PDIE8 - Processamento Digital de Imagem

Repositorio dos Códigos feitos em aula da Disciplina

Feito por: Giovana Perazzolo Menato e Vinicius de Souza Santos

Lecionado por: Murilo Varges

Exercicio 3

[TRANSFORMAÇÕES GEOMÉTRICAS]:

Para cada filtro implementar utilizando apenas numpy, utilizando pillow, utilizando opencv e utilizando scipy.

- Escala: Redução em 1.5x e aumentar em 2.5x;
- Rotação em 45º, 90º e 100º;
- Translação utilizar os parâmetros que quiser nas coordenadas x e y;
- Translação em 35 pixel no eixo X, 45 eixo Y;

Utilizando a Biblioteca Numpy

Escala: Redução em 1.5x e aumentar em 2.5x

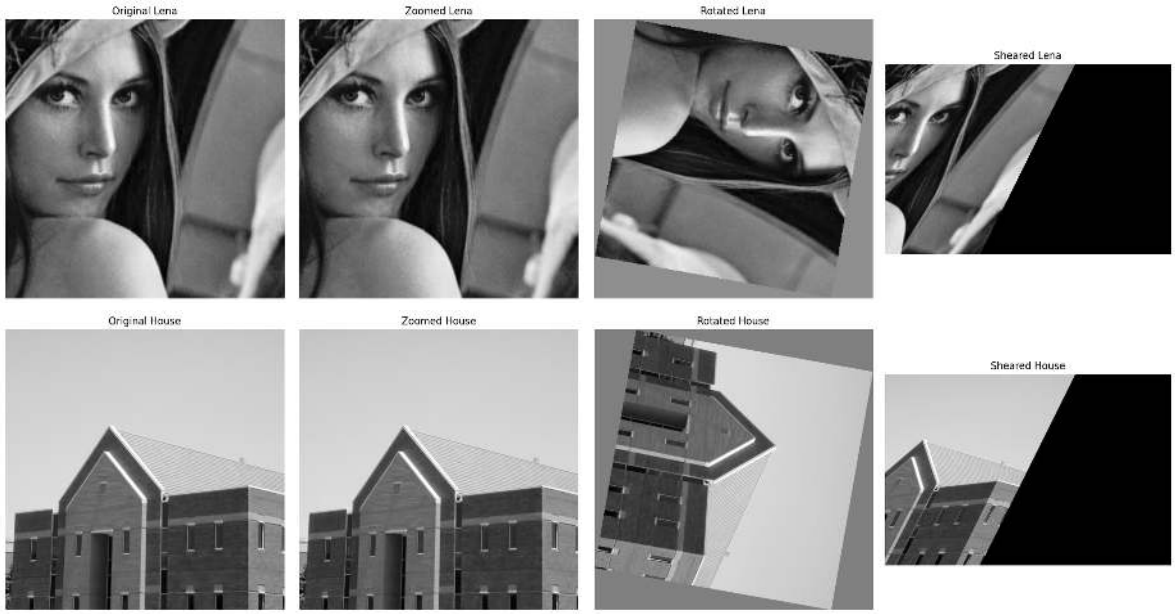
Rotação em 45º, 90º e 100º

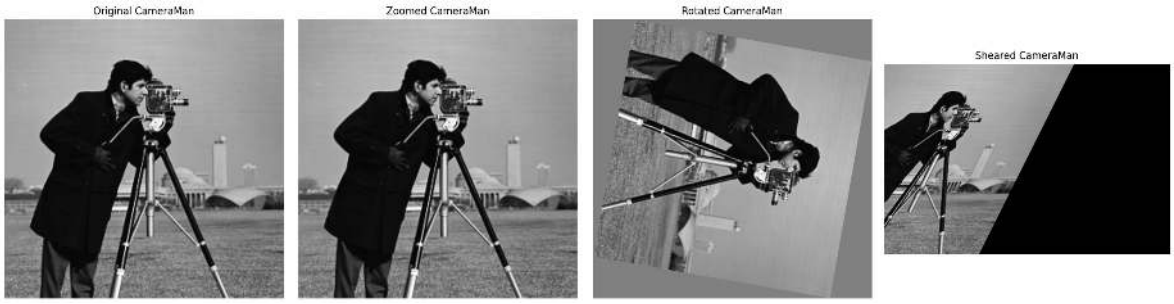
Importando as Biliotecas

In [ ]:
```
import numpy as np
from numpy import asarray
from PIL import Image
from scipy import ndimage
```

In [ ]:
```
def apply_transforms(image_path, title):
    # Open image
    image_in = Image.open(image_path)

    # Convert image to numpy array
    image_np = np.array(image_in)

    # Zoom or Shrink image
    image_np_zoom = ndimage.zoom(image_np, (2.5, 2.5))

    # Rotation image 45º
    image_np_rotate  = ndimage.rotate(image_np, -100, cval=128)

    # Shear image
```

```
    height, width = image_np.shape
    transform = [[1, 0, 0],
                 [0.5, 1, 0],
                 [0, 0, 1]]
    image_np_shear = ndimage.affine_transform(image_np,
                                              transform,
                                              offset=(0, -height//2, 0),
                                              output_shape=(height, width+height//2))

    # Display images using matplotlib
    fig, axs = plt.subplots(1, 4, figsize=(20, 5))
    axs[0].imshow(image_np, cmap='gray')
    axs[0].set_title(f'Original {title}')
    axs[1].imshow(image_np_zoom, cmap='gray')
    axs[1].set_title(f'Zoomed {title}')
    axs[2].imshow(image_np_rotate, cmap='gray')
    axs[2].set_title(f'Rotated {title}')
    axs[3].imshow(image_np_shear, cmap='gray')
    axs[3].set_title(f'Sheared {title}')

    for ax in axs:
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Paths to the images
image_pathLena = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imag
image_pathHouse = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Ima
image_pathCamera = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Im

# Apply transformations to each image
apply_transforms(image_pathLena, 'Lena')
apply_transforms(image_pathHouse, 'House')
apply_transforms(image_pathCamera, 'CameraMan')
```





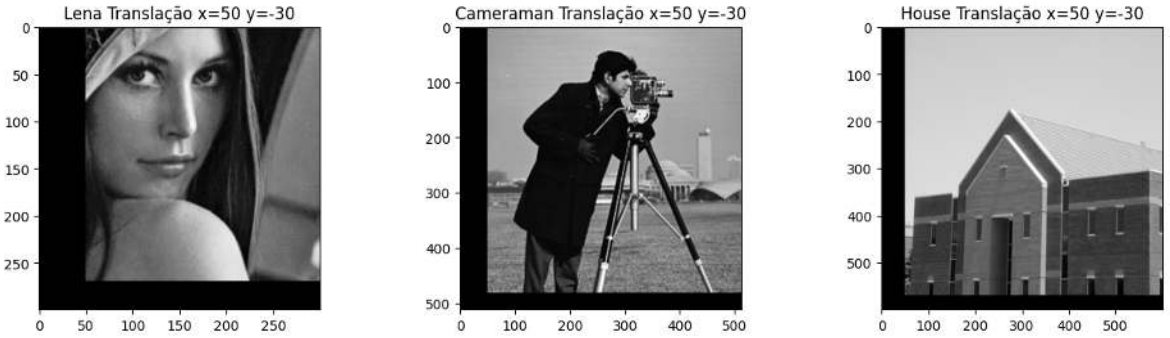
Translação utilizar os parâmetros que quiser nas coordenadas x e y

Importando Bibliotecas

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
from PIL import Image
```

In [ ]:
```
# Load the images
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

lena = np.array(Image.open(lena_path).convert('L'))  # Convert to grayscale
cam = np.array(Image.open(cam_path).convert('L'))
house = np.array(Image.open(house_path).convert('L'))

# Copy the images
translLena = lena.copy()
translCam = cam.copy()
translHouse = house.copy()

# Define the translation (shift) vector
shift_x = 50
shift_y = -30
shift_vector = [shift_y, shift_x]

# Apply the translation to each image
translated_image_lena = ndimage.shift(translLena, shift_vector, mode='constant',
translated_image_cam = ndimage.shift(translCam , shift_vector, mode='constant',
translated_image_house = ndimage.shift(translHouse, shift_vector, mode='constant

# Display the translated images using matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=50 y=-30")
plt2.title.set_text("Cameraman Translação x=50 y=-30")
plt3.title.set_text("House Translação x=50 y=-30")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")
```

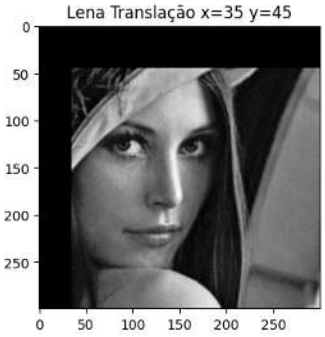
```
plt.show()
```

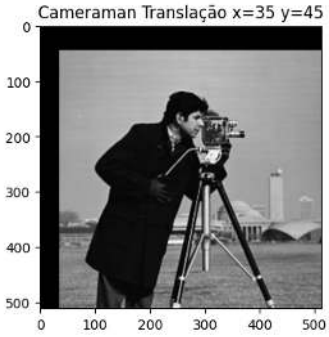


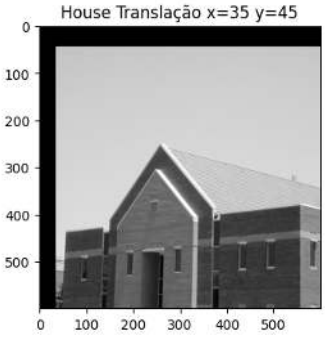
Translação em 35 pixel no eixo X, 45 eixo Y

In [ ]:

```
# Load the images
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

lena = np.array(Image.open(lena_path).convert('L'))  # Convert to grayscale
cam = np.array(Image.open(cam_path).convert('L'))
house = np.array(Image.open(house_path).convert('L'))

# Copy the images
translLena = lena.copy()
translCam = cam.copy()
translHouse = house.copy()

# Define the translation (shift) vector
shift_x = 35
shift_y = 45
shift_vector = [shift_y, shift_x]

# Apply the translation to each image
translated_image_lena = ndimage.shift(translLena, shift_vector, mode='constant',
translated_image_cam = ndimage.shift(translCam , shift_vector, mode='constant',
translated_image_house = ndimage.shift(translHouse, shift_vector, mode='constant

# Display the translated images using matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=35 y=45")
plt2.title.set_text("Cameraman Translação x=35 y=45")
plt3.title.set_text("House Translação x=35 y=45")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```







Utilizando a Biblioteca Pilow

Escala: Redução em 1.5x e aumentar em 2.5x

Rotação em 45º, 90º e 100º

Importando Bibliotecas

In [ ]:
```
from PIL import Image, ImageOps, ImageFilter
import matplotlib.pyplot as plt
```

In [ ]:
```
def apply_transforms(image_path, title):
    # Open image
    image_in = Image.open(image_path)

    # Zoom or Shrink image
    size = tuple(int(dim * 2.5) for dim in image_in.size)
    image_zoom = image_in.resize(size, Image.ANTIALIAS)

    # Rotation image 45º
    image_rotate = image_in.rotate(100, resample=Image.BICUBIC, fillcolor=128)

    # Shear image
    width, height = image_in.size
    m = -0.5  # Shear factor
    shear_matrix = (1, m, -m*height/2, 0, 1, 0)
    image_shear = image_in.transform((width + int(m*height), height), Image.AFFI

    # Display images using matplotlib
    fig, axs = plt.subplots(1, 4, figsize=(20, 5))
    axs[0].imshow(image_in, cmap='gray')
    axs[0].set_title(f'Original {title}')
    axs[1].imshow(image_zoom, cmap='gray')
    axs[1].set_title(f'Zoomed {title}')
    axs[2].imshow(image_rotate, cmap='gray')
    axs[2].set_title(f'Rotated {title}')
    axs[3].imshow(image_shear, cmap='gray')
    axs[3].set_title(f'Sheared {title}')

    for ax in axs:
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Paths to the images
```

```
image_pathLena = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imag
image_pathHouse = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Ima
image_pathCamera = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Im

# Apply transformations to each image
apply_transforms(image_pathLena, 'Lena')
apply_transforms(image_pathHouse, 'House')
apply_transforms(image_pathCamera, 'CameraMan')
```

```
C:\Users\vinny\AppData\Local\Temp\ipykernel_11320\1383239210.py:7: DeprecationWar
ning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use
LANCZOS or Resampling.LANCZOS instead.
  image_zoom = image_in.resize(size, Image.ANTIALIAS)
```

Translação utilizar os parâmetros que quiser nas coordenadas x e y

In [ ]:
```
# Load the images
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

lena = Image.open(lena_path).convert('L')  # Convert to grayscale
cam = Image.open(cam_path).convert('L')
house = Image.open(house_path).convert('L')

# Define the translation (shift) vector
shift_x = 50
shift_y = -30

# Apply the translation to each image
```

```
translated_image_lena = lena.transform(lena.size, Image.AFFINE, (1, 0, shift_x,
translated_image_cam = cam.transform(cam.size, Image.AFFINE, (1, 0, shift_x, 0,
translated_image_house = house.transform(house.size, Image.AFFINE, (1, 0, shift_

# Display the translated images using matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=50 y=-30")
plt2.title.set_text("Cameraman Translação x=50 y=-30")
plt3.title.set_text("House Translação x=50 y=-30")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```

Translação em 35 pixel no eixo X, 45 eixo Y

In [ ]:

```
# Load the images
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

lena = Image.open(lena_path).convert('L')  # Convert to grayscale
cam = Image.open(cam_path).convert('L')
house = Image.open(house_path).convert('L')

# Define the translation (shift) vector
shift_x = 35
shift_y = 45

# Apply the translation to each image
translated_image_lena = lena.transform(lena.size, Image.AFFINE, (1, 0, shift_x,
translated_image_cam = cam.transform(cam.size, Image.AFFINE, (1, 0, shift_x, 0,
translated_image_house = house.transform(house.size, Image.AFFINE, (1, 0, shift_

# Display the translated images using matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=35 y=45")
plt2.title.set_text("Cameraman Translação x=35 y=45")
plt3.title.set_text("House Translação x=35 y=45")
```

```
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```

Utilizando a Biblioteca OpenCV

Escala: Redução em 1.5x e aumentar em 2.5x

Rotação em 45°, 90° e 100°

Importando Bliblioteca

In [ ]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
def apply_transforms(image_path, title):
    # Open image
    image_in = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Zoom or Shrink image
    zoom_scale = 2.5
    image_zoom = cv2.resize(image_in, None, fx=zoom_scale, fy=zoom_scale, interp

    # Rotation image 45º
    rows, cols = image_in.shape
    M_rotate = cv2.getRotationMatrix2D((cols/2, rows/2), -100, 1)
    image_rotate = cv2.warpAffine(image_in, M_rotate, (cols, rows), borderValue=

    # Shear image
    M_shear = np.float32([[1, 0.5, -0.5*cols/2], [0, 1, 0]])
    image_shear = cv2.warpAffine(image_in, M_shear, (int(cols + 0.5*rows), rows)

    # Display images using matplotlib
    fig, axs = plt.subplots(1, 4, figsize=(20, 5))
    axs[0].imshow(image_in, cmap='gray')
    axs[0].set_title(f'Original {title}')
    axs[1].imshow(image_zoom, cmap='gray')
    axs[1].set_title(f'Zoomed {title}')
    axs[2].imshow(image_rotate, cmap='gray')
    axs[2].set_title(f'Rotated {title}')
```

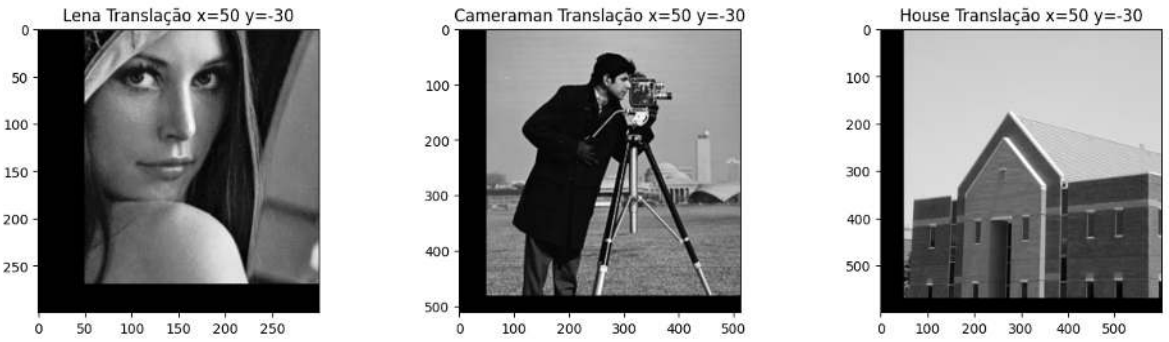
```
    axs[3].imshow(image_shear, cmap='gray')
    axs[3].set_title(f'Sheared {title}')

    for ax in axs:
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Paths to the images
image_pathLena = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imag
image_pathHouse = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Ima
image_pathCamera = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Im

# Apply transformations to each image
apply_transforms(image_pathLena, 'Lena')
apply_transforms(image_pathHouse, 'House')
apply_transforms(image_pathCamera, 'CameraMan')
```

Translação utilizar os parâmetros que quiser nas coordenadas x e y

In [ ]:

```
def apply_translation(image_path, shift_x, shift_y, title):
    # Carregar a imagem
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Definir a matriz de translação
    M = np.float32([[1, 0, shift_x], [0, 1, shift_y]])

    # Aplicar a translação
    translated_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0])
```

```
    return translated_image

# Caminhos para as imagens
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

# Definir o vetor de translação
shift_x = 50
shift_y = -30

# Aplicar a translação a cada imagem
translated_image_lena = apply_translation(lena_path, shift_x, shift_y, 'Lena')
translated_image_cam = apply_translation(cam_path, shift_x, shift_y, 'Cameraman'
translated_image_house = apply_translation(house_path, shift_x, shift_y, 'House'

# Exibir as imagens traduzidas usando matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=50 y=-30")
plt2.title.set_text("Cameraman Translação x=50 y=-30")
plt3.title.set_text("House Translação x=50 y=-30")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```

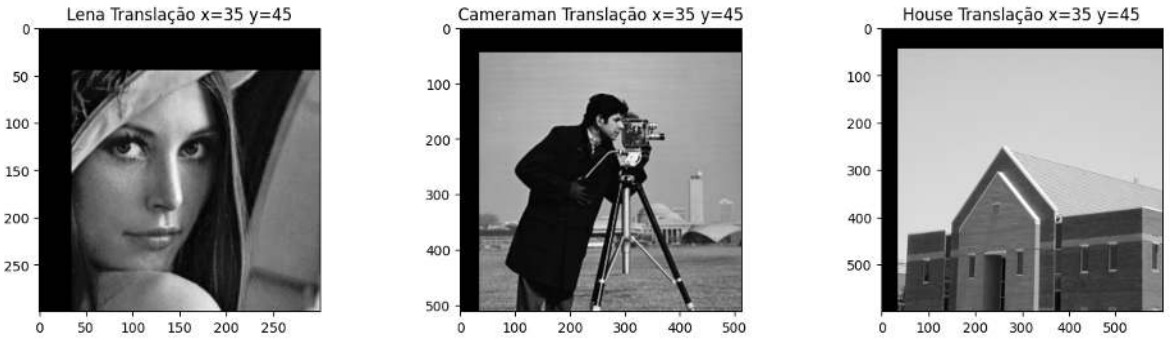


Translação em 35 pixel no eixo X, 45 eixo Y

In [ ]:

```
def apply_translation(image_path, shift_x, shift_y):
    # Carregar a imagem
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Definir a matriz de translação
    M = np.float32([[1, 0, shift_x], [0, 1, shift_y]])

    # Aplicar a translação
    translated_image = cv2.warpAffine(image, M, (image.shape[1], image.shape[0])

    return translated_image

# Caminhos para as imagens
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
```

```
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

# Definir o vetor de translação
shift_x = 35
shift_y = 45

# Aplicar a translação a cada imagem
translated_image_lena = apply_translation(lena_path, shift_x, shift_y)
translated_image_cam = apply_translation(cam_path, shift_x, shift_y)
translated_image_house = apply_translation(house_path, shift_x, shift_y)

# Exibir as imagens traduzidas usando matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=35 y=45")
plt2.title.set_text("Cameraman Translação x=35 y=45")
plt3.title.set_text("House Translação x=35 y=45")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```



Utilizando a Biblioteca Scipy

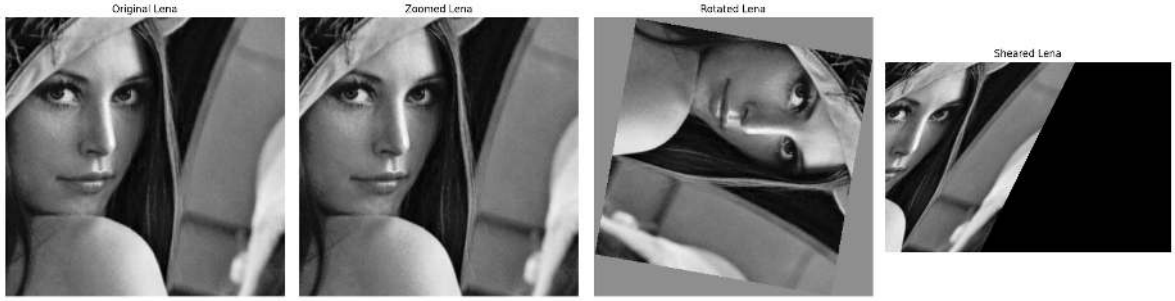
Importando Biblioteca

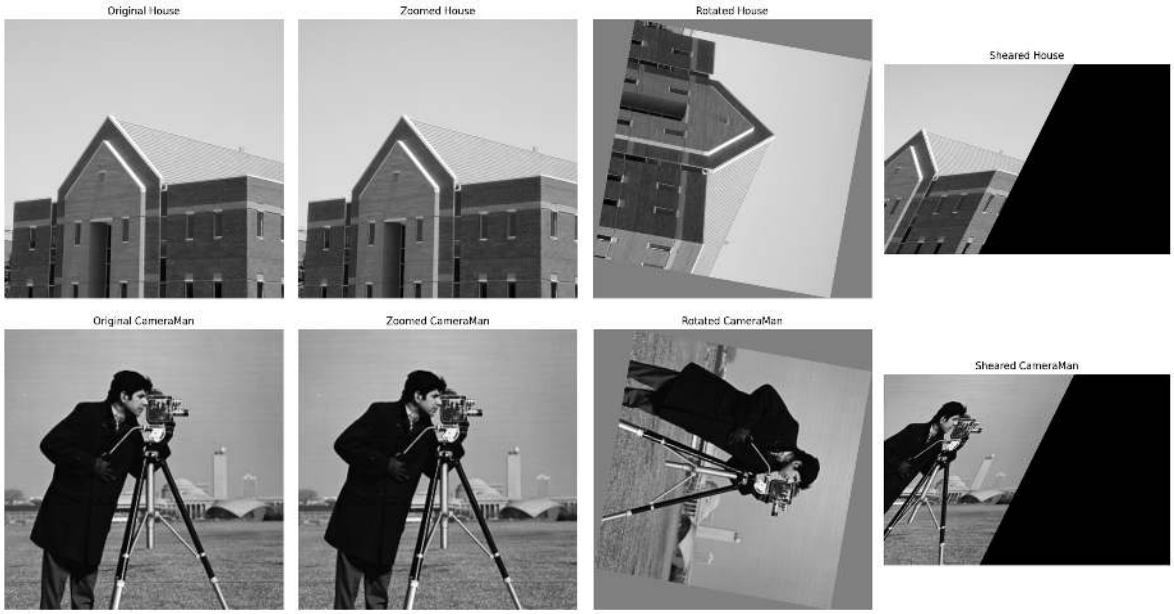
In [ ]:
```
import matplotlib.pyplot as plt
from scipy import ndimage
import imageio
```

In [ ]:
```
def apply_transforms(image_path, title):
    # Open image
    image_in = imageio.imread(image_path)

    # Convert to grayscale if the image is RGB
    if image_in.ndim == 3:
        image_in = imageio.imread(image_path, as_gray=True)

    # Zoom or Shrink image
    image_np_zoom = ndimage.zoom(image_in, 2.5)
```

```
    # Rotation image 45º
    image_np_rotate  = ndimage.rotate(image_in, -100, cval=128)

    # Shear image
    height, width = image_in.shape
    transform = [[1, 0, 0],
                 [0.5, 1, 0],
                 [0, 0, 1]]
    image_np_shear = ndimage.affine_transform(image_in,
                                              transform,
                                              offset=(0, -height//2),
                                              output_shape=(height, width+height//2))

    # Display images using matplotlib
    fig, axs = plt.subplots(1, 4, figsize=(20, 5))
    axs[0].imshow(image_in, cmap='gray')
    axs[0].set_title(f'Original {title}')
    axs[1].imshow(image_np_zoom, cmap='gray')
    axs[1].set_title(f'Zoomed {title}')
    axs[2].imshow(image_np_rotate, cmap='gray')
    axs[2].set_title(f'Rotated {title}')
    axs[3].imshow(image_np_shear, cmap='gray')
    axs[3].set_title(f'Sheared {title}')

    for ax in axs:
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Paths to the images
image_pathLena = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imag
image_pathHouse = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Ima
image_pathCamera = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Im

# Apply transformations to each image
apply_transforms(image_pathLena, 'Lena')
apply_transforms(image_pathHouse, 'House')
apply_transforms(image_pathCamera, 'CameraMan')
```

```
C:\Users\Vinicius\AppData\Local\Temp\ipykernel_8292\1969412905.py:3: DeprecationW
arning: Starting with ImageIO v3 the behavior of this function will switch to tha
t of iio.v3.imread. To keep the current behavior (and make this warning disappea
r) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image_in = imageio.imread(image_path)
```



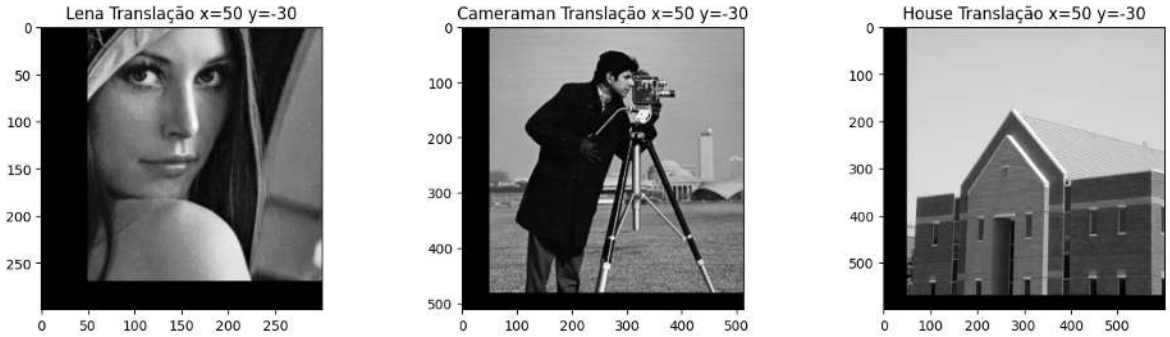


Translação utilizar os parâmetros que quiser nas coordenadas x e y

In [ ]:
```
def apply_translation(image_path, shift_x, shift_y, title):
    # Carregar a imagem
    image = imageio.imread(image_path)

    # Aplicar a translação
    translated_image = ndimage.shift(image, (shift_y, shift_x))
    return translated_image

# Definir o vetor de translação
shift_x = 50
shift_y = -30

# Caminhos para as imagens
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

# Aplicar a translação a cada imagem
translated_image_lena = apply_translation(lena_path, shift_x, shift_y, 'Lena')
translated_image_cam = apply_translation(cam_path, shift_x, shift_y, 'Cameraman'
translated_image_house = apply_translation(house_path, shift_x, shift_y, 'House'

# Exibir as imagens traduzidas usando matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=50 y=-30")
plt2.title.set_text("Cameraman Translação x=50 y=-30")
plt3.title.set_text("House Translação x=50 y=-30")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")
```

```
plt.show()
```

```
C:\Users\Vinicius\AppData\Local\Temp\ipykernel_8292\309799501.py:3: DeprecationWa
rning: Starting with ImageIO v3 the behavior of this function will switch to that
of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path)
```



Translação em 35 pixel no eixo X, 45 eixo Y

In [ ]:
```
def apply_translation(image_path, shift_x, shift_y):
    # Carregar a imagem
    image = imageio.imread(image_path)

    # Aplicar a translação
    translated_image = ndimage.shift(image, (shift_y, shift_x))
    return translated_image

# Definir o vetor de translação
shift_x = 35
shift_y = 45

# Caminhos para as imagens
lena_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Au
cam_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/Aul
house_path = '/Meu Drive/Faculdade/Aula/2023.2/Processamento Digital de Imagem/A

# Aplicar a translação a cada imagem
translated_image_lena = apply_translation(lena_path, shift_x, shift_y)
translated_image_cam = apply_translation(cam_path, shift_x, shift_y)
translated_image_house = apply_translation(house_path, shift_x, shift_y)

# Exibir as imagens traduzidas usando matplotlib
fig = plt.figure(figsize=(15, 5))
plt1 = plt.subplot(1, 3, 1)
plt2 = plt.subplot(1, 3, 2)
plt3 = plt.subplot(1, 3, 3)
plt1.title.set_text("Lena Translação x=35 y=45")
plt2.title.set_text("Cameraman Translação x=35 y=45")
plt3.title.set_text("House Translação x=35 y=45")
plt.subplots_adjust(wspace=0.5)

plt1.imshow(translated_image_lena, cmap="gray")
plt2.imshow(translated_image_cam, cmap="gray")
plt3.imshow(translated_image_house, cmap="gray")

plt.show()
```

```
C:\Users\Vinicius\AppData\Local\Temp\ipykernel_8292\231646287.py:3: DeprecationWa
rning: Starting with ImageIO v3 the behavior of this function will switch to that
of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(image_path)
```