Entrega 6

Rectificación

Deshaz la deformación de perspectiva de la imagen de un plano para medir distancias (tomando manualmente referencias conocidas). Por ejemplo, mide la distancia entre las monedas en `coins.png` o la distancia a la que se realiza el disparo en `gol-eder.png`. Las coordenadas reales de los puntos de referencia y sus posiciones en la imagen deben pasarse como parámetro en un archivo de texto. Aunque puedes mostrar la imagen rectificada para comprobar las operaciones, debes marcar los puntos y mostrar el resultado sobre la imagen original. Verifica los resultados con imágenes originales tomadas por ti.

Para este ejercicio he implementado un fichero que mide la distancia deshaciendo la deformación de perspectiva de la imagen de un plano. Para ello, he creado el fichero `rectificacion.py` el cual se descompone en varias funciones las cuales procederé a explicar.

En primer lugar he creado una función llamada `cargar_txt()` la cual, como bien describe su nombre, carga el contenido del fichero `.txt` que contiene las referencias de puntos de la imagen y guarda los puntos de referencia de la imagen (en pixeles) y los reales (en centímetros) en dos variables de tipo `np.array`.

In [ ]
```python
def cargar_txt(ruta):
    """Carga los puntos de referencia desde un archivo de texto."""
    img_pts = []
    real_pts = []
    try:
        with open(ruta, 'r') as f:
            for linea in f:
                linea = linea.strip()
                if linea and not linea.startswith('#'):
                    parts = linea.split(",")
                    if len(parts) == 4:
                        try:
                            img_x, img_y, real_x, real_y = map(float, parts)
                            img_pts.append([img_x, img_y])
                            real_pts.append([real_x, real_y])
                        except ValueError:
                            print(f"Advertencia: Ignorando línea mal formada: {linea}")
                    else:
                        print(f"Advertencia: Ignorando línea con número incorrecto de valores: {linea}")
    except FileNotFoundError:
        print(f"Error: No se encontró el archivo de referencias: {ruta}")
        return None, None
    except Exception as e:
        print(f"Error leyendo el archivo de referencias: {e}")
        return None, None

    if len(img_pts) < 4:
        print("Error: Se necesitan al menos 4 puntos de referencia.")
        return None, None

    # Convertir a np.arrays
    img_final = np.array(img_pts, dtype=np.float32)
    real_final = np.array(real_pts, dtype=np.float32)

    return img_final, real_final
```

La siguiente función, llamada `seleccionar_punto()`, ante el evento del click del ratón, y las coordenada `x,y` donde se ha hecho click, si el numero de puntos que se han seleccionado hasta el momento es menor que 2, añade el nuevo punto en la lista de puntos y lo dibuja en la imagen.

In [ ]
```python
def seleccionar_punto(evento, x, y, flags, param):
    """Callback del ratón para seleccionar puntos."""
    global puntos_a_medir, imagen_a_pintar

    if evento == cv2.EVENT_LBUTTONDOWN:
        if len(puntos_a_medir) < 2:
            puntos_a_medir.append((x, y))
            # Dibujar punto en la copia de la imagen
            cv2.circle(imagen_a_pintar, (x, y), 5, (203, 192, 255), -1)
            cv2.circle(imagen_a_pintar, (x, y), 6, (0, 0, 0), 1) # Borde negro
            cv2.imshow("Rectificación de Perspectiva - Selección de Puntos", imagen_a_pintar)
            print(f"Punto {len(puntos_a_medir)} seleccionado: ({x}, {y})")
        else:
            print("Ya has seleccionado 2 puntos. Pulsa 'r' para reiniciar.")
```

Por último, la función principal, el `main()`. En él se cargan la imagen y puntos de referencia originales, y tras ello, se calcula la matriz de homografía.

La matriz de homografía permite mapear puntos de una imagen a otra cuando ambas imágenes ven el mismo plano desde diferentes puntos de vista. Es decir, es la que permite pasar de coordenadas en el plano de la imagen a coordenadas reales.

Después, guardo una copia de la imagen original para poder dibujar los puntos y líneas sobre ella, creo el frame con su nombre y por último, creo un callback para recibir el evento del click del ratón y llamar a la función anterior.

Una vez inicializado todo, entramos en un bucle infinito, el cual hice con un `while True:` y no con `for key, frame in autoStream():` ya que no necesito capturar la cámara en ningún momento.

Una vez dentro del bucle, muestro la copia de la imagen y guardo si se ha pulsado alguna tecla (`q` para terminar o `r` para reiniciar los puntos seleccionados).

Finalmente, si el número de puntos seleccionados es 2, a partir de los puntos `a` y `b` en la imagen, creo el array con los puntos en la imagen y con él, el array con los puntos reales, del que se obtendrán los puntos `a` y `b` reales. Una vez conseguidos los puntos reales, se calcula la distancia entre ellos, se dibuja la línea y se escribe la distancia sobre dicha línea.

In [ ]
```python
def main(ruta_imagen, ruta_txt):
    global puntos_a_medir, imagen_a_pintar

    # Cargar imagen original
    img_original = cv2.imread(ruta_imagen)
    if img_original is None:
        print(f"Error: No se pudo cargar la imagen: {ruta_imagen}")
        return

    # Cargar puntos de referencia
    pts_img, pts_real = cargar_txt(ruta_txt)
    if pts_img is None or pts_real is None:
        return

    print(f"Cargados {len(pts_img)} puntos de referencia.")

    # Calcular homografía
    homografia, _ = cv2.findHomography(pts_img, pts_real, cv2.RANSAC, 5.0)
    if homografia is None:
        print("Error: No se pudo calcular la homografía. Verifica los puntos de referencia.")
        return

    # Preparar para interacción
    imagen_a_pintar = img_original.copy()
    nombre_frame = "Rectificación de Perspectiva - Selección de Puntos"
    cv2.namedWindow(nombre_frame)
    cv2.setMouseCallback(nombre_frame, seleccionar_punto)

    print("\n------ Ayuda ------")
    print("Pulsa 'r' para reiniciar la selección de puntos o 'q' para salir.")
    print("Haz click en dos puntos en la imagen para medir la distancia real entre ellos.")
    print("Los puntos seleccionados se dibujarán en la imagen.")
    print("--------------------\n")

    while True:

        cv2.imshow(nombre_frame, imagen_a_pintar)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('q'):
            break
        elif key == ord('r'):
            print("Reiniciando puntos.")
            puntos_a_medir = []
            imagen_a_pintar = img_original.copy() # Restaurar imagen limpia

        # Si se han seleccionado dos puntos
        if len(puntos_a_medir) == 2:
            a, b = puntos_a_medir # Descomponer los puntos seleccionados en las puntos a y b

            array_puntos_imagen = np.array([[a, b]], dtype=np.float32)

            array_puntos_real = cv2.perspectiveTransform(array_puntos_imagen, homografia)

            if array_puntos_real is None:
                print("Transformación de perspectiva incompleta, error al tranformar.")
                puntos_a_medir = []
                imagen_a_pintar = img_original.copy()
                continue

            a_real = array_puntos_real[0, 0]
            b_real = array_puntos_real[0, 1]

            distancia = np.linalg.norm(a_real - b_real) # Distancia en centímetros

            cv2.line(imagen_a_pintar, a, b, (255, 0, 0), 2)

            x_texto = int((a[0] + b[0]) / 2)
            y_texto = int((a[1] + b[1]) / 2) - 10 # Un poco arriba
            texto = f"{distancia:.2f}cm" # Asume unidades de referencias.txt
            cv2.putText(imagen_a_pintar, texto, (x_texto, y_texto),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2, cv2.LINE_AA) # Borde blanco
            cv2.putText(imagen_a_pintar, texto, (x_texto, y_texto),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv2.LINE_AA) # Texto negro

            print(f"Distancia entre los dos puntos: {distancia:.2f} centímetros")

            # Muestro resultado final
            cv2.imshow(nombre_frame, imagen_a_pintar)

            # Limpio los puntos para poder seleccionar un nuevo par
            puntos_a_medir = []

            print("Pulsa 'r' para reiniciar los puntos, 'q' para salir o selecciona 2 nuevos puntos.")

    cv2.destroyAllWindows()
    print("Programa finalizado.")
```

Veamos un ejemplo de su ejecución.

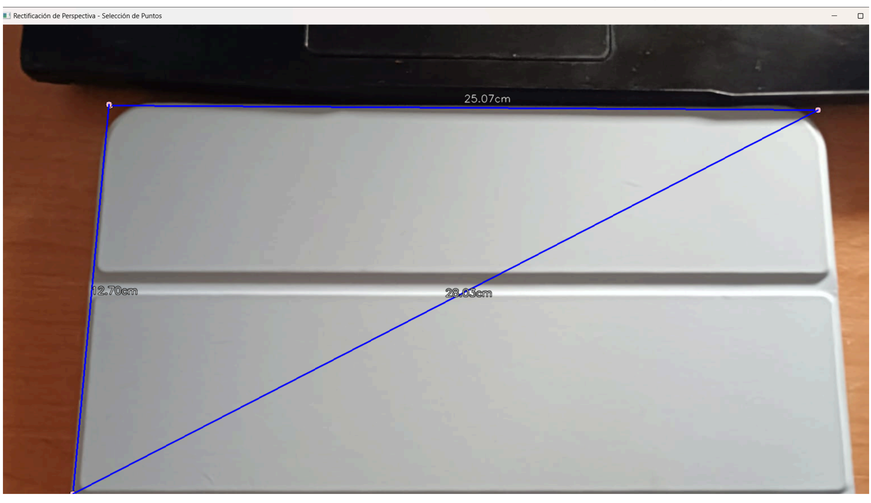
En primer lugar, explicar que para la ejecución de este programa se necesitan dos parámetros, la `imagen` de referencia y el fichero `.txt` con los `puntos` de referencia.

Su ejecución desde el directorio `/Entregas/Entrega6/src` es la siguiente:

```
python rectificacion.py ../images/ipad.jpg ./points.txt
```

Ahora si, pasamos al caso práctico.

Sabiendo que las medidas de un ipad en horizontal son 24.86cm de ancho por 17.95cm de alto, y seleccionando aproximadamente dos tercios de alto y todo el ancho, además de la hipotenusa formando un triángulo, obtenemos el siguiente resultado.



Como se puede observar, el triángulo esta formado por el ancho de 25.07cm, el alto de 12.70cm, y la hipotenusa resultante es de 28.03cm. Si calculamos la hipotenusa de un triángulo rectángulo con catetos de dichas longitudes, el resultado es de 28.10cm, lo cual difiere minimamente del resultado obtenido en la imagen, seguramente debido a que los puntos no están seleccionados de manera precisa sino aproximada y por tanto el triángulo no es rectángulo.

Además, esta es la salida de la consola de comandos tras ejecutar el programa.

```
(base) c:\Users\Jormo\Desktop\Jorge\Universidad\4o\2oCuatri\VIA\DoscModificados\Entregas\Entrega6\src>python rectificacion.py ../images/ipad.jpg ./points.txt
Cargados 4 puntos de referencia.

------ Ayuda ------
Pulsa 'r' para reiniciar la selección de puntos o 'q' para salir.
Haz click en dos puntos en la imagen para medir la distancia real entre ellos.
Los puntos seleccionados se dibujarán en la imagen.
--------------------

Punto 1 seleccionado: (199, 143)
Punto 2 seleccionado: (1453, 151)
Distancia entre los dos puntos: 25.07 centímetros
Pulsa 'r' para reiniciar los puntos, 'q' para salir o selecciona 2 nuevos puntos.
Punto 1 seleccionado: (133, 833)
Punto 2 seleccionado: (197, 141)
Distancia entre los dos puntos: 12.70 centímetros
Pulsa 'r' para reiniciar los puntos, 'q' para salir o selecciona 2 nuevos puntos.
Punto 1 seleccionado: (133, 831)
Punto 2 seleccionado: (1453, 151)
Distancia entre los dos puntos: 28.03 centímetros
Pulsa 'r' para reiniciar los puntos, 'q' para salir o selecciona 2 nuevos puntos.
Programa finalizado.

(base) c:\Users\Jormo\Desktop\Jorge\Universidad\4o\2oCuatri\VIA\DoscModificados\Entregas\Entrega6\src>
```

Y este es el contenido del fichero `points.txt` para la imagen de referencia `ipad.jpg`.

```
205,144,0,0
108,1168,0,18
1455,147,25,0
1541,1159,25,18
```

Donde el formato que tiene es el siguiente:

Se necesitan 4 puntos de referencia de la imagen que vienen dados de la siguiente manera:

- Punto_1 (esquina superior izquierda (0,0)): Coordenada_x_en_pixeles, Coordenada_y_en_pixeles, Coordenada_x_en_centimetros, Coordenada_y_en_centimetros
- Punto_2 (esquina inferior izquierda (0,y)): Coordenada_x_en_pixeles, Coordenada_y_en_pixeles, Coordenada_x_en_centimetros, Coordenada_y_en_centimetros
- Punto_3 (esquina superior derecha (x,0)): Coordenada_x_en_pixeles, Coordenada_y_en_pixeles, Coordenada_x_en_centimetros, Coordenada_y_en_centimetros
- Punto_4 (esquina inferior derecha (x,y)): Coordenada_x_en_pixeles, Coordenada_y_en_pixeles, Coordenada_x_en_centimetros, Coordenada_y_en_centimetros

Donde cada uno de esos puntos se corresponde con las esquinas que delimitan un cuadrado/rectángulo de `x,y` centímetros reales. En el caso del ejemplo, el cuadrado formado es de 25x18 centímetros.