SISTEMAS SENSORIAIS

AULA 2 - *Operações Geométricas*

Objetivo:

Estudo, implementação e teste de operações geométricas (translação, rotação e escalamento) sobre imagens digitais.

Procedimento:

Notas para a resolução das questões:
- para aceder ao valor dos pixéis utilize o método de endereçamento absoluto descrito no final deste enunciado.
- a dimensão da imagem deverá manter-se. Pixeis que tenham coordenadas fora dos limites da imagem são perdidos.

1 – Implemente uma função que efetue a translação da imagem de um deslocamento (D_x, D_y) introduzido pelo utilizador. Este desvio poderá ser positivo ou negativo caso se pretenda um deslocamento no sentido inverso.

Translação:
$$x_{destino} = x_{origem} + D_x$$
$$y_{destino} = y_{origem} + D_y$$

2 – Implemente uma função que efetue a rotação <u>inversa</u> da imagem de um ângulo (θ), definido em graus, introduzido pelo utilizador. Este ângulo poderá ser positivo ou negativo, caso se pretenda uma rotação no sentido inverso.

Matriz de Rotação direta:
$$\begin{bmatrix} X_{destino} \\ Y_{destino} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_{origem} \\ Y_{origem} \end{bmatrix}$$

Nota: em anexo encontra-se a adaptação da matriz de rotação direta para a matriz de rotação inversa.

3 – Implemente uma função que efectue o escalamento (zoom) da imagem de um factor de escala (S) introduzido pelo utilizador. Este factor de escala deverá ser >1 para aumentar a imagem e <1 para reduzir a dimensão da imagem.

Para melhorar a eficácia desta operação escolha com o rato o ponto de referência (x_0,y_0) onde aplicar a operação de escalamento.

Matriz de Escalamento direto:
$$\begin{bmatrix} X_{destino} \\ Y_{destino} \end{bmatrix} = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} X_{origem} \\ Y_{origem} \end{bmatrix}$$

Matriz de rotação inversa

Da teoria das Matrizes sabemos que:

$$A = M.B \Leftrightarrow A.M^T = B \tag{1}$$

Sendo,

$$\begin{bmatrix} X_{Destino} \\ Y_{Destino} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} X_{origem} \\ Y_{origem} \end{bmatrix} \tag{2}$$

Então:

$$\begin{bmatrix} X_{origem} \\ Y_{origem} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}^T . \begin{bmatrix} X_{Destino} \\ Y_{Destino} \end{bmatrix} \Leftrightarrow \begin{bmatrix} X_{origem} \\ Y_{origem} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} . \begin{bmatrix} X_{Destino} \\ Y_{Destino} \end{bmatrix} \tag{3}$$

Então vem,

$$\begin{cases} X_{origem} = X_{Destino}.\cos\theta - Y_{Destino}\sin\theta \\ Y_{origem} = X_{Destino}.\sin\theta + Y_{Destino}.\cos\theta \end{cases} \tag{4}$$

Como a rotação deverá ser centrada na imagem é necessário alterar o referencial (x,y) para o centro da imagem (**W**idth/2, **H**eight/2), usando:

$$\begin{cases} X_{\text{centrado}} = X - \dfrac{W}{2} \\ Y_{\text{centrado}} = \dfrac{H}{2} - Y \end{cases} \tag{5}$$

Substituindo em (4) vem:

$$\begin{cases} X_{origem} - \dfrac{W}{2} = \left(X_{destino} - \dfrac{W}{2} \right).\cos\theta - \left(\dfrac{H}{2} - Y_{destino} \right).\sin\theta \\ \dfrac{H}{2} - Y_{origem} = \left(X_{destino} - \dfrac{W}{2} \right).\sin\theta + \left(\dfrac{H}{2} - Y_{destino} \right).\cos\theta \end{cases} \tag{6}$$

Finalmente,

$$\begin{cases} X_{origem} = \left(X_{destino} - \dfrac{W}{2} \right).\cos\theta - \left(\dfrac{H}{2} - Y_{destino} \right).\sin\theta + \dfrac{W}{2} \\ Y_{origem} = \dfrac{H}{2} - \left(X_{destino} - \dfrac{W}{2} \right).\sin\theta - \left(\dfrac{H}{2} - Y_{destino} \right).\cos\theta \end{cases} \tag{7}$$

Endereçamento Relativo vs Absoluto

- Endereçamento Relativo:

  A imagem vai ser percorrida pixel a pixel avançando o apontador inicial (dataptr).

```
// obter apontador do inicio da imagem
MIplImage m = img.MIplImage;
byte* dataPtr = (byte*)m.imageData.ToPointer();

for (int y = 0; y < h; y++)
{
        for (int x = 0; x < w; x++)
        {


                // avança apontador pixel a pixel
                dataPtr += nC;
        }

        //no fim da linha avança alinhamento
        dataPtr += padding;
}
```

- Endereçamento Absoluto:

  É possível aceder a todos os pixels (x,y) da imagem calculando o endereço do pixel pretendido.

```
// obter apontador do inicio da imagem
MIplImage m = img.MIplImage;
byte* dataPtr = (byte*)m.imageData.ToPointer();

for (int y = 0; y < h; y++)
{
        for (int x = 0; x < w; x++)
        {
                // calcula endereço do pixel no ponto (x,y)
                blue = (byte)(dataPtr + y * widthstep + x * nC)[0];

        }
}
```