

```

1  from google.colab import drive
2  drive.mount('/content/drive')
3  import numpy as np
4  import scipy.io
5  import pandas as pd
6  from skimage import color
7  from skimage import io
8  from skimage.transform import radon, iradon, iradon_sart, rescale
9  from skimage.metrics import structural_similarity
10 from skimage.metrics import peak_signal_noise_ratio
11 import math
12 import cv2
13 from sklearn import metrics
14 import matplotlib.pyplot as plt

```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.moun

```

1 ctScans = scipy.io.loadmat('/content/drive/My Drive/CCE-AIMIA/ctscan_hw1.mat')
2 ctMasks = scipy.io.loadmat('/content/drive/My Drive/CCE-AIMIA/infmsk_hw1.mat')

```

```

1 (ms,ns,cs)= (ctScans['ctscan']).shape
2 (mm,nm,cm)= (ctMasks['infmsk']).shape
3 print((ms,ns,cs))
4 print((mm,nm,cm))

```

```

(512, 512, 3554)
(512, 512, 3554)

```

```

1 ctscansarray = []
2 ctmaskarray = []
3 for i in range(cm):
4     ctscansarray.append((ctScans['ctscan'][:, :, i]))
5     ctmaskarray.append((ctMasks['infmsk'][:, :, i]))

```

```

1 kmeansSeg_image = []
2 labels_reshaped = []
3 k = 3 # number of clusters (K)
4 # define stopping criteria
5 criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
6 for i in range(len(ctscansarray)):
7     # reshape the image to a 2D array of pixels
8     image = ctscansarray[i]
9     ct_pixel_values = image.reshape(-1,1)
10    # convert to float
11    ct_pixel_values = np.float32(ct_pixel_values)
12    #print(ct_pixel_values.shape)

```

```

13 _, labels, (centers) = cv2.kmeans(ct_pixel_values, k, None, criteria, 10, cv2.KMEANS_R/
14 # convert back to 8 bit values
15 centers = np.uint8(centers)
16 # flatten the labels array
17 labels = labels.flatten()
18 # convert all pixels to the color of the centroids
19 segmented_image = centers[labels.flatten()]
20 # reshape back to the original image dimension
21 segmented = segmented_image.reshape(image.shape)
22 kmeansSeg_image.append(segmented)
23 labels_resaped.append(labels.reshape(512,512))

```

```

1 tp = 0
2 tn = 0
3 fn = 0
4 fp = 0
5 Sensitivity = []
6 Specificity = []
7 Accuracy = []
8 Dice_score = []
9 (rows, columns) = (512, 512)
10 for m in range(len(ctmasksarray)):
11     ground_truth = ctmasksarray[m]
12     KsegLabels = labels_resaped[m]
13     for i in range(rows):
14         for j in range(columns):
15             if ground_truth[i][j] == 1 and KsegLabels[i][j] == 0:
16                 tp = tp + 1
17             if ground_truth[i][j] == 2 and KsegLabels[i][j] == 2:
18                 tn = tn + 1
19             if ground_truth[i][j] == 1 and KsegLabels[i][j] == 2:
20                 fn = fn + 1
21             if ground_truth[i][j] == 2 and KsegLabels[i][j] == 0:
22                 fp = fp + 1

```

```

23
24     try:
25         TPR = float(tp)/(tp+fn)
26         Sensitivity.append(TPR)
27         FPR = float(tn)/(tn+fp)
28         Specificity.append(FPR)
29         Acc = ((tp+tn)/(tp+tn+fn+fp))*100
30         Accuracy.append(Acc)
31         dice = float(2*tp)/((2*tp)+fp+fn)
32         Dice_score.append(dice)
33     except ZeroDivisionError:
34         TPR=0
35

```

```

1 Sum_Sensitivity=0
2 Sum_Specificity=0
3 Sum_Accuracy=0

```

```

4 Sum_Dice_score=0
5 print ('\n*****Average Accuracy, Sensitivity, Specificity, Avg Dice_Socre*****')
6
7 for i in range(len(Sensitivity)):
8     Sum_Sensitivity+=Sensitivity[i]
9     Sum_Specificity+=Specificity[i]
10    Sum_Accuracy+=Accuracy[i]
11    Sum_Dice_score+=Dice_score[i]
12
13 Avg_sensitivity = Sum_Sensitivity/len(Sensitivity)
14 print("\nAverage Sensitivity is:",Avg_sensitivity)
15 Avg_Specificity = Sum_Specificity/len(Specificity)
16 print("\nAverage Specificity is:", Avg_Specificity)
17 Avg_Accuracy = Sum_Accuracy/len(Accuracy)
18 print("\nAverage Accuracy(%):" ,Avg_Accuracy)
19 Avg_Dice_score = Sum_Dice_score/len(Dice_score)
20 print("\nAverage Dice_score:" ,Avg_Dice_score)

```

*****Average Accuracy, Sensitivity, Specificity, Avg Dice_Socre*****

Average Sensitivity is: 0.4917266328796307

Average Specificity is: 0.4707865949677128

Average Accuracy(%): 47.322597631898546

Average Dice_score: 0.11304791992172793

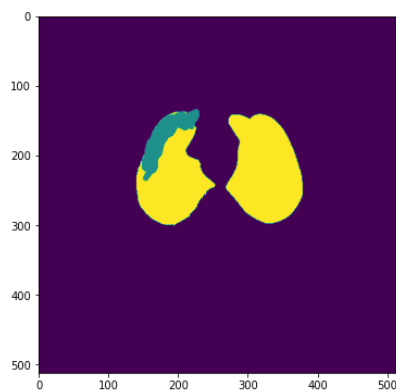
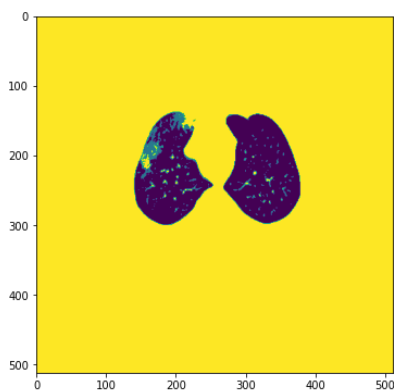
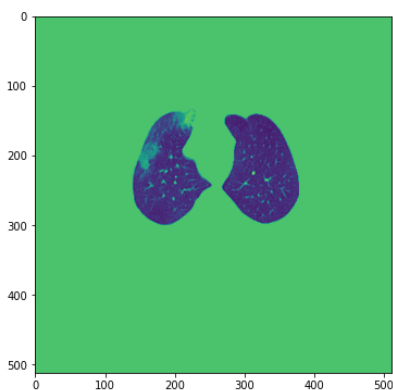
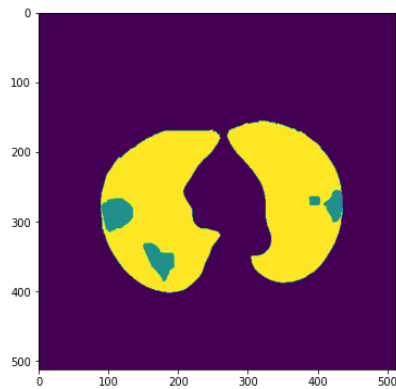
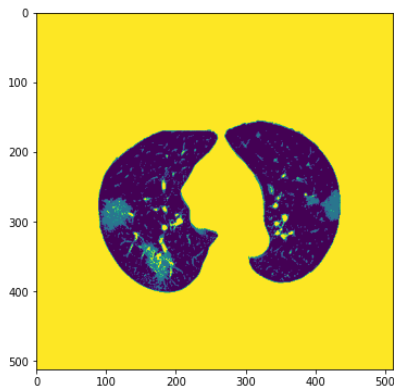
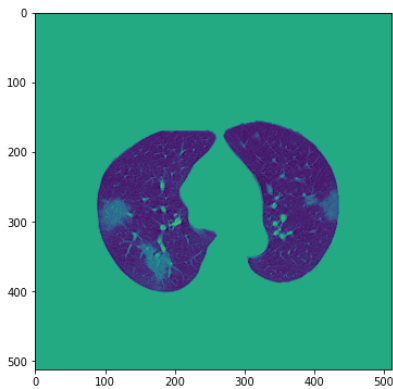
```

1 print ('\n*****Ctslice, k-means segmentation, infection mask*****')
2 num1 = np.random.randint(0, len(kmeansSeg_image))
3 num2 = np.random.randint(0, len(kmeansSeg_image))
4 # show the image
5 fig, (ax1,ax2) = plt.subplots(2, 3,figsize=(20, 20))
6 ax1[0].imshow(ctscansarray[num1])
7 ax1[1].imshow(kmeansSeg_image[num1])
8 ax1[2].imshow(ctmasksarray[num1])
9 ax2[0].imshow(ctscansarray[num2])
10 ax2[1].imshow(kmeansSeg_image[num2])
11 ax2[2].imshow(ctmasksarray[num2])

```

*****Ctslice, k-means segmentation, infection mask*****

<matplotlib.image.AxesImage at 0x7f6e6344c250>



[Colab paid products](#) - [Cancel contracts here](#)

