Exercise solutions for odd numbered problems

Chapter 1

1. If you are familiar with any other programming language, list the differences 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 the `join` method and join the contents 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 research and discuss some of the benefits of each module over another.

   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. Modify the program from Question 4 to read a Microsoft Excel file instead.

Solution:

You need to install openpyxl and cv2 module in order to run the code below.

The list of images in the xlsx files is opened using the load_workbook function. The list of active sheets and the list of cells are obtained. A for-loop is then used to loop through all the cells. The full path to the file is obtained by concatenating the folder name, 'figure' and the file name which is the content of the cell. Finally OpenCV is used to read the image and the image shape is printed.

```
import os
import openpyxl
import cv2

book = openpyxl.load_workbook('images.xlsx')
# get all active sheets
sheet = book.active

# get the 9 cells in the first sheet
cells = sheet['A1': 'A9']
```

```
# get content of each cell (i.e.,) the file name and create
# full path by adding the folder name
for c in cells:
    fname = c[0].value # the content of a cell
    fname_path = os.path.join('figure', c[0].value)
    print(fname_path)

    # read the content of the file using cv2 and print shape
    im = cv2.imread(fname_path)
    print(im.shape)
```

6. 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 $microns^2$?

   Solution: Total number of pixels in the image $= 100 * 100 = 10,000$ 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 $= 2 * 2 = 4$ microns^2

   Total area of foreground pixels in micron squared $= 1,000 * 4 = 4,000$ microns^2.

   Total area of background pixels in micron squared $= 9,000 * 4 = 36,000$ 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 dividing 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

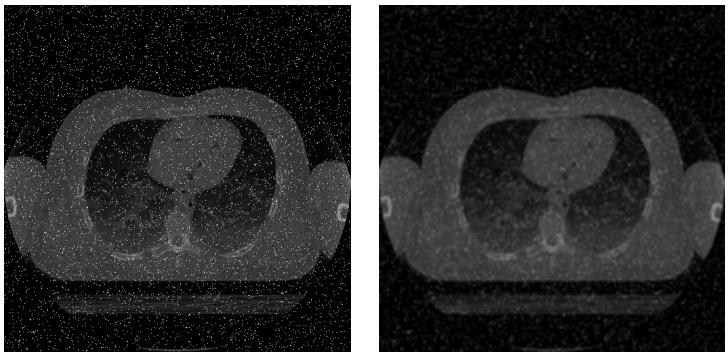
1. Write a Python program to apply a mean filter on an image with salt-and-pepper noise. Describe the output, including the mean filter's ability to remove the noise.

   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
import cv2

# opening the image and converting it to a greyscale image
a = cv2.imread('figure/ct_saltandpepper.png')
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# 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)

cv2.imwrite('mean_ctsaltandpepper.png', b)
```

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 filter or min filter be used for removing salt-and-pepper noise?

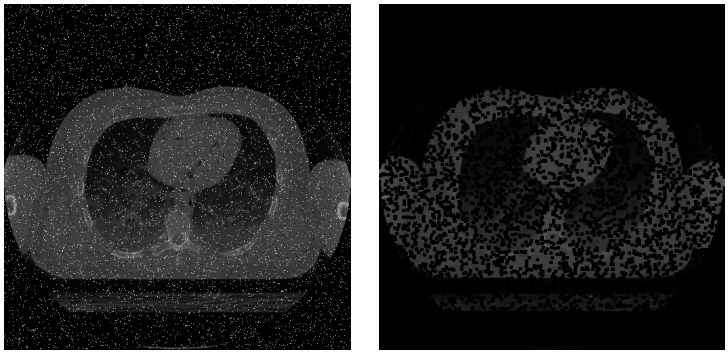
Solution:

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.ndimage
import cv2

# opening the image and converting it to a greyscale image
a = cv2.imread('figure/ct_saltandpepper.png')
```

```
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)


# performing minimum filter
b = scipy.ndimage.filters.minimum_filter(a,size=5,
footprint=None,output=None,mode='reflect',
cval=0.0,origin=0)


cv2.imwrite('min_ctsaltandpepper.png', b)
```

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). The pseudo code for the program will be as follows:

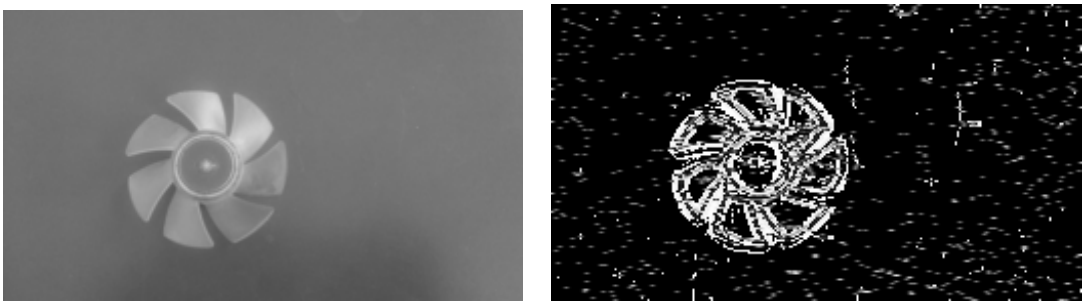
   (a) Read the image.

   (b) Apply the LoG filter assuming a standard deviation of 0.1 and store the image as im1.

   (c) Apply the LoG filter assuming a standard deviation of 0.2 and store the image as im2.

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

```
import scipy.ndimage
import cv2

# opening the image and converting it to a greyscale image
a = cv2.imread('figure/spinwheel.png')
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# 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

cv2.imwrite('diff_log.png', b)
```



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

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

Chapter 5

3. Consider 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 $K > 1$? What will be the impact on the histogram of the output image in relation to the input image?

   Solution: The change in pixel intensity can be better understood, if we consider one specific 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 which is in the range of [0,255].
   - 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 and image enhancement?

   Solution:

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

7. In sigmoid correction, the choice of the cutoff and gain will determine the quality of the output image. The readers are recommended to try different settings for the hyper-parameter to understand their effect.

   Solution: You need to modify the parameter 'gain' used in the example code. For the input image given in the sigmoid correction example, we tried multiple values of gain. A gain of 3 did not enhance the image in comparison to the input image. A gain of 9 improved the contrast in the image. Finally a gain of 27 improved it significantly albeit at the cost of the making the image darker.

Chapter 6

1. Consider any of the images used in this chapter. Then, rotate or translate the image by various angles and distance, and for each case, study the histogram. Are the histograms of the input and output image different for different transformations?

   Solution: For small values of rotation and translation the histogram might exhibit a small difference between the input image and the transformed image. To understand this better, consider an image of size 100 x 100. Let's say the image is translated along the x-axis by 1 pixel and also assume the image size will not change due to transformation and that any new pixels that are created are assigned a pixel value of 0. When the 10,000 pixels (100 x 100) when translated by 1 pixel along the x-axis, then 100 pixels in the last column will be lost while 100 pixels will be gained with a pixel value of 0. This changes the pixel values of only 100 pixels out of 10,000, a small proportion of only 1%. This will not significantly impact the histogram.

   However for large values of rotation and translation the histogram will be significantly different between the input image and the transformed image as more pixels are affected.

Chapter 7

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

   Solution:

   In section 7.3, 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 10,000-by-10,000 pixels that needs to be convolved with a filter of size 100-by-100. Comment about the most efficient method for convolving. Would it be convolution in the spatial domain or Fourier?

   Solution:

   In convolution of images in spatial domain we know that at every pixel position, we perform N^2 multiplication followed by N^2 additions where N x N is the size of the filter. This is repeated for M^2 pixels where M x M is the image size. 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 8

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

   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 9

1. Perform skeletonization on the image in Figure 9.2(a). What do you observe?

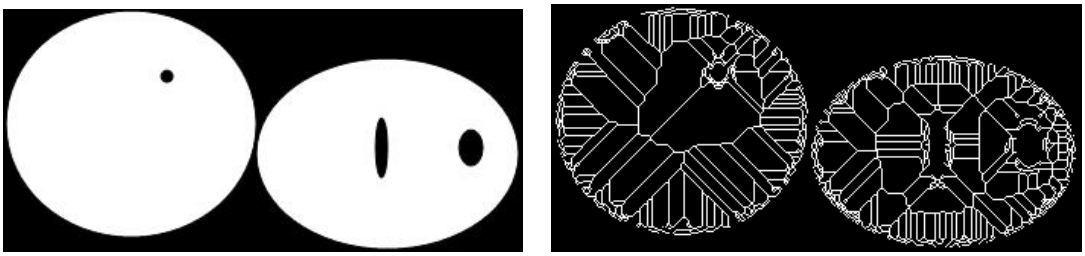
   Solution:

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

```
import numpy as np
from skimage.morphology import skeletonize
import scipy
import cv2

# opening the image and converting it to a greyscale image
a = cv2.imread('figure/dil_image1.png')
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# converting a to an ndarray and normalizing it
a = a/np.max(a)
# performing skeletonization
b = skeletonize(a)

# b is a binary image, we can convert it to gray scale
```

```
# by multiplying by a constant like 255
c = b*255
cv2.imwrite('skeleton.png', c)
```

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 a 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 the opening operation, the cells are eroded and then dilated immediately. The erosion operation separates the two cells. The dilation operation in the opening ensures that the cell retains their size.

Chapter 10

1. The Hough transform is one method for finding the diameter of a circle. This process of finding the diameter is slow. Suggest a method for determining the approximate diameter of a circle, given only pixels corresponding to the two blood vessels in Figure 10.3(a).

   Solution:

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

```
import math
from skimage.morphology  import label
from skimage.measure import regionprops
import cv2

# opening the image and converting it to grayscale
a = cv2.imread('figure/houghcircles_segmented.png')
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# labelling is preformed on a
c = label(a)

# c1 is saved as label_output.png
cv2.imwrite('label_output.png', c)

# 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. 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 this. Hint: regionprops will be needed.

   Solution:

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

   (a) The image is 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.

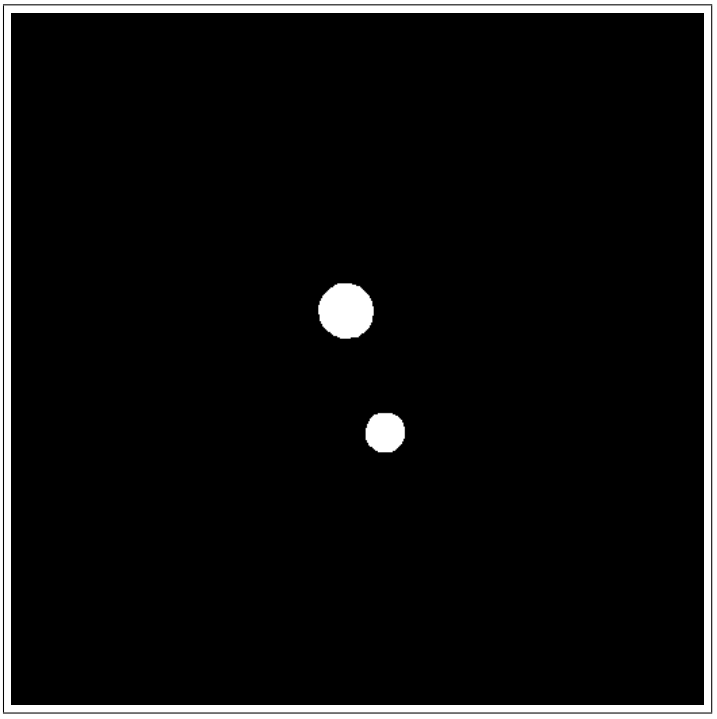


Figure 5: Segmented image of two blood vessels.

(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 11

1. You are given a neuron that performs addition of y = x1*w1+x2*w2, where x1 and x2 are the inputs and w1 and w2 are weights. Write the back-propagation equation for it. Also write the update equation for w1 and w2.

   Solution:

   We are given

   $$y = w_1 x_1 + w_2 x_2 \tag{0.1}$$

   Let's assume that we are solving a regression problem. Hence, we can assume a squared loss as the loss function

   $$L = (\hat{y} - y)^2 \tag{0.2}$$

   The derivative of the loss w.r.t to $\hat{y}$ is

   $$\frac{\partial L}{\partial \hat{y}} = 2 * (\hat{y} - y) \tag{0.3}$$

   The derivative of the loss w.r.t to w_1 is

   $$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_1} = 2 * (\hat{y} - y) * x_1 \tag{0.4}$$

   Similarly the derivative of the loss w.r.t to w_2 is

   $$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2} = 2 * (\hat{y} - y) * x_2 \tag{0.5}$$

The update equation is given by,

$$w_1 = w_1 - \epsilon \frac{\partial L}{\partial w_1} \tag{0.6}$$

where ϵ is the learning rate.

If we substitute the value of the derivative from 0.4 in to 0.6, we obtain

$$w_1 = w_1 - \epsilon * 2 * (\hat{y} - y) * x_1 \tag{0.7}$$

Similarly we can obtain the new value of w_2 as,

$$w_2 = w_2 - \epsilon * 2 * (\hat{y} - y) * x_2 \tag{0.8}$$

At the start of the training, we assign a random value to w_1 and w_2. Then we use a single data point and compute the predicted value ($\hat{y}$) during forward propagation using equation 0.1. We then use the predicted value to calculate new value of w_1 and w_2 using equations 0.7 and 0.8 respectively. We will continue this process until we obtain the optimal value of w_1 and w_2.

3. Why is sigmoid no longer popular as an activation function? Conduct research on this topic.

Solution:

Hint: Read about vanishing gradient problem.

Chapter 12

1. What is the effect of increasing the number of convolution layers in a neural network?

   Solution:

   Increasing the number of convolution layers will increase the computation. To understand the impact, let's consider an example. Consider an image of size MxM convolved with a kernel of size nxn. During convolution for every pixel position in the convolved image, we need to perform nxn multiplication and nxn addition for a total of 2xnxn computation. Since there are MxM pixels, the total computation will be $2M^2n^2$. If there are k convolution layers, then there will be $2kM^2n^2$. Thus if we increase the number of convolution layers, we will increase the amount of computation.

   It has been found that for complex classification problems, typically need more number of layers.

Chapter 13

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

$$HU = \left(\frac{\mu - \mu_w}{\mu_w}\right) * 1000 \tag{0.9}$$

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

$$HU = \left(\frac{\frac{\mu_w}{2} - \mu_w}{\mu_w} \right) * 1000 \tag{0.10}$$

$$HU = \left(\frac{-\frac{\mu_w}{2}}{\mu_w} \right) * 1000 \tag{0.11}$$

$$HU = -500 \tag{0.12}$$

Chapter 14

1. Calculate the Larmor frequency for all atoms listed in Table 14.1 assuming magnetic field strength of 1.5T.

   Solution: The Larmor frequency is given by

   $$f = \gamma B \tag{0.13}$$

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

   Table 14.1 in the book 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 14.3 is viewed looking down in the z direction, the magnetic field path will appear as a circle. Why?

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.

Solution:

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

Chapter 15

1. If the objective has a magnification of 20X and the eyepiece has a magnification of 10X, what is the total magnification?

   Solution:

   Based on the Equation 15.2, 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 16

1. The accelerating voltage of an SEM is 10kV. Calculate the wavelength of the generated electron.

   Solution:

   Plug in the value of voltage (V) as 10,000 in to Equation 16.5. This gives a wavelength of 0.0122 nano-meter.