Procesamiento de Imágenes, Audio y Vídeo

Práctica 2:
Transformaciones geométricas

Manuel Alejandro Torrealba Torrealba, Alejandro Alemán Alemán

(Obligatorio – 5 puntos)

1a. Desarrollar una aplicación que lleve a cabo transformaciones de la imagen en tiempo real a través de una interfaz basada en trackbars.

En este primer ejercicio se implementó una aplicación interactiva en Python utilizando la biblioteca OpenCV (cv2), cuyo objetivo es permitir aplicar transformaciones geométricas básicas (traslación, rotación y escalado) a una imagen en tiempo real.

Para ello, se diseñó una única ventana denominada Transformaciones, en la que se integran todas las barras deslizantes (trackbars) necesarias para controlar los parámetros de cada transformación. Los valores de las trackbars se actualizan continuamente dentro de un bucle principal, aplicando los cambios de forma inmediata sobre la imagen.

Traslaciones

Las traslaciones horizontal y vertical se gestionan mediante las barras tx y ty, cuyo rango se define entre 0 y 2·w / 2·h (anchura y altura de la imagen).
Los valores obtenidos se transforman en desplazamientos relativos (tx - w, ty - h) para permitir movimiento en ambas direcciones.
La transformación se aplica mediante la matriz afín:

$$T = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \end{bmatrix}$$

y la función cv.warpAffine().

Rotaciones

La rotación se implementó con cv.getRotationMatrix2D() y cv.warpAffine().
El centro de giro se define de forma dinámica mediante las barras Centro X y Centro Y, y el ángulo de rotación con la barra Ángulo.
Esto permite rotar la imagen alrededor de cualquier punto, no solo del centro geométrico.

Escalados

Se incorporaron transformaciones de escalado uniforme y no uniforme.
Las barras Escalado X y Escalado Y permiten modificar los factores de escala horizontal y vertical (en porcentaje), y la barra Uniforme activa el modo de escalado igual en ambos ejes.
El escalado se aplica con la matriz:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \end{bmatrix}$$

y cv.warpAffine().

Combinación de transformaciones

Las tres transformaciones se aplican secuencialmente: primero rotación, luego escalado y finalmente traslación, lo que produce una transformación compuesta y fluida en tiempo real.

Gestión de interfaz y reinicio

Se añadió una función reset_trackbars() que devuelve todas las barras deslizantes a sus valores iniciales al pulsar la tecla R, permitiendo volver al estado original sin reiniciar el programa.
El cierre de la aplicación se realiza presionando ESC, garantizando la finalización correcta con:

cv.destroyAllWindows()
cv.waitKey(1)

1b. Dada una imagen trazar una ventana de proyección y proyectar la imagen

En este ejercicio se desarrolló una aplicación con el objetivo de realizar una transformación proyectiva (homografía) entre dos imágenes. La aplicación permite que el usuario seleccione manualmente los cuatro puntos de correspondencia entre la imagen original (fuente) y una imagen destino (ventana de proyección), sobre la cual se representa la imagen transformada.

Funcionamiento general

La aplicación carga inicialmente una imagen y genera una segunda ventana en blanco del mismo tamaño que actuará como superficie de proyección.
Mediante la función cv.setMouseCallback(), se definen dos callbacks del ratón: uno para la ventana Original y otro para la ventana Destino.

Cada vez que el usuario hace clic con el botón izquierdo:

- En la imagen original, se marcan puntos **rojos** numerados (1–4).
- En la imagen destino, se marcan puntos **azules** también numerados.

Cuando ambas listas (pts_src y pts_dst) contienen cuatro puntos, se calcula la matriz de homografía con:

- M = cv.getPerspectiveTransform(np.float32(pts_src), np.float32(pts_dst))

y se aplica la transformación con:

- proyectada = cv.warpPerspective(img, M, (img2.shape[1], img2.shape[0]))

Transformación proyectiva

La homografía es una transformación geométrica representada por una matriz 3×3 que relaciona dos planos de proyección, permitiendo reproducir cambios de perspectiva como si se rotara la cámara o se modificara el punto de vista.

A diferencia de la transformación afín, la proyectiva no conserva el paralelismo de las líneas y puede representar efectos de profundidad o inclinación.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Visualización y control

- Se muestran simultáneamente tres ventanas: Original, Destino y Proyección, permitiendo observar la correspondencia entre los puntos y el resultado de la transformación.
- El programa incluye un control de reinicio (‘r’) que limpia las listas de puntos y restaura las imágenes a su estado inicial.
- La tecla ESC cierra todas las ventanas de manera segura mediante cv.destroyAllWindows() y cv.waitKey(1).

1c. Desarrollar una aplicación que lleve a cabo distorsiones de la lente.

En este ejercicio se implementó una aplicación para simular los efectos de distorsión óptica de una lente.
El objetivo fue representar, de forma visual e interactiva, cómo varían las deformaciones de una imagen en función de los coeficientes de distorsión radial, controlados mediante una interfaz gráfica con trackbars.

Funcionamiento general

La aplicación carga una imagen original y crea una ventana llamada Distorsión, en la que se representan los resultados en tiempo real.
Mediante dos barras deslizantes denominadas K1 y K2, el usuario puede modificar los coeficientes de distorsión, cuyos valores se mapean del rango de las trackbars (0–1000) a un rango real de –0.005 a +0.005.
Estos valores se aplican a un modelo de cámara simplificado, definido por una matriz intrínseca (que contiene los parámetros focales y el punto principal) y un vector de coeficientes de distorsión.

Modelo matemático

La distorsión radial se describe mediante las siguientes expresiones:

$$x_{\text{corr}} = x(1 + k_1 r^2 + k_2 r^4), \qquad y_{\text{corr}} = y(1 + k_1 r^2 + k_2 r^4)$$

donde r es la distancia del punto al centro óptico de la imagen.

Los parámetros k1 y k2 controlan el tipo y magnitud de la distorsión:

- k>0k>0: produce un efecto de almohadón (pincushion), donde los bordes se contraen.
- k<0k<0: genera un efecto de barril (barrel), con bordes que se expanden hacia fuera.

En el código, los valores se aplican mediante la función cv2.undistort(), que utiliza la matriz de cámara (cam) y el vector de distorsión (distCoeff) para generar la imagen corregida o distorsionada, dependiendo del signo de los coeficientes.

Interfaz y control

- La ventana Distorsión muestra la imagen transformada y los valores actuales de k1 y k2, actualizados dinámicamente mediante cv2.putText().
- Se implementó una función reset_trackbars() que restablece los parámetros a su estado inicial (valores neutros), activada con la tecla R.
- La tecla ESC permite cerrar el programa de forma controlada con cv2.destroyAllWindows() y cv2.waitKey(1).

(Optativo – 5 puntos)

– Marcar el punto de giro con el ratón

Se creó una funcionalidad que permite definir interactivamente el centro de rotación mediante clic sobre la imagen.

Cada vez que el usuario hace clic, las coordenadas del punto se actualizan y la rotación se recalcula en tiempo real usando cv.getRotationMatrix2D((cx, cy), ang, 1.0) y cv.warpAffine().

El punto de giro se representa con círculos concéntricos y se muestra el ángulo actual junto a las coordenadas del centro.

La tecla R reinicia el punto al centro y ESC finaliza la ejecución.

– Trasladar la imagen arrastrándola con el ratón y visualizarlo en tiempo real

Se implementó una función de arrastre con el ratón (drag and drop) que permite trasladar la imagen de forma continua mientras se mantiene pulsado el botón izquierdo. Mediante cv.setMouseCallback() se detectan los eventos de clic, movimiento y liberación, actualizando las variables de desplazamiento (tx, ty) y aplicando la transformación afín:

$$T = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \end{bmatrix}$$

con cv.warpAffine().
El movimiento se refleja en tiempo real y puede reiniciarse con la tecla R, mientras que ESC finaliza la ejecución.

– Hacer la parte obligatoria sobre vídeo en lugar de sobre imagen

Se amplió la práctica implementando los apartados 1a, 1b y 1c sobre un vídeo en lugar de imágenes estáticas.

Mediante cv.VideoCapture(), se procesan los fotogramas en tiempo real, aplicando las transformaciones de traslación, rotación y escalado, la proyección y la distorsión de lente de forma dinámica.

Cada fotograma se transforma con las mismas funciones y matrices empleadas en los ejercicios previos, permitiendo visualizar los efectos geométricos sobre secuencias en movimiento.

– Dada una imagen seleccionar tres puntos de la imagen original y tres puntos en una imagen destino y realizar la transformación afín

En este ejercicio se implementó una aplicación que permite calcular y aplicar una transformación afín a partir de tres puntos seleccionados manualmente en dos imágenes distintas.
A diferencia de la homografía del apartado 1b (que requiere cuatro puntos y permite

deformaciones de perspectiva), la transformación afín preserva el paralelismo de las líneas y las proporciones relativas entre los puntos de la imagen, permitiendo rotar, trasladar, escalar o inclinar una figura sin introducir efectos de profundidad.

Funcionamiento general

La aplicación muestra dos ventanas (Original y Destino) y permite al usuario marcar con el ratón tres puntos en cada una.
Los puntos seleccionados en la primera imagen definen las coordenadas de referencia (pts_src), mientras que los puntos de la segunda (pts_dst) definen las posiciones deseadas.

Una vez que ambos conjuntos contienen tres puntos, se calcula la matriz de transformación afín con:

```
M = cv.getAffineTransform(np.float32(pts_src), np.float32(pts_dst))
```

y se aplica la transformación mediante:

```
cv.warpAffine(img, M, (img2.shape[1], img2.shape[0]))
```

Modelo matemático

La transformación afín se expresa como:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

donde los parámetros aij determinan la rotación, el escalado y la cizalla, mientras que tx y ty representan la traslación.
Esta transformación conserva la **linealidad y el paralelismo**, pero no necesariamente las distancias o los ángulos.

– Calcular la imagen especular a partir de una imagen

Se creó una función reflejar_imagen() que genera la imagen especular de una entrada original.

– Tratar una recta que será el eje de reflexión y “reflejar” la imagen

Se desarrolló una aplicación que permite definir una recta arbitraria como eje de reflexión y generar el reflejo de la imagen respecto a dicha línea.
El usuario selecciona dos puntos sobre la imagen mediante clics del ratón, que determinan la dirección y posición del eje.
Una vez definidos ambos puntos, se calcula la matriz de reflexión en función del ángulo de la recta, utilizando la relación:

$$M = \begin{bmatrix} \cos(2\theta) & \sin(2\theta) \\ \sin(2\theta) & -\cos(2\theta) \end{bmatrix}$$

donde θ es el ángulo que forma la recta con el eje X.
La matriz se combina con una traslación adecuada para mantener la posición original de la línea y se aplica mediante cv.warpAffine().

Durante la ejecución, la línea de reflexión se dibuja dinámicamente sobre la imagen original y también se representa sobre la imagen reflejada para facilitar la comparación visual.
El programa permite reiniciar la selección de puntos con la tecla R y salir con ESC.

– Otras aportaciones

Como aportación adicional, se desarrolló una herramienta de zoom interactivo que permite ampliar o reducir la imagen de forma fluida, manteniendo la calidad visual y el control total sobre el centro de ampliación.
El sistema utiliza eventos de ratón (cv.EVENT_MOUSEWHEEL) para detectar el movimiento del trackpad o la rueda del ratón, modificando el factor de escala (scale) y aplicando la transformación con:

```
M = cv.getRotationMatrix2D((center_x, center_y), 0, scale)

img_zoom = cv.warpAffine(img, M, (w, h))
```

De esta forma, el usuario puede acercar o alejar el área de interés en tiempo real, centrando el zoom en la posición actual del cursor.
Además, el programa incluye controles alternativos con teclado (‘+’, ‘–’, ‘R’ y ESC) y una interfaz enriquecida con superposiciones de texto que indican el nivel de zoom, el centro actual y las instrucciones de uso.

Conclusiones

La práctica cumple todos los apartados obligatorios y desarrolla la totalidad de los optativos propuestos, además de incluir una aportación propia de zoom.
El sistema resultante constituye una herramienta completa e interactiva para la exploración de transformaciones geométricas sobre imágenes y vídeo, destacando por su modularidad, estabilidad y claridad visual en tiempo real.