# 

# Trabajo Práctico 2

## Lucía Masciangelo, Francisco Rodríguez y Barros, Julieta Texier

## 11-06-2024

## Procesamiento de Imágenes I

## Tecnicatura en Inteligencia Artificial

## Requisitos previos para correr los ejercicios:

1. Clonar el repositorio <https://github.com/franciscoryb1/PDI-TUIA-RodriguezYBarros>
2. Acceder a la carpeta: cd .\TP2
3. Tener python instalado en el entorno a utilizar. Se puede descargar desde <https://www.python.org/downloads/>
4. Instalar las librerías que serán usadas en los ejercicios:
   * OpenCV: pip install opencv-python
   * Matplotlib: pip install matplotlib
   * numpy: pip install numpy

## Ejercicio 1: Detección y clasificación de componentes electrónicos

Primero importamos las librerías y creamos una función para mostrar las imágenes (imshow).

Cargamos la imagen 'Placa.png' que se encuentra en la misma carpeta donde va a correr el script.

Convertimos la imagen a escala de grises, aplicamos el filtro de suavizado MedianBlur y probamos con varios umbrales de Canny para detectar bordes. (elegimos los umbrales 0.2\*255 y 0.8\*255 para encontrar el chip y los capacitores). Después aplicamos Gradiente morfológico con un kernel de (5,5).

Y obtuvimos los siguientes resultados:

Texto

Descripción generada automáticamente con confianza baja

Después calculamos los componentes conectados y generamos un bounding box para encontrar el chip y otros boundings box para cada uno de los capacitores.

Una vez que tenemos detectado el chip, lo guardamos en una imagen nueva llamada chip.png.

Un circuito electrónico

Descripción generada automáticamente con confianza media

Hacemos lo mismo con los capacitores, pero sumándole una etiqueta que identifique que tamaño tiene el capacitor (también guardamos una imagen llamada capacitores según tamaños, y otra imagen que sea solo los capacitores sin distinción de tamaño)

Un circuito electrónico

Descripción generada automáticamente con confianza mediaPantalla de video juego

Descripción generada automáticamente con confianza media

Ahora, para poder detectar las resistencias sin problemas vamos a “eliminar” de la imagen original a los capacitores y al chip, para eso creamos una imagen negra y con componentes conectados detectamos la parte de la imagen donde se encuentran el chip y los capacitores, y después pintamos de la imagen original de negro esos pixeles

Imagen que contiene circuito, computadora

Descripción generada automáticamente

Con esta imagen obtenida volvemos a hacer el proceso de convertir la imagen a escala de grises, binarizamos la imagen, aplicamos el filtro de suavizado MedianBlur y probamos con varios umbrales de Canny para detectar bordes (elegimos los umbrales 0.2\*255 y 0.6\*255 para encontrar las resistencias) aplicamos clausura con el kernel (20,1) y por último, aplicamos Gradiente morfológico con un kernel de (5,5).

Diagrama

Descripción generada automáticamente

Después calculamos los componentes conectados y aplicamos bounding box para encontrar las resistencias.

Un circuito electrónico

Descripción generada automáticamente con confianza media

Por último, agregamos las resistencias a la imagen con fondo negro en la que ya tenemos el chip y los capacitores y creamos otra solo con las resistencias, en ese mismo código pusimos un contador para después imprimir por consola la cantidad de resistencias encontradas.

En resumen, la salida del ejercicio es:

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## Ejercicio 2: Detección de patentes

Al igual que en el ejercicio 1 importamos las librerías necesarias y creamos una función para mostrar las imágenes (imshow).

Contamos con una carpeta, donde se encuentran 12 imágenes de autos, dónde el objetivo es detectar la patente y sus componentes (letras y números).

En primer lugar creamos una función llamada recortar patente, a la que le pasamos una img previamente cargada, y dentro de ella, convertimos esa img a escala de grises, luego aplicamos un GaussianBlur (probamos con distintos umbrales y tipos de blur), luego aplicamos canny, sobre la img blurrred, donde también probamos con distintos umbrales hasta quedarnos con (250,300). Después de eso, aplicamos técnicas de morfología, para hacer una clusura y cerrar algunos huecos que nos causaban problemas.

Nota: en esta función probamos haciendo umbralado, y otros métodos de morfología pero no funcionaban bien.

Con la img ya procesada, aplicamos la función que detecta componentes conectados, para detectar la patente. Filtramos por área y por relación de aspecto (si el área es mayor a 300, y además el ancho es mas grande que el alto, entonces detecta la patente). Con las coordenadas x, y, el ancho y alto, recortamos el sector dónde se encuentra la patente, en la img original (no la procesada).

Una vez que tenemos el sector de la patente segmentada, se lo pasamos a la siguiente función, llamada detectar\_componentes, que también toma como entrada una img, le aplicamos la función que realizamos anteriormente y te devuelve esa img recortada con las letras y los números de la patente recuadrados en un bounding box. Nuevamente procesamos la img, pero esta vez, lo que hacemos es pasarla a escala de grises, binarizarla con un umbral de 113 (fuimos probando con distintos y éste es el que mejor nos funcionó), y nuevamente detectar componentes conectados, pero con una conectividad de 4. (elegimos 4 porque con 8 conectados, había algunas diagonales que se conectaban con el borde de las patentes), sin embargo en algunas imágenes había, por ejemplo, los números 2, que fallan. Igualmente, con 4 conectados es más lo que ganamos que lo que perdemos en detección.

Por último, leemos una por una las img de la carpeta patentes, y a cada una le aplicamos la función detectar\_componentes, para que nos muestre las img de las patentes segmentadas con los bounding boxs realizados.

Nota: con el umbral 113 detectamos los siguientes componentes:

# img1: 4 de 6

# img2: 6 de 6

# img3: 6 de 6

# img4: 6 de 6

# img5: 6 de 6

# img6: 5 de 6

# img7: 5 de 6

# img8: 4 de 6

# img9: 6 de 6

# img10: 6 de 6

# img11: 1 de 6

# img12: 6 de 6

Un total de 61 caracteres detectados de un total de 72 (84,72%)

También probamos con un umbral de 109 para intentar solucionar algunos problemas y conseguimos esto:

# img1: 4 de 6

# img2: 6 de 6

# img3: 6 de 6

# img4: 6 de 6

# img5: 5 de 6

# img6: 6 de 6

# img7: 6 de 6

# img8: 3 de 6

# img9: 6 de 6

# img10: 4 de 6

# img11: 1 de 6

# img12: 6 de 6

Un total de 59 caracteres detectados de un total de 72 (81,94%)