Trabajo Práctico 1

Lucía Masciangelo, Francisco Rodríguez y Barros, Julieta Texier

30-04-2024

Procesamiento de Imágenes I

Tecnicatura en Inteligencia Artificial

Requisitos previos para correr los ejercicios:

- Clonar el repositorio https://github.com/franciscoryb1/PDI-TUIA-RodriguezYBarros
- Acceder a la carpeta: cd .\TP1
- Tener python instalado en el entorno a utilizar. Se puede descargar desde https://www.python.org/downloads/
- Instalar las librerías que serán usadas en los ejercicios:
    - OpenCV: pip install opencv-python
    - Matplotlib: pip install matplotlib
    - numpy: pip install numpy

Ejercicio 1: Ecualización local de histograma

Este ejercicio utiliza la librería OpenCV para procesar imágenes y realizar la ecualización local del histograma y la librería matplotlib.pyplot para devolver la imagen.

Primero cargamos la imagen 'Imagen_con_detalles_escondidos.tif' que se encuentra en la misma carpeta donde va a correr el script.

Creamos una función para ecualizar el histograma, a la cual le vamos a pasar la imagen y un tamaño de ventana.

Dentro de la función lo que hacemos es ir iterando sobre la imagen moviéndonos en ancho y en alto a lo largo de la imagen para obtener una ventana y en ella aplicar la ecualización local con la función cv2.equalizeHist(ventana). Una vez que se hizo la ecualización colocamos los píxeles ecualizados en la imagen de salida.

Esto lo hacemos para diferentes tamaños de ventana aplicando la función de la siguiente manera:

por ejemplo, **img_salida1 = ecualizacion_local_histograma(img,10)** donde le decimos que el tamaño de la ventana es 10 y que la imagen a utilizar es img

Salida: Al correr el programa se abrirá un visualizador con la imagen original y los resultados obtenidos con diferentes valores de ventana, en la cuales se puede hacer zoom para apreciar mejor las mismas.

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:

- **A)** 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 círculo para poder contar correctamente la cantidad de píxeles blancos que hay en cada círculo, 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 píxeles 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 píxeles 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 cuantas respuestas fueron correctas y si la persona aprobó o no, estos datos los vamos a usar en el apartado C.

- **B)** En el apartado b se requiere que con la misma imagen de entrada, se validen los datos del encabezado y mostrar la validación por pantalla.

En primer lugar desarrollamos una función que acepta como único parámetro de entrada una imagen que contenga directamente el contenido del campo que se desea analizar y su salida es un diccionario que contiene:

- Cantidad de caracteres.

- Cantidad de espacios.

- Cantidad de palabras.

Con esta salida, posteriormente hacemos las validaciones para corroborar si cumple o no con los requisitos de cada campo.

Dentro de la función, lo primero que hacemos es aplicar un umbral adaptativo Gaussiano, utilizando una ventana de 11x11 píxeles y luego invertimos el resultado del umbral binario, esto para que los píxeles que superen el umbral sean establecidos en el valor máximo (255) y los que estén por debajo sean establecidos en cero.

- Usando **cv2.findContours** encontramos los contornos de la imagen binaria.

- Se crea una imagen vacía del mismo tamaño que la imagen original utilizando **np.zeros_like.**

- Usamos **cv2.drawContours** para dibujar los contornos obtenidos en la nueva imagen generada con un grosor de línea negativo rellenando completamente los contornos con el color especificado (255, 255, 255).

Una vez realizado el tratamiento de la imagen para que cada uno de los caracteres quede bien marcado, pasamos a detectar componentes conectados usando la funcion **cv2.connectedComponentsWithStats** analizando todos los píxeles adyacentes (8 conectados).

Con los resultados obtenidos, nos centramos en la variable **stats** que contiene una lista de arrays, donde cada array es un componente de la imagen, detallando su primera coordenada en x, en y, su área, ancho y alto del mismo.

Lo primero fue ordenar la lista stats y filtrarla quedándonos con los componentes cuya área era menor a 50 pixeles, esto para descartar el componente correspondiente al fondo.

Para detectar las palabras, definimos un umbral máximo de separación entre letras en 9 pixeles, y comparando la distancia entre una letra y su siguiente calculada restando las coordenadas del eje x de la letra siguiente y la actual evaluamos si supera o no el umbral, siendo que, si lo supera, se trata de un espacio de separación entre palabras.

Si detectamos, por ejemplo, 1 espacio, quiere decir que tenemos 2 palabras.

Todos estos valores obtenidos (Número de caracteres, Número de Espacios, Cantidad de palabras) se almacenan en un diccionario y lo retorna la función, siendo evaluados posteriormente.

- **C)** 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

- **D)** 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.