# 

# 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 aplicamos bounding box para encontrar el chip por un lado y los capacitores por otro.

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

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 (que también guardamos en una imagen llamada capacitores según tamaños, y otra imagen que sea solo los capacitores)

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: Corrección de multiple choice

Utiliza las siguientes librerías: cv2, numpy y matplotlib.

Archivos necesarios: Los 5 archivos que contienen las imágenes de los multiple choice que están en la misma carpeta donde va a correr el script.

Este ejercicio se divide en 4 ítems:

1. primero tenemos que poder dividir la imagen de los multiple choice en la cantidad de filas que tiene para poder analizar después si la respuesta es correcta o no para cada pregunta.

Para ello realizamos una función que tiene como argumento la imagen. Binarizo esa imagen para que me quede con 2 colores (blanco y negro). Utilizo el umbral 244 ya que si utilizo un umbral mayor me genera ruido dentro de los círculos seleccionados, y si utilizo un umbral menor me genera ruido alrededor de los cirulos con las opciones; dejando una imagen con fondo negro y con los círculos con las opciones en blanco. Saco el encabezado donde están los nombres, fecha, legajo, etc ya que para esta parte no lo necesito, es decir, recorto la imagen (puedo poner una coordenada específica ya que el encabezado en todos los exámenes está en la misma posición). Luego, detectamos las filas que tienen al menos un valor igual a 255 (blanco) e identificamos sus índices, modifico y reordeno esos índices para poder detectar los renglones, agrupándolos de a 2. Por último, obtengo esos renglones que los guardo en una lista, cada uno con un índice y su imagen.

Ahora que tenemos los renglones, en cada uno vamos a identificar si la respuesta fue correcta o no. Para ello primero le pasamos la función cv2.findContours(img, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) para detectar los círculos en cada renglón. Luego detectamos el cuadrado delimitador del circulo para poder contar correctamente la cantidad de píxeles blancos que hay en cada circulo, y así poder detectar cual es la opción seleccionada.

Después definimos por cada renglón, si no hay respuestas seleccionadas, si hay más de una respuesta seleccionada, o si hay una sola.

La cantidad de píxeles blancos de los círculos sin seleccionar son: {E:179, D:167, C:164, B:187, A:195}, y de los círculos cuando están seleccionados tienen como min 345 pixeles blancos.

* Si la cantidad de pixeles blancos que hay en cada círculo es menor a 200 significa que no seleccionó ninguna opción y por ende respondió mal;
* Si la suma de los pixeles es mayor a 1100 significa que seleccionó más de una respuesta y por ende también respondió mal;
* Si no pasa ninguno de los dos casos anteriores, significa que eligió una sola opción, y en este caso tenemos dos opciones: que haya elegido bien la respuesta o que no.
* Que haya elegido bien la respuesta: cuando la respuesta que eligió sea la misma a la correspondiente en la lista de respuesta correctas.
* Que haya elegido mal la respuesta: cuando la respuesta que eligió no sea la misma a la correspondiente en la lista de respuestas correctas.

Para correrlo tengo que pasarle a mi función **corrección(img)** la imagen que quiero calificar.

Salida: Al correr el programa me devolverá si resolvió bien o mal cada ítem del multiple choice. Además, devuelve cuantes respuestas fueron correctas y si la persona aprobó o no, estos datos los vamos a usar en el apartado C.

1. LES DEJO ESTE TMB PARA HACER
2. Este ejercicio solo pide que analicemos los resultados de mis funciones anteriores, entonces podemos decir que:

Imagen\_1: Juan Perez aprobó con 20 respuestas correctas

Imagen\_2: Jorge desaprobó con 7 respuestas correctas

Imagen\_3: Pedro Monti desaprobó con 5 respuestas correctas

Imagen\_4: Alfredo desaprobó con 8 respuestas correctas

Imagen\_5: María Suarez no tuvo respuestas correctas, por ende desaprobó

1. El objetivo es devolver una imagen con los nombres de los que hicieron los exámenes y con alguna clasificación de aprobado o no aprobado.

Para eso primero creamos una imagen toda en blanco, y tomando la lista de imágenes que guardamos con los crop de los nombres, va iterando en esa lista para ir pegando en nuestra imagen nueva un nombre abajo del otro. Al mismo tiempo va escribiendo al lado del nombre si la persona aprobó o no aprobó.

Salida: Devuelve una imagen con los nombres de los 5 exámenes y si aprobaron o no.