análisis más avanzados en el ámbito de la visión por computadora, como el reconocimiento de formas o la detección de imperfecciones.

Bibliografía

OpenCV modules. OpenCV. (n.d.). https://docs.opencv.org/4.x/index.html

NumPy Org. (2024, June 17). NumPy. https://numpy.org/

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing, Global Edition*. Pearson Higher Ed.

Tangentes, cuerdas y arcos – Círculos y Pi. Mathigon. Retrieved November 24, 2024, from https://es.mathigon.org/course/circles/tangets-chords-arcs