Exercise solutions for odd numbered problems

Chapter 1

1. If you are familiar with any other programming language, list the difference between that language and Python.

   Solution: Solution varies.

3. Create a list of state names such as states = ['Minnesota','Texas','New York','Utah','Hawaii']. Add another entry 'California' to the end of the list. Then, print all the values of this list.

   Solution:

```
>>> states = ['Minnesota','Texas','New York','Utah','Hawaii']
>>> states.append(California)
>>> print states
```

5. Create a 2D list of size 3-by-3 with the following elements 1,2,3| 4,5,6 | 6, 7,8

   Solution:

```
>>> a = [[1,2,3],[4,5,6],[6,7,8]]
```

```
To access the center element 5, use
>>> print a[1][1]
```

7. Look up documentation for 'join' method and join the content of the list ['Minneapolis','MN','USA'] and obtain the string 'Minneapolis, MN, USA'

   Solution:

```
>>> loc = ['Minneapolis','MN','USA']
>>> loc_string = ",".join(loc)
```

Chapter 2

1. Python is an open-source and free software. Hence, there are many modules created for image processing. Perform a research and discuss some of the benefits of each module over other.

   Solution: Solution varies.

3. Why is it more convenient to arrange the various functions as modules?

   Solution:

Python modules are

- Reusable. Thus, a Python module can be shared with others.
- Provide simple interface. The module functions can be documented, so that the interface to the user is transparent.

5. Create a numpy array of size 5-by-5 containing all random values. Determine the transpose and inverse of this matrix.

   Solution:

```
>>> import numpy as np
>>> a = np.random.rand(5,5)
>>> mytranspose = np.transpose(a)
>>> print mytranspose
>>> myinv = np.linalg.inv(a)
>>> print myinv
```

Chapter 3

1. An image of size 100-by-100 has isotropic pixel size of 2-by-2 microns. The number of pixels in the foreground is 1000. What is the area of the foreground and background in square microns?

Solution: Total number of pixels in the image $= 100x100 = 10,000$ pixels^2

Total number of foreground pixels (given) $= 1,000$ pixels^2

Total number of background pixels $= 10,000 - 1,000 = 9,000$ pixels^2

Area of each pixel $= 2x2 = 4$microns^2

Total area in square microns $= 9,000x4 = 36,000$ microns^2.

3. A histogram plots the frequency of occurrence of the various pixel values. This plot can be converted to a probability density function or pdf, so that the y-axis is the probability of the various pixel values. How can this be accomplished?

   Solution: Histogram contains the frequency of occurrence of each pixel value. Thus, histogram[i] = frequency of occurrence of pixel value i The pdf describes the probability of occurrence of each pixel value. It can be obtained by diving each value of histogram by the total number of pixels in the image. Thus,

$$pdf[i] = \frac{\text{histogram[i]}}{\text{total number of pixels in the image}}.$$

Chapter 4

1. Write a Python program to apply mean filter on an image with salt-and-pepper noise. Describe the output including the mean filter's ability to remove the noise.

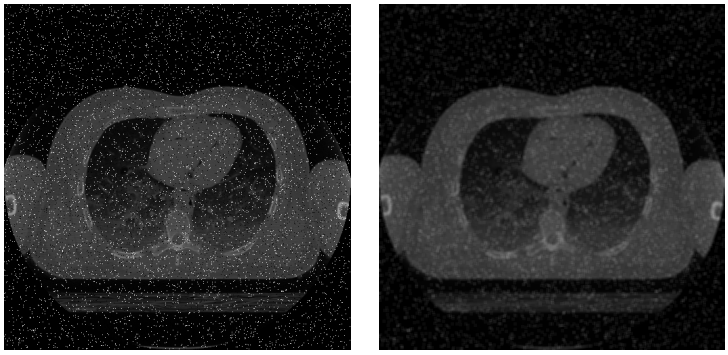
   Solution:

The following program is similar to the one described in the mean filter section. The difference is in the input image used. The image with salt-and-pepper noise is the 'ct_saltandpepper.png'.

```
import numpy as np
import scipy.ndimage
from scipy.misc.pilutil import Image

# opening the image and converting it to a greyscale image
a = Image.open('../figure/ct_saltandpepper.png').convert('L')
# initializing the filter of size 5 by 5
# the filter is divided by 25 for normalization
k = np.ones((5,5))/25
# performing convolution
b = scipy.ndimage.filters.convolve(a, k)
# b is converted from an ndarray to an image
b = scipy.misc.toimage(b)
# b.save('mean_ctsaltandpepper.png')
b.show()
```

The output of the program is given in Figure 1. The mean filter averages the pixel intensity of the neighbors. Hence it includes the bright and the dark pixel that need to be removed in its calculation of average. Hence, a mean filter cannot remove salt-and-pepper noise. As we discussed previously, a median filter is the best choice for removing this noise.



(a) Input image. (b) Output image.

Figure 1: The output of mean filter on salt-and-pepper noise.

3. Can max or min filter be used for removing salt-and-pepper noise?

   Solution:

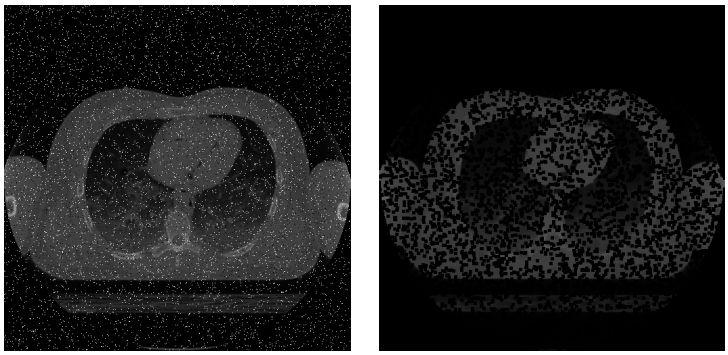
   The following program is similar to the one described in the min filter section. The difference is in the input image used. The image with salt-and-pepper noise is the 'ct_saltandpepper.png'.

```
import scipy.misc
import scipy.ndimage
from scipy.misc.pilutil import Image

# opening the image and converting it to a greyscale image
a = Image.open('../figure/ct_saltandpepper.png').convert('L')
# performing maximum filter
b = scipy.ndimage.filters.minimum_filter(a,size=5,
footprint=None,output=None,mode='reflect',
cval=0.0,origin=0)
# b is converted from an ndarray to an image
b = scipy.misc.toimage(b)
```

```
#b.save('max_ct_saltandpepper.png')
b.show()
```

The output of the program is given in Figure 2. The min filter replaces the given pixel with the minimum value of its neighbors. Since the 'pepper pixels', have minimum value, the given pixel value is replaced by the pixel value from 'pepper pixels'. Hence the 'pepper pixels' grow in size. Hence, a min filter cannot remove salt-and-pepper noise. Similar argument can be made for max filter. The max filter will cause the 'salt pixels' to grow. Hence there will be more brighter pixels. As we discussed previously, a median filter is the best choice for removing this noise.



(a) Input image. (b) Output image.

Figure 2: The output of min filter on salt-and-pepper noise.

5. Write a Python program to obtain the difference of the Laplacian of Gaussian (LoG).

Solution:

The pseudo code for the program will be as follows:

(a) Read the image

(b) Apply LoG assuming a standard deviation of 0.9 and store the image as im1

(c) Apply LoG assuming a standard deviation of 1.3 and store the image as im2

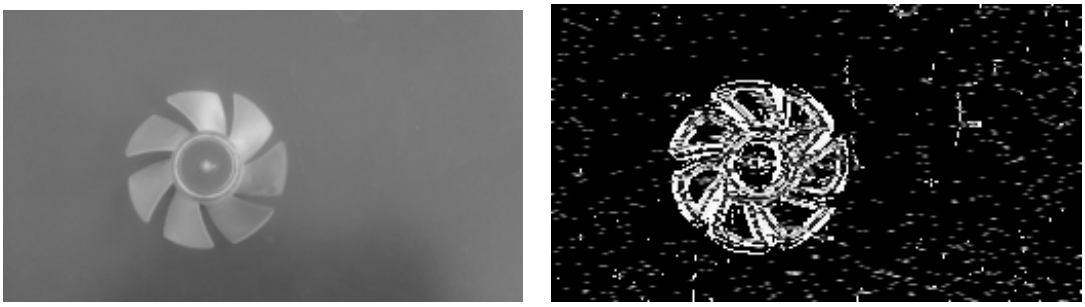
(d) Find the difference between the two images and store the resulting image.

```
import scipy.misc
import scipy.ndimage
from scipy.misc.pilutil import Image

# opening the image and converting it to grayscale
a = Image.open('../figure/spinwheel.png').convert('L')
# performing Laplacian of Gaussian with sigma = 0.9
im1 = scipy.ndimage.filters.gaussian_laplace(a,0.9,mode='reflect')
# performing Laplacian of Gaussian with sigma = 1.3
im2 = scipy.ndimage.filters.gaussian_laplace(a,1.3,mode='reflect')
# determining the difference for obtaining edge
b = im1-im2
# b is converted from an ndarray to an image
b = scipy.misc.toimage(b)
#b.save('diff_log.png')
b.show()
```

The output of the program is given in Figure 2. The min filter replaces the given pixel with the minimum value of its neighbors. Since the 'pepper pixels', have minimum value, the given pixel value is replaced by the pixel value from 'pepper pixels'. Hence the 'pepper pixels' grow in size. Hence, a min filter cannot remove salt-and-pepper noise. Similar argument can be made for max filter. The max filter will cause the 'salt

pixels' to grow. Hence there will be more brighter pixels. As we discussed previously, a median filter is the best choice for removing this noise.



(a) Input image. (b) Output image.

Figure 3: The output of the difference in laplacian of gaussian filter.

Chapter 5

3. An image transformation where every pixel value is multiplied by a constant (K). What will be the effect on the image assuming $K < 1, K = 1$ and $K > 1$?

   Solution: The change in pixel intensity can be better understood, if we consider one case. Let us assume that a given image is 8-bit (i.e.,) pixel value range is [0,255]. Let us consider one of the pixel in this image with an intensity of 100.

   - For $K < 1$, the output image will have a pixel value < 100. If the value of $K < 100$, the output pixel value > 1. If the value of $K > 100$, the output pixel value will be close to 0. In such case, the value will be truncated to 0. This will result in an undesirable blotchy effect.

- For $K = 1$, the output image will have a pixel value = 100 and hence will remain unchanged.
- For $K > 1$, the output image will have a pixel value > 100. If the value of K is in the interval (1,2], the output pixel value will be within the range of the image [0,255]. If the value of $K > 3$, the output image will have a pixel value > 300. Since the image can only store values less than 255, the pixel value will be truncated to 255. This will result in an undesirable blotchy effect.

Although this example might be trivial, the effect of multiplication factors causing round off or truncation in pixel intensity value can be seen in any image transformation.

5. The window or level operation allows us to modify the image, so that all pixel values can be visualized. What is the difference between window or level operation and image enhancement?

   Solution:

   The window or level operation does not change the underlying pixel intensity while image enhancement does.

Chapter 6

3. The central pixel in the Fourier image is brighter compared to other pixel. Why?

   Solution:

In section 6.3, we discussed the equation of Fourier transform of two-dimensional functions such as images. The central pixel correspond to $u = 0$ and $v = 0$. By substituting the value in the equation, it can seen that the central pixel value in the Fourier transformed image is the average of all the pixel intensities in the original image.

5. Consider an image of size M x M pixels that needs to be convolved with a filter of size N-by-N where $M >> N$. Comment about the most efficient method for convolving. Would it be convolution in spatial domain or Fourier?

   Solution:

   From our experience will convolution of images in spatial domain, we know that at every pixel position, we perform N^2 multiplication followed by N^2 additions. This is repeated for M^2 pixels. Thus the total operation is $2M^2N^2$. If $M = 10,000$ and $N = 100$, it results in 10^{12} operations.

   In Fourier domain, there is only one multiplication ($2M^2$ operations) and two Fourier transform operations. Hence, for large image data sets, the Fourier domain based convolution is efficient.

Chapter 7

3. What happens if you zoom in to the image using ImageJ while keeping the image size the same. Try different levels of zoom levels (2X, 3X, and 4X). Explain the cause of

change in threshold value for segmentation.

Solution:

Zooming in to the image while maintaining the image size will result in cropping of pixels outside the window. Thus some pixels will be lost.

Hint: This changes the content of the image significantly and hence the histogram and the segmentation threshold.

Chapter 8

1. Perform skeletonization on the image in Figure 8.2(a). What do you observe?

   Solution:

   The following program is similar to the one described in the skeletonization section. The difference is in the input image used.

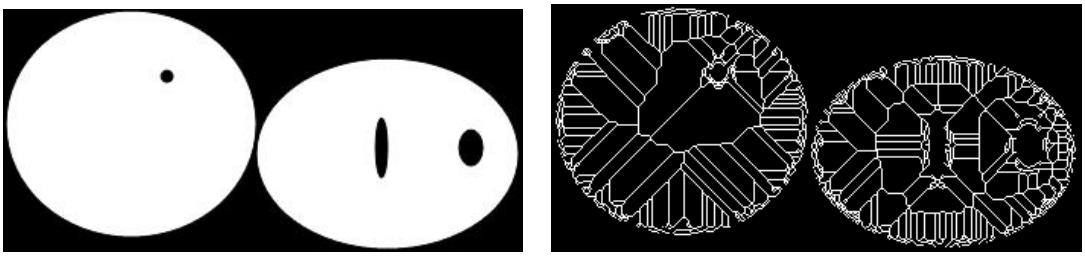
```
from scipy.misc.pilutil import Image
import numpy
from skimage.morphology import skeletonize
import scipy

# opening the image and converting it to grayscale
a = Image.open('../figure/dil_image1.png').convert('L')
# converting a to an ndarray and normalizing it
a = scipy.misc.fromimage(a)/numpy.max(a)
```

```
# performing skeletonization
b = skeletonize(a)
# converting b from an ndarray to an image
c = scipy.misc.toimage(b)
# saving the image

#c.save('skeleton.png')
c.show()
```

The output of the program is given in Figure 4. The left image is the original image and the right image is the skeleton. As it can be seen, the skeleton of circle is not a regular geometric shape but rather a series of lines. The number and complexity of the skeleton grows depending on the number of holes (dark regions) in the circle.



(a) Input image. (b) Output image.

Figure 4: The output of the skeletonization process.

3. Imagine an image containing two cells that are next to each other with few pixels overlapping, what morphological operation would you use to separate them?

   Solution:

Erosion reduces the size of object in an image and can be used to separate the two cells. If measuring the size of cell is the final outcome, then erosion will result in smaller size than the original. The problem can be overcome by using opening operation instead. In opening operation, the cells are eroded and then dilated immediately. The erosion operation separates the two cells. The dilation operation ensures that the cell retains their size.

Chapter 9

1. Hough transform is one method for finding the diameter of a circle. The process of finding diameter is slow. Suggest a method for determining the approximate diameter of a circle, given only pixels corresponding to the two blood vessels from an image segmented from Figure 9.3(a).

   Solution:

   The following program is similar to the one described in the regionprops section.

```
import numpy, math
import scipy.misc
from skimage.measure  import label
from scipy.misc.pilutil import Image
from skimage.measure import regionprops



# opening the image and converting it to grayscale
```

```
a = Image.open('../figure/houghcircles_segmented.png').convert('L')
# a is converted to an ndarray
a = scipy.misc.fromimage(a)
# labelling is preformed on a
c = label(a)
# c is converted from an ndarray to an image
c1 = scipy.misc.toimage(c)
# c1 is saved as label_output.png
#c1.save('label_output.png')
# on the labelled image c, regionprops is performed
d = regionprops(c)

print "The diameter of the circles are: "
for props in d:
    print 2*math.sqrt(props['Area']/math.pi)
```

The input to the program is given in Figure 5. The two circles are the segmented blood vessels obtained from Figure 9.3(a). The two circles are labelled using skimage.morphology.label. The labelled images are then measured using regionprops. The regionprops function calculates many properties of the object but since we are interested only in area, we specify the same. The variable 'd' is a list of dictionary. The individual values of the object's area are obtained in the loop and the diameter is determined from the area of the circle.

3. Consider an image with 100 coins of various sizes spread on a uniform background.

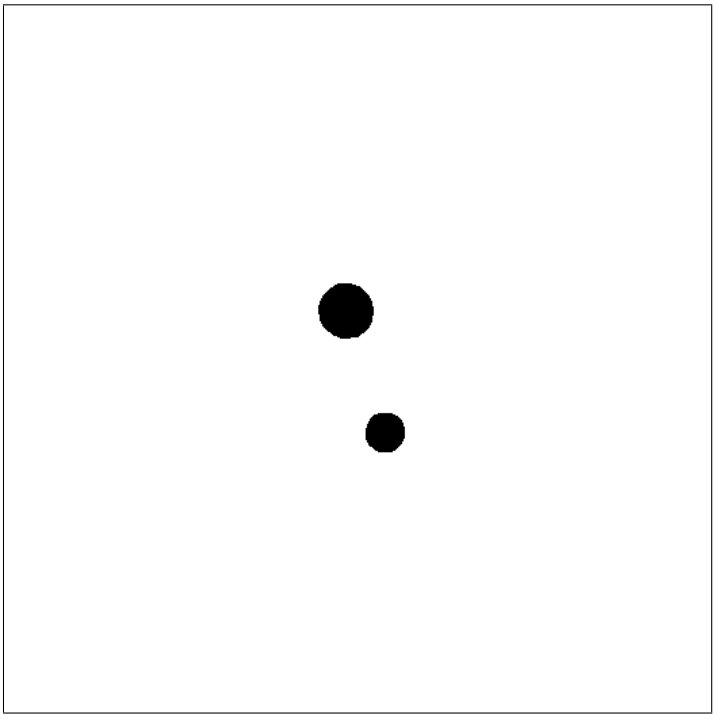


Figure 5: Segmented image of two blood vessels.

Assume that the coins do not touch each other, write a pseudo code to determine the number of coins for each size. Brave soul: Write a Python program to accomplish the same.

Solution:

The segmentation and counting can be achieved using the following pseudo code.

(a) The image has to be segmented, so that the coins can be separated from the background.

(b) The regionprops function is called with Area as one of the inputs similar to the previous example.

(c) Prepare a histogram of various sized coins. The x-axis will be the coin size (i.e., area) and the y-axis is the count. The diameter calculated from area can be used

as an alternate for the x-axis.

The segmentation method would depend on the nature of histogram in the original image.

Chapter 10

5. What is the HU value of a material whose linear attenuation coefficient is half of the linear attenuation coefficient of water?

   Solution:

   HU is given by

$$HU = \left(\frac{\mu - \mu_w}{\mu_w} \right) * 1000 \tag{0.1}$$

   Substitute $\mu = \frac{\mu_w}{2}$ in the above equation. Thus,

$$HU = \left(\frac{\frac{\mu_w}{2} - \mu_w}{\mu_w} \right) * 1000 \tag{0.2}$$

$$HU = \left(\frac{-\frac{\mu_w}{2}}{\mu_w} \right) * 1000 \tag{0.3}$$

$$HU = -500 \tag{0.4}$$

Nuclei	γ (MHz/T)	f (MHz)
H^1	42.58	63.87
P^{31}	17.25	25.88
Na^{23}	11.27	16.91
C^{13}	10.71	16.07

Table 1: Gyromagnetic ratio and Larmor frequency of few elements.

Chapter 11

1. Calculate the Larmor frequency for all atoms listed in Table 11.1 assuming magnetic field strength of 1.5T.

   Solution: The Larmor frequency is given by

   $$f = \gamma B \tag{0.5}$$

   where γ is the Gyromagnetic ratio, f is the Larmor frequency, and B is the strength of the external magnetic field.

   Table 11.1 lists the value of γ for various elements. Using the above equation, we can calculate the Larmor frequency (f). The calculated values are shown in Table 1.

3. If the plot in Figure 11.3 is viewed looking down in the -z direction, the magnetic field path will appear as a circle. Why?

   Solution:

   The values of Mx and My have cos and sin dependencies, similar to the parametric form of a circle.

Chapter 12

1. If the objective has a magnification of 20x and the eye-piece has magnification of 10x. What is the total magnification?

   Solution:

   Based on equation 12.2, the total magnification is the product of objective and eye-piece. Hence the total magnification is 200x.

3. In the same turret setup, if a cell occupies 10% of the field of view for an objective magnification of 20x, what would be the field of view percentage for 40x?

   Solution:

   A 40x magnification causes the object to look twice as big as 20x magnification. Hence, an object occupying 10% field of view under 20x will occupy 20% field of view under 40x.

Chapter 13

1. The accelerating voltage of an SEM is 10kV. Calculate the wave- length of the generated electron?

   Solution:

Plug in the value of voltage (V) as 10,000 in equation 13.5. This gives a wavelength of 0.0122 nano-meter.