# Question 19

Write a program to implement image rotation operation by amount of a) 25° b) 45° c) 60°

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import random
from skimage.transform import rotate
```

## Images to process

In [2]:
```python
path_inp = '../../images/dat/'   # path for input files
path_out_orig = 'originals/'      # path for output files: originals
path_out_conv = 'converted/'      # path for output files: converted

filenames = [
    'f256',
    'l256',
    'o256'
]

ext_inp = '.dat'     # file extention for input
ext_out = '.bmp'     # file extention for output
```

### Convert images to numpy array and store in a list of tuples as (filename, np.array)

In [3]:
```python
# Stores the list of dictionaries for the filename, original image, converted image/s
images = []

# Iterate for all filenames
for idx, filename in enumerate(filenames):
    # Store image pixels as uint8 2D array
    image = np.array(
        [i.strip().split() for i in open(path_inp + filename + ext_inp).readlines()],
        dtype='uint8'
    )

    # Add (filename, numpy array of image) into images list
    images.append({
        'filename': filename,
        'orig': image,
        'equalized': None
    })

    # Save original image as .dat file
    np.savetxt(
        path_out_orig + ext_inp[1:] + '/' + filename + ext_inp,
        image,
        fmt=' %d',
        newline=' \n'
    )
```

### Display input images

In [4]:
```python
# Matrix dimensions
cols = 3
rows = 1

# Create figure with rows × cols subplots
fig, axs = plt.subplots(rows, cols, dpi=80, sharex=True, sharey=True)
fig.set_size_inches(4 * cols, 4.5 * rows)

# Iterate for all images
for idx, image_dict in enumerate(images):
    filename = image_dict['filename']
    image = image_dict['orig']

    # Set subplot title as '"filename" (rows, cols)'
    axs[idx].set_title('"{}" {}'.format(
        filename + ext_inp,
        image.shape
    ))
    # Add subplot to figure plot buffer
    axs[idx].imshow(
        image,
        cmap='gray',
        vmin=0,
        vmax=255
    )

    # Save original image as .bmp file
    plt.imsave(
        path_out_orig + ext_out[1:] + '/' + filename + ext_out,
        image,
        cmap='gray',
```
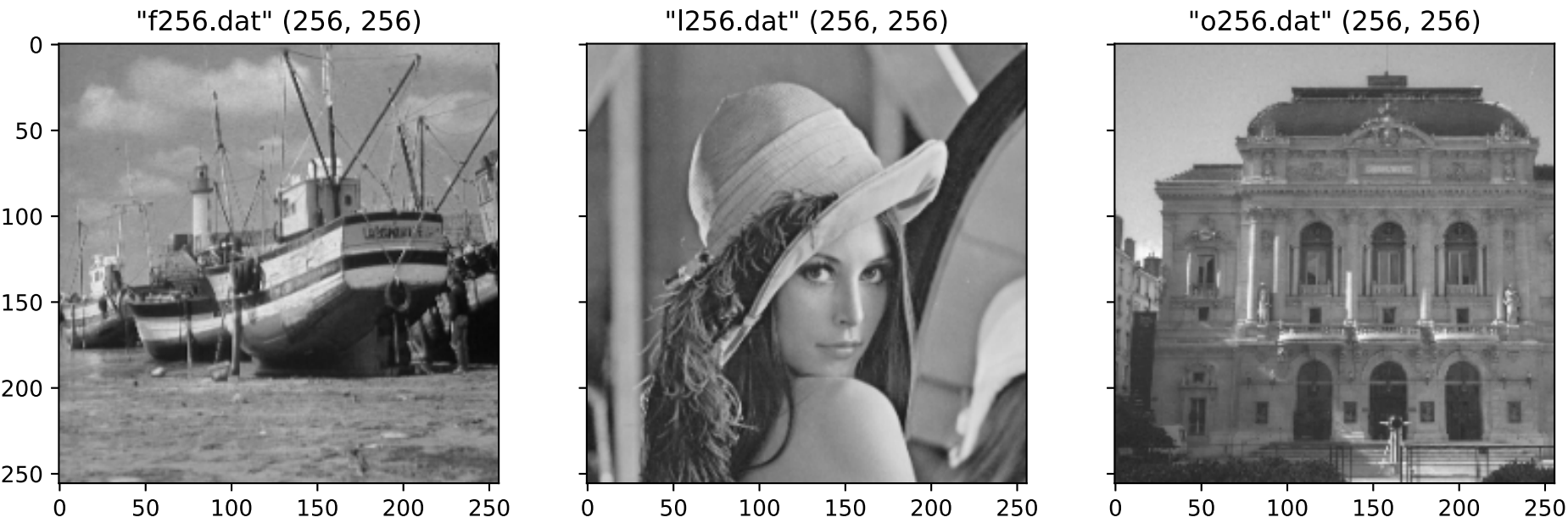
```
                vmin=0,
                vmax=255
        )

    # Hide x labels and tick labels for top plots and y ticks for right plots
    for ax in axs.flat:
        ax.label_outer()

    # Display the figure
    plt.show()
```



"f256.dat" (256, 256)    "l256.dat" (256, 256)    "o256.dat" (256, 256)

## Rotate Image

In [5]:
```python
def rotate_image(image, degrees: int):
    return rotate(image, degrees, resize=True) * 255
```

In [6]:
```python
rows, cols = len(images), 4

# Create figure with rows × cols subplots
fig, axs = plt.subplots(rows, cols, dpi=80)
fig.set_size_inches(4.5 * cols, 4.5 * rows)

# Iterate for all images
for idx, image_dict in enumerate(images):
    filename = image_dict['filename']

    angles = [25, 45, 60]
    orig = image_dict['orig']
    rotated = list(map(lambda x: rotate_image(image, x), angles))

    axs[idx, 0].set_title('"{}"'.format(filename))
    axs[idx, 0].imshow(orig, cmap='gray', vmin=0, vmax=255)

    for i, angle in enumerate(angles):
        axs[idx, i + 1].set_title(f'rotated {angle}°'.format(filename))
        axs[idx, i + 1].imshow(rotated[i], cmap='gray', vmin=0, vmax=255)

        # Save pixel values of original image's histogram as a 2D matrix in a .dat file
        np.savetxt(
            path_out_conv + ext_inp[1:] + '/' + filename + f'_rotated_{angle}' + ext_inp,
            rotated[i],
            fmt=' %d',
            newline=' \n'
        )

        # Save noisy image as .bmp file
        plt.imsave(
            path_out_conv + ext_out[1:] + '/' + filename + f'_rotated_{angle}' + ext_out,
            rotated[i],
            cmap='gray',
            vmin=0,
            vmax=255
        )

# Save and display the figure
plt.savefig('rotate_image.jpg')
plt.show()
```
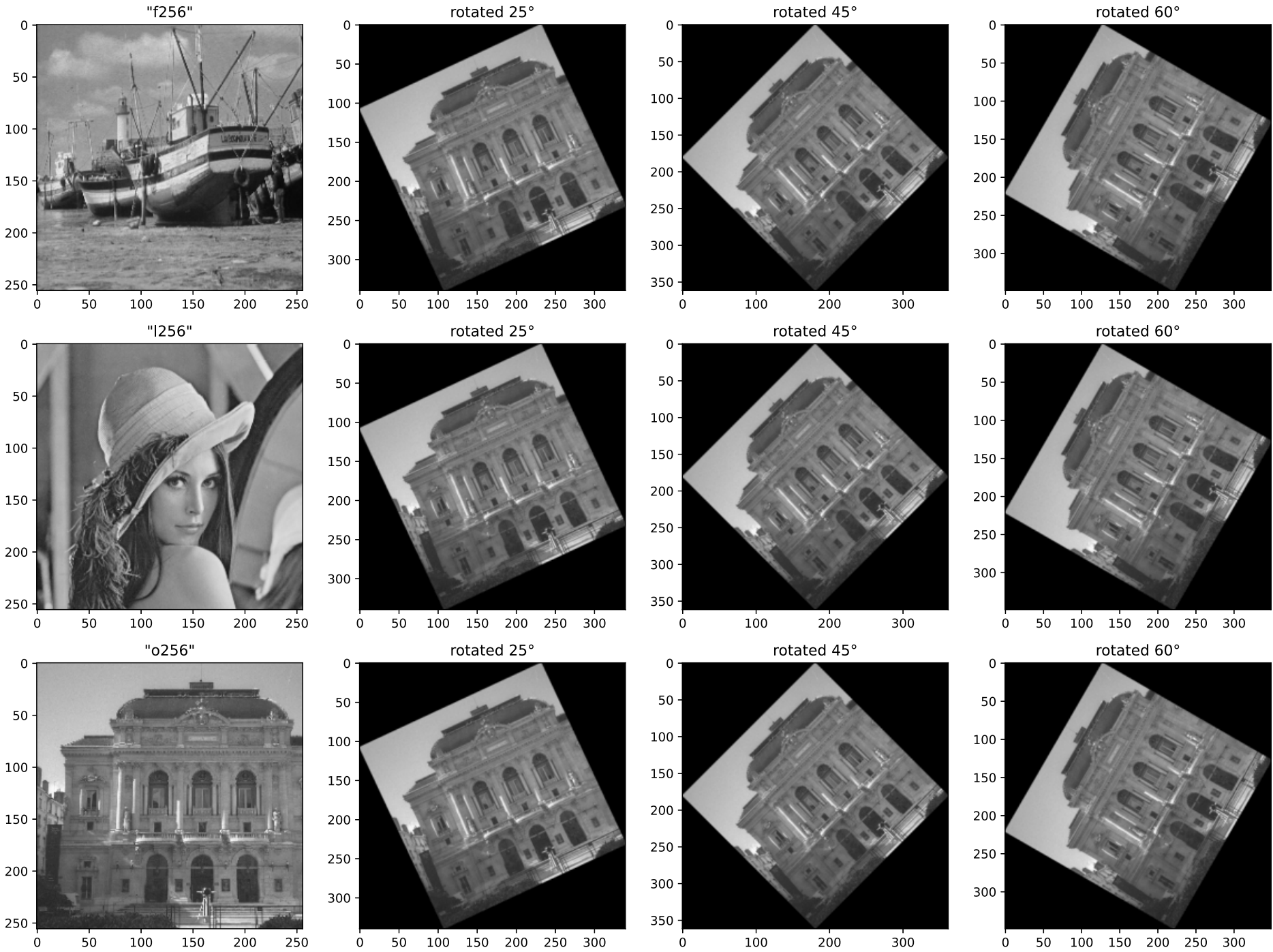
## Resource

**GitHub repository:** Image Processing and Pattern Recognition - Anindya Kundu (meganindya)