

Question 5

Implement image negation operation using logical NOT operation and verify results with using linear transformation function.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import math
```

Images to process

```
In [2]: 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 = [
    'a256',
    'ba256',
    'n256',
    'o256',
    'p256',
    'z256'
]

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]: # 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
    })

    # 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]: # Matrix dimensions
cols = 3
rows = -(-len(filenames) // cols)

# Create figure with rows x 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[int(idx // cols), idx % cols].set_title('"{}" {}'.format(
        filename + ext_inp,
        image.shape
    ))

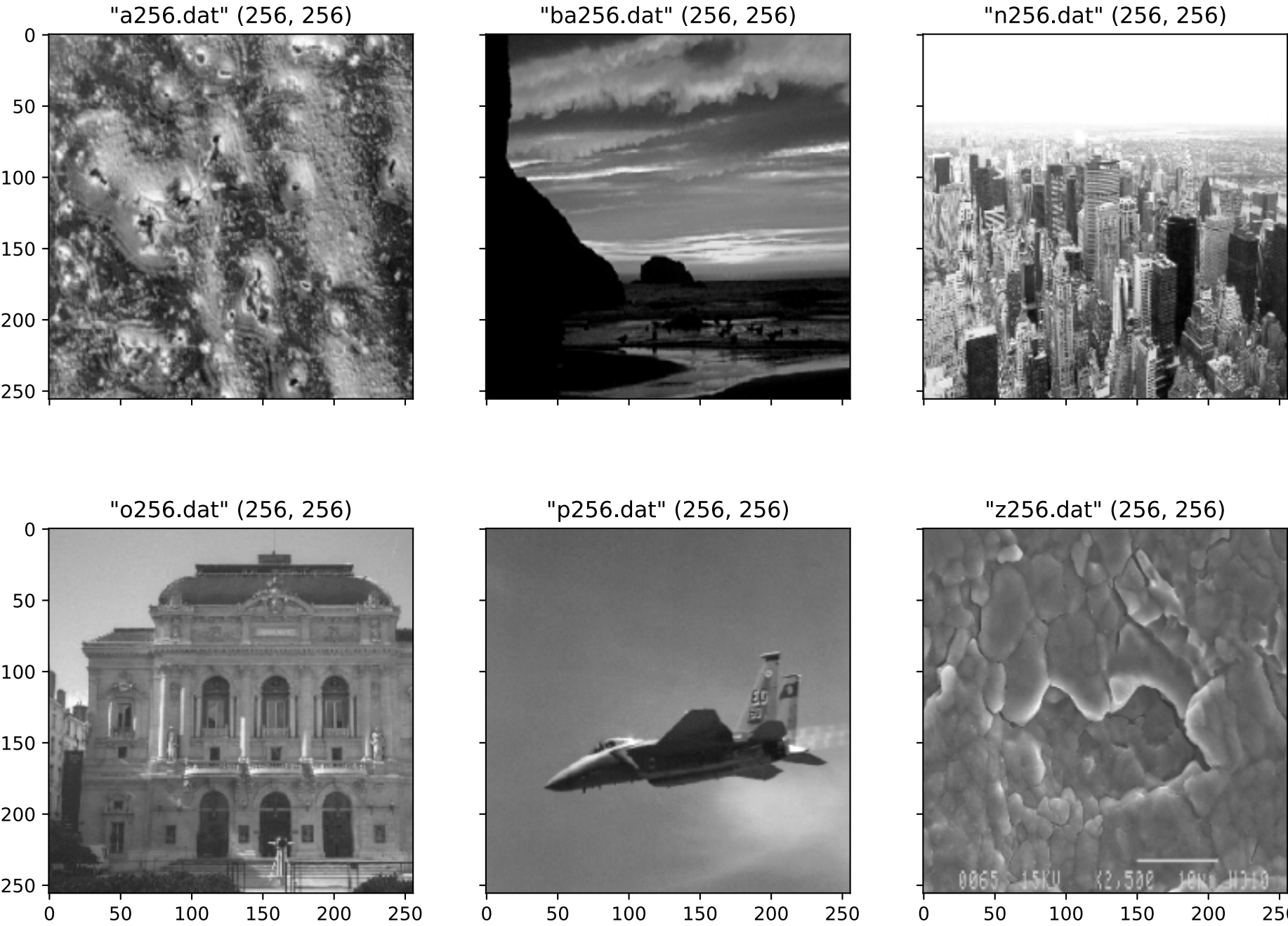
    # Add subplot to figure plot buffer
    axs[int(idx // cols), idx % cols].imshow(
        image,
        cmap='gray',
        vmin=0,
        vmax=255
    )

    # 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()
```



Negation

```
In [5]: def negate(image):
        height, width = image.shape

        neg_img = np.array(np.zeros(image.shape), dtype='uint8')
        for i in range(height):
            for j in range(width):
                pix = image[i][j]
                mask = 1
                neg = 0
                for b in range(8):
                    if not pix & mask:
                        neg |= 1 << b
                    mask = mask << 1
                neg_img[i][j] = neg

        return neg_img
```

```
In [6]: rows, cols = len(images), 2

        # Create figure with rows x 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']
            orig = image_dict['orig']

            neg_img = negate(orig)

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

            axs[idx, 1].set_title('negated')
```

```

    axs[idx, 1].imshow(neg_img, cmap='gray', vmin=0, vmax=255)

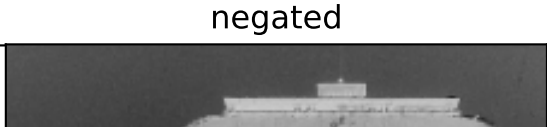
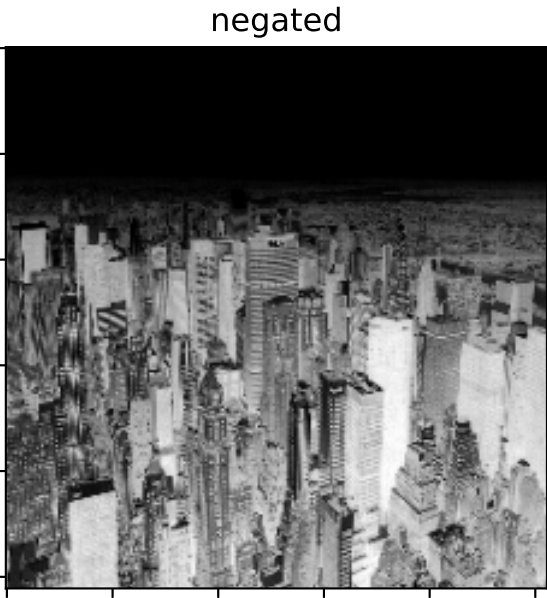
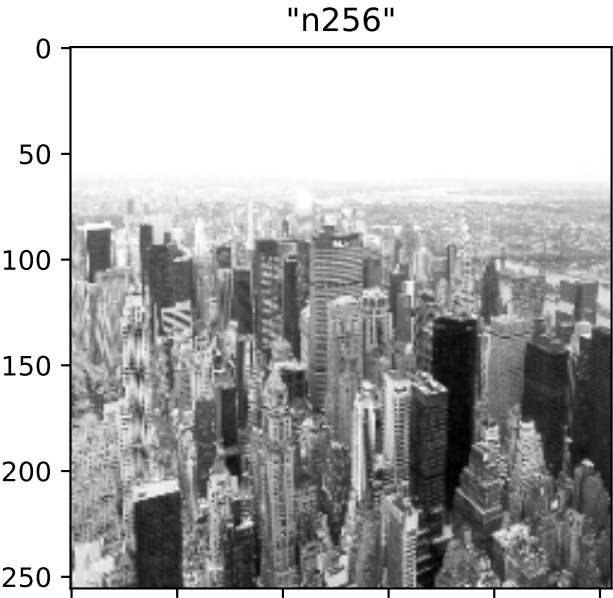
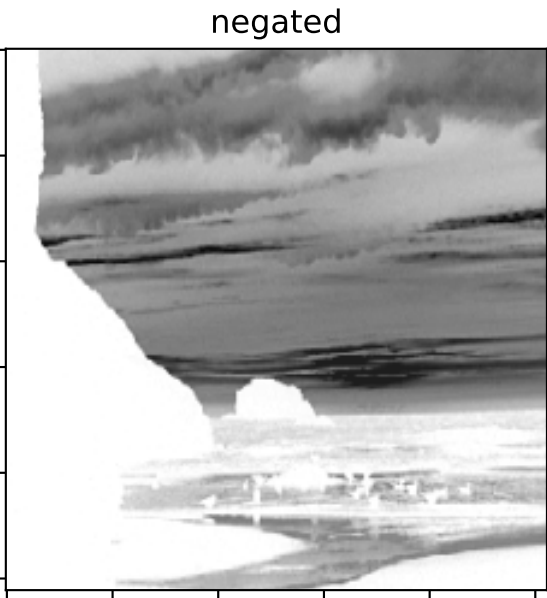
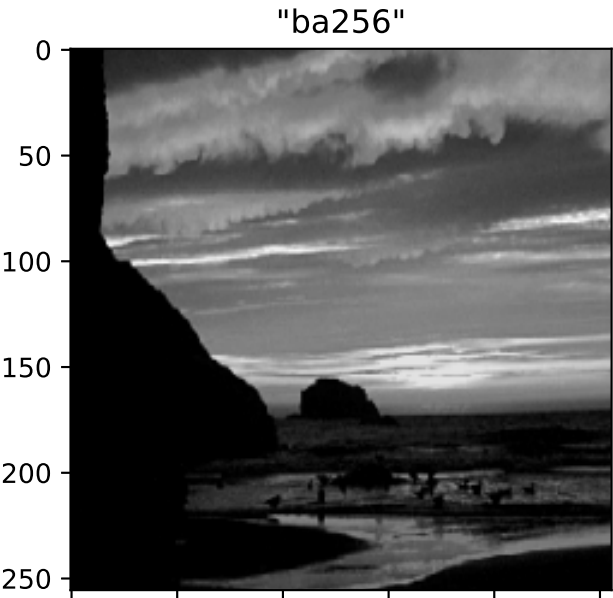
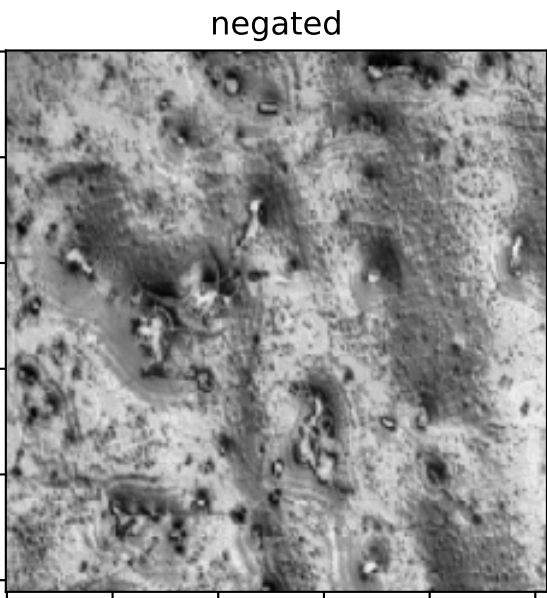
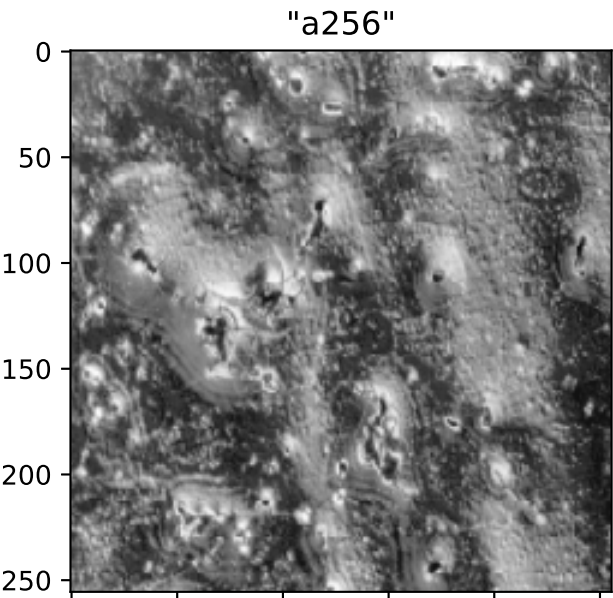
    # Save threshold image as .bmp file
    plt.imsave(
        path_out_conv + ext_out[1:] + '/' + filename + '_neg' + ext_out,
        neg_img,
        cmap='gray',
        vmin=0,
        vmax=255
    )

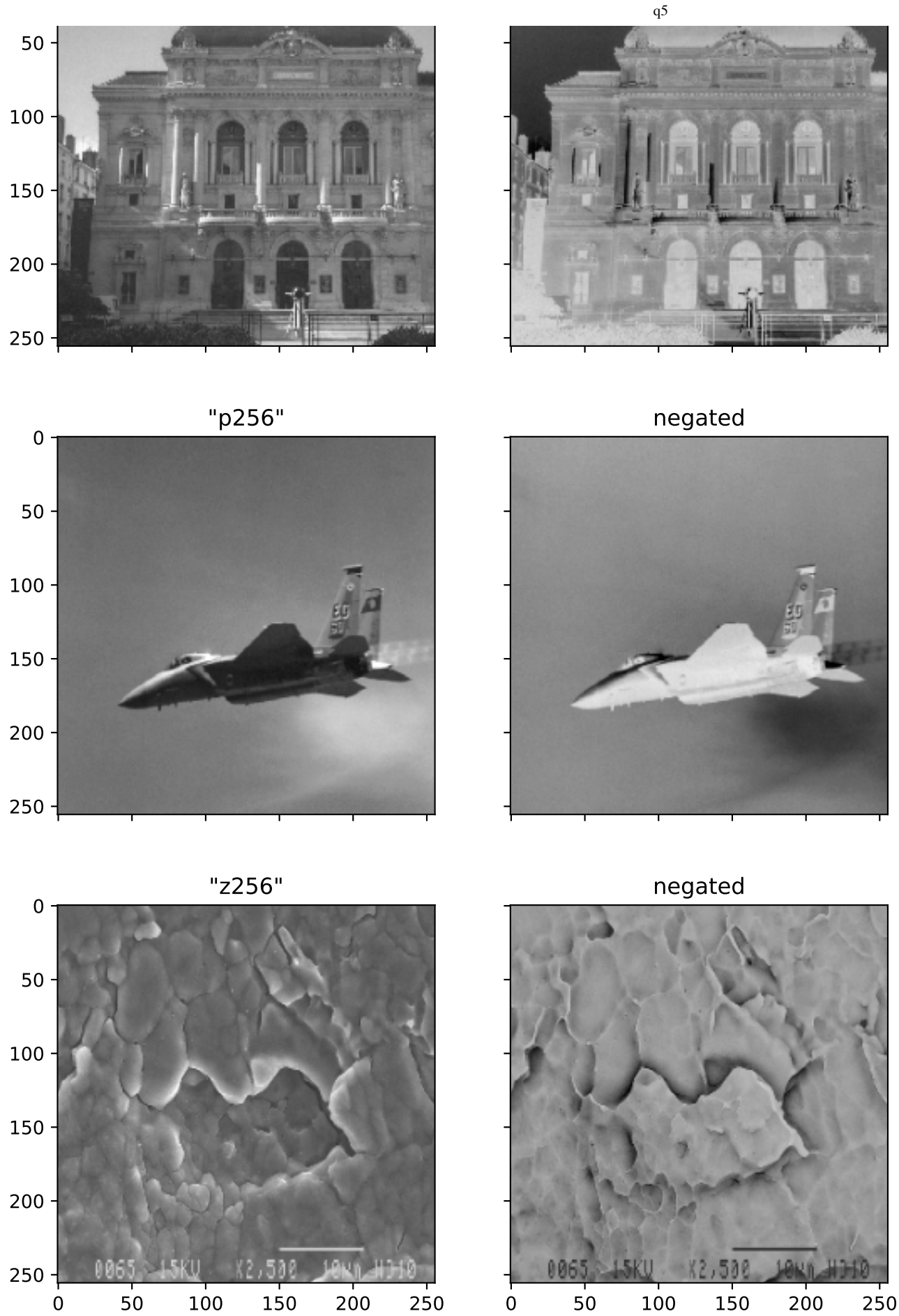
    # Save pixel values of threshold image as a 2D matrix in a .dat file
    np.savetxt(
        path_out_conv + ext_inp[1:] + '/' + filename + '_neg' + ext_inp,
        neg_img,
        fmt='%d',
        newline='\n'
    )

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

    # Save and display the figure
    plt.savefig('negated_comp.jpg')
    plt.show()

```





Resource

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