Name:	Divya Madhav Patil
Roll No:	
Class/Sem:	BE/VII
Experiment No.:	3
Title:	Bio Medical Image Preprocessing, Segmentation.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim- Bio Medical Image Preprocessing, Segmentation

Theory:

There are 5 preprocessing stages of images.

These steps are: Transforming to HU, Removing Noises, Tilt Correction, Crop Images and Padding.
After applying these preprocessing steps to data, we see that model accuracy got increased significantly. After loading our image data in DICOM format, we will transform it to Hounsfield Unit form.

Transforming to HU

The Hounsfield Unit (HU) is a relative quantitative measurement of the intensity of radio waves used by radiologists for better explanation and understanding of computed tomography (CT) images. The absorption/attenuation coefficient of radiation within a tissue is used during CT reconstruction to produce a grayscale image. The linear transformation produces a Hounsfield scale that displays as gray tones. More dense tissue, with greater X-ray beam absorption, has positive values and appears bright; less dense tissue, with less X-ray beam absorption, has negative values and appears dark.

Removing Noises

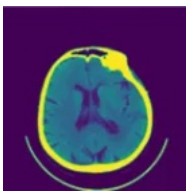
During preprocess, removing noises is a very important stage since, the data is improved after the implementation we can see it more clearly. So, model can be trained better.
Tilt Correction:
Tilt correction is the alignment of brain image in a proposed way. When tilt experienced by brain CT images, it may result in misalignment for medical applications. It is important because when we train the model, it can see the whole data through the same alignment. Manually correcting the tilt on a large scale data is time-consuming and expensive. Thus, there is a need for an automatic way of performing tilt correction in preprocessing before the training.

Crop Image and Add Pad:

Cropping image is needed to place the brain image at the center and get rid of unnecessary parts of image. Also, some brain images might be placed in different location within general image. By cropping image and adding pads, we will make sure almost all the images are in same location within general image itself.

Code :

```
import cv2
import numpy as np
from IPython.display import Image, display
from matplotlib import pyplot as plt


# Plot the image
def imshow(img, ax=None):
  if ax is None:
    ret, encoded = cv2.imencode(".jpg", img)
    display(Image(encoded))
  else:
    ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    ax.axis('off')

#Image loading
img = cv2.imread("/content/unnamed.png")

#image grayscale conversion
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Show image
imshow(img)
```



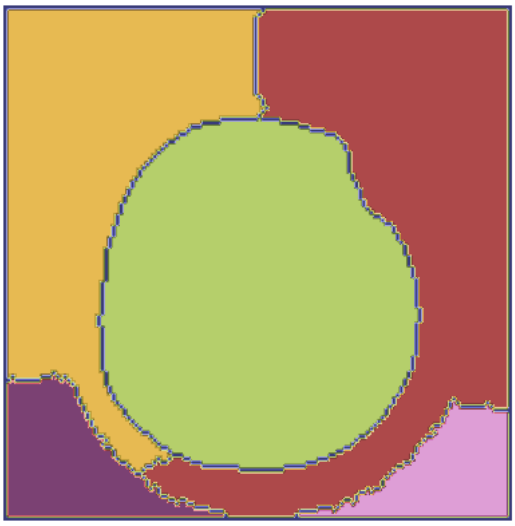
```
#Threshold Processing
ret, bin_img = cv2.threshold(gray,
              0, 255,
              cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
imshow(bin_img)
```

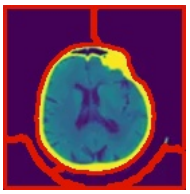
```
# noise removal
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
bin_img = cv2.morphologyEx(bin_img,
            cv2.MORPH_OPEN,
            kernel,
            iterations=2)
imshow(bin_img)
```

```
# Marker labelling
# sure foreground
ret, markers = cv2.connectedComponents(sure_fg)

# Add one to all labels so that background is not 0, but 1
markers += 1
# mark the region of unknown with zero
markers[unknown == 255] = 0

fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(markers, cmap="tab20b")
```

```
ax.axis('off')
plt.show()
```

```
# watershed Algorithm
markers = cv2.watershed(img, markers)

fig, ax = plt.subplots(figsize=(5, 5))
ax.imshow(markers, cmap="tab20b")
ax.axis('off')
plt.show()


labels = np.unique(markers)

coins = []
for label in labels[2:]:

# Create a binary image in which only the area of the label is in the foreground
#and the rest of the image is in the background
    target = np.where(markers == label, 255, 0).astype(np.uint8)

# Perform contour extraction on the created binary image
    contours, hierarchy = cv2.findContours(
        target, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE
    )
    coins.append(contours[0])

# Draw the outline
img = cv2.drawContours(img, coins, -1, color=(0, 23, 223), thickness=2)
imshow(img)
```





Conclusion:

The preprocessing and segmentation of biomedical images are critical steps in preparing data for further analysis, diagnosis, or model training. These steps help enhance the quality and consistency of the image data, which is essential for accurate medical assessments and the development of machine learning models in the healthcare domain. The application of these preprocessing techniques can significantly improve model accuracy and ensure that the images are ready for use in medical research and clinical practice.