Nome: João P S Oliveira / 2893577 - Eng da Computação

```
import numpy as np
import matplotlib.pyplot as plt
%pip install scikit-image
from skimage import io, transform, img_as_float
```

1) Qual é a função das matrizes de transformação afim (como identidade, escala etc.) e como cada elemento da matriz afeta a imagem transformada? As matrizes de transformação afim são usadas para aplicar operações geométricas em imagens, como:

- Identidade: Mantém a imagem inalterada.
- Escala: Aumenta ou diminui o tamanho da imagem ao multiplicar as coordenadas por fatores específicos.
- Rotação: Gira a imagem em torno de um ponto de origem usando ângulos em radianos.
- Translação: Move a imagem de um ponto para outro sem alterar sua forma.

Cada elemento da matriz afim define uma parte da transformação:

- Os elementos na diagonal controlam o escalamento.
- Os elementos fora da diagonal (para rotação e cisalhamento) ajustam a forma.
- A última coluna geralmente define a translação (deslocamento).

2) Como a escolha das diferentes variáveis de transformação, como escalamento, rotação e translação, influencia o resultado final? Escalamento: Modifica o tamanho da imagem. Fatores maiores ampliam a imagem, enquanto menores reduzem. Rotação: Gira a imagem em torno de um ponto de referência. O ângulo determina o quanto a imagem é rotacionada. Translação: Move a imagem para uma nova posição. Valores maiores deslocam a imagem mais longe do ponto original. A combinação dessas variáveis pode deformar, mover ou ajustar o tamanho e orientação da imagem de forma significativa.

3) Principais Conclusões do Código O código aplica diferentes transformações afins a uma imagem usando funções predefinidas. Cada transformação (identidade, escala, rotação, translação, cisalhamento) altera a imagem de forma previsível e visualmente distinta. O uso de matrizes permite manipular facilmente as propriedades geométricas da imagem, ajustando seu tamanho, forma e posição.

```
def apply_affine_transformation(image, T, title):
    transformed_image = transform.warp(image, T.inverse,
output_shape=image.shape)

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.title("Imagem Original")
    plt.imshow(image, cmap='gray')
    plt.axis("off")

    plt.subplot(1, 2, 2)
```

```
    plt.title(f"Transformação: {title}")
    plt.imshow(transformed_image, cmap='gray')
    plt.axis('off')

    plt.show()

# Carregamento da imagem
image_path = "CaoCinza.jpg"
image = img_as_float(io.imread(image_path, as_gray=True))

# Definição de várias transformações afins
T_identidade = transform.AffineTransform(matrix=np.array([[1, 0, 0],
[0, 1, 0], [0, 0, 1]]))
T_escala = transform.AffineTransform(scale=(1.5, 0.5))
T_rotacao = transform.AffineTransform(rotation=np.deg2rad(45))
T_translacao = transform.AffineTransform(translation=(50, 30))
T_cisalhamento_hor = transform.AffineTransform(shear=np.deg2rad(30))
T_cisalhamento_ver = transform.AffineTransform(matrix=np.array([[1,
np.tan(np.deg2rad(30)), 0], [0, 1, 0], [0, 0, 1]]))

# Aplicação das transformações
apply_affine_transformation(image, T_identidade, "Identidade")
apply_affine_transformation(image, T_escala, "Escala")
apply_affine_transformation(image, T_rotacao, "Rotação")
apply_affine_transformation(image, T_translacao, "Translação")
apply_affine_transformation(image, T_cisalhamento_hor, "Cisalhamento
Horizontal")
apply_affine_transformation(image, T_cisalhamento_ver, "Cisalhamento
Vertical")
```

Imagem Original

Transformação: Escala

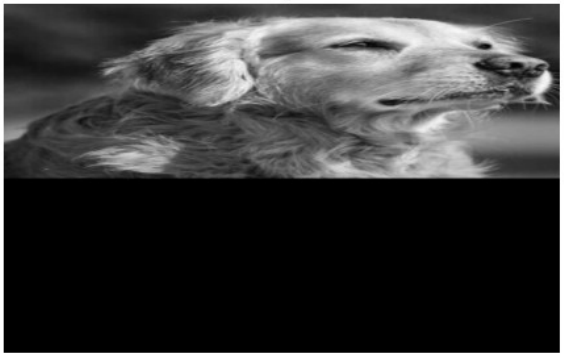

Imagem Original

Transformação: Rotação

Imagem Original

Transformação: Translação

Imagem Original

Transformação: Cisalhamento Horizontal

Imagem Original

Transformação: Cisalhamento Vertical