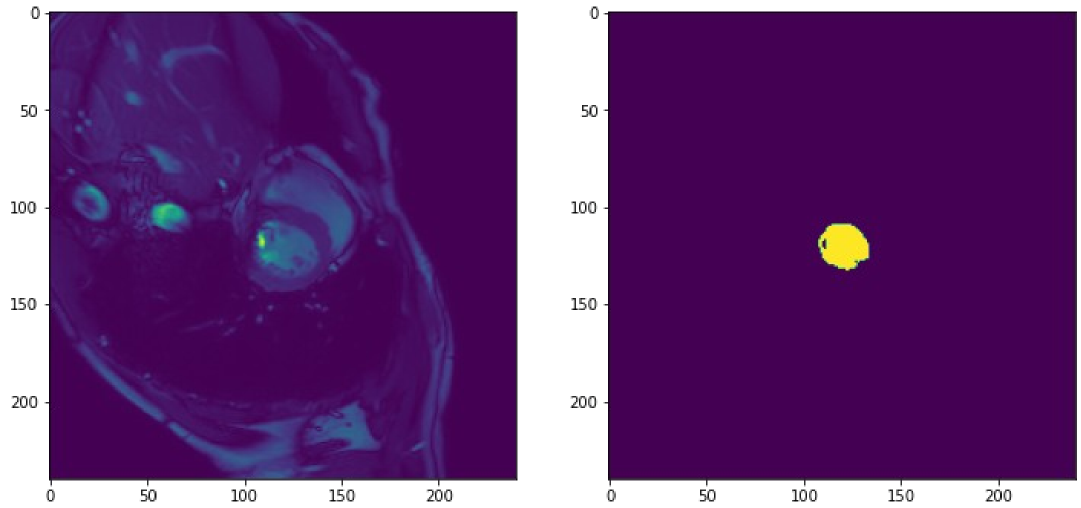
Bishop’s university(Medical Imaging)
(Dr Russell Butler – Summer 2022)
Members:

Razieh Shahsavar (002 341 606) Maryam Bayatzadeh(002 338 161)

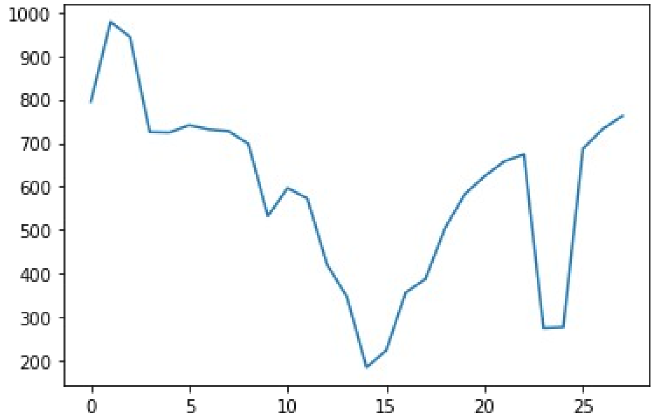
Assignment3

Part 1:

at the first, we denoise the image by median filter to dilation and erosion on the image then we set the seed point [115,5,115] and threshold=[255] manually from the image to show more clearly the segmentation:



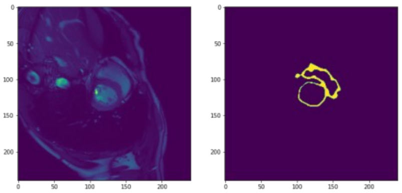
we show all results of registration functions for the left ventricle in 28 time and slice 5 of y_axis:



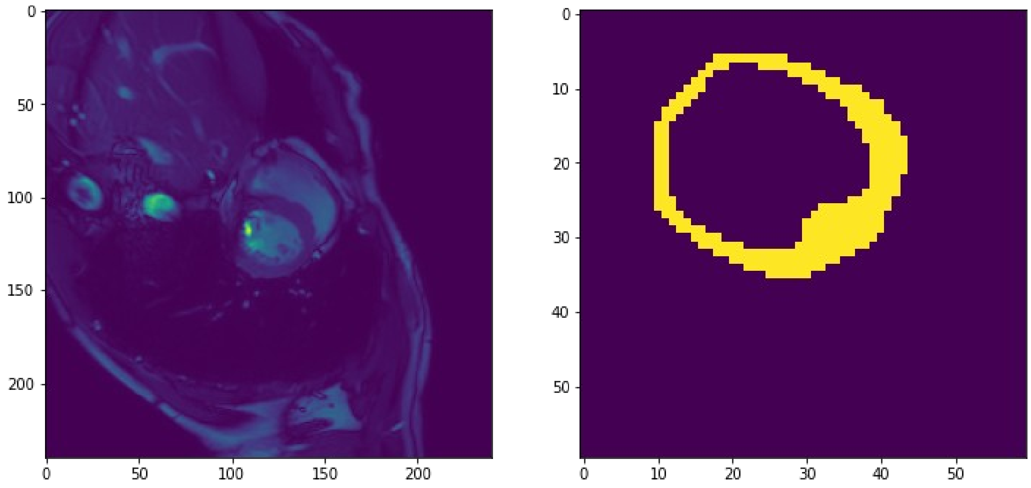
```
import nibabel as nib
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as ndimage
import cv2

# read the nifti image as 2d array
img = nib.load('/home/razieh/assig3/1401_sbtfe_bh.nii.gz')
img_data = img.get_fdata()
# print(img_data)
# plot the 4d image
```

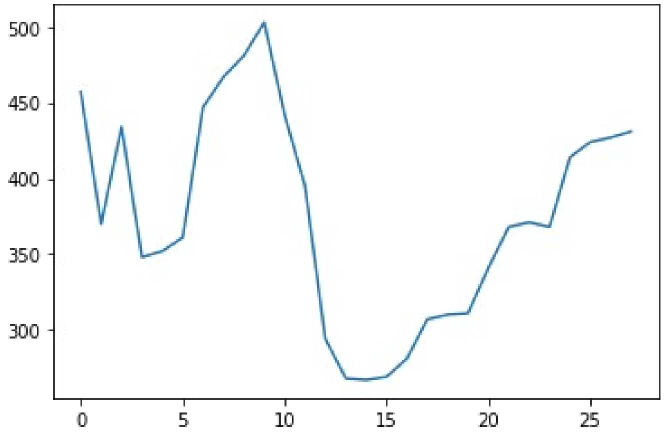
```
plt.imshow(img_data[:,5,:,12])
plt.show()



# denoise the image_data by median filter to dilation & erosion the image TO show the different part of image clearly
def img_denoise(img):
    img_data_denoise = ndimage.median_filter(img, size=4)

    # # plot the denoised image
    # plt.imshow(img_data_denoise[:,5,:,10])
    # plt.show()
    return img_data_denoise




# write function to check the homogeneity of the segmented image
def homeg(avg_old_point, new_point):
    return np.abs(avg_old_point - new_point)

# segmentation function by using region growing
def segmnetation_region_growing(img,img_segmentation , tresh,seed):
    # img_t=np.zeros(img.shape)

    #specify the seed point and set the seed point and evry point that is same as the ssed point equal to 1
    x=seed[0];y=seed[1];z=seed[2]
    img_segmentation[x,y,z]=1

    # calculate the average of pixels that specified the same as seed point to compare with threshold value
    avg=np.mean(img[np.where(img_segmentation==1)])

    # check matrix border and homegenous criterion for the 4-neighborhood
    if(x+1 < img.shape[0] and img_segmentation[x+1,y,z]==0 and homeg(avg,img[x+1,y,z])<=tresh):
    # if(x+1 < img.shape[0] and img_t[x+1,y,z]==0 and homeg(avg,img[x+1,y,z])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x+1,y,z])

    if(x-1 >= 0 and img_segmentation[x-1,y,z]==0 and homeg(avg,img[x-1,y,z])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x-1,y,z])

    if(y+1 < img.shape[1] and img_segmentation[x,y+1,z]==0 and homeg(avg,img[x,y+1,z])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x,y+1,z])

    if(y-1 >= 0 and img_segmentation[x,y-1,z]==0 and homeg(avg,img[x,y-1,z])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x,y-1,z])

    if(z+1 < img.shape[2] and img_segmentation[x,y,z+1]==0 and homeg(avg,img[x,y,z+1])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x,y,z+1])

    if(z-1 >= 0 and img_segmentation[x,y,z-1]==0 and homeg(avg,img[x,y,z-1])<=tresh):
        segmnetation_region_growing(img,img_segmentation,tresh,[x,y,z-1])

    # return img_segmentation

# create img_segmentation by size img_data.shape[0] to hold the segmented image
img_segment=np.zeros(img_data.shape[0:3])

# call the function to segment the image and save the segmented image in img_segment
# set the seed point manually from the image to show more clearly the segmentation
segmnetation_region_growing(img_denoise(img_data)[:,:,:,12],img_segment,255,[125,5,115])

# plot the segmented image
plt.figure(figsize=(12,6))
plt.subplot(121); plt.imshow(img_data[:,5,:,12])#; plt.axis('off')
plt.subplot(122); plt.imshow(img_segment[:,5,:]*255)#; plt.axis('off')





#function to calculate the time series of the segmented image in the last axis with value 28 as time in x axis and value of number of pixels in y axis
def time_series():
    thred=255
    time_series_x = np.zeros(np.shape(img_data)[3])
    for i in range(np.shape(img_data)[3]):
      print(i)
      try:
        img_segment=np.zeros(img_data.shape[0:3])
        segmnetation_region_growing(img_denoise(img_data)[:,:,:,i],img_segment,thred,[125,5,115])
      except:
        img_segment=np.zeros(img_data.shape[0:3])
        try:
          thred=thred-2
          segmnetation_region_growing(img_denoise(img_data)[:,:,:,i],img_segment,thred,[125,5,115])
        except:
          thred=thred-4
          segmnetation_region_growing(img_denoise(img_data)[:,:,:,i],img_segment,thred,[125,5,115])
      time_series_x[i] = np.sum(img_segment[:,5,:])
      # plot the img_segment
      plt.imshow(img_segment[:,5,:])
      plt.show()
      # print (time_series_x[i])


    return time_series_x

# call the function to calculate the time series of the segmented image and plot the time series
time_series_x = time_series()
plt.plot(time_series_x)
plt.show()
```

Part2)



At the first we segment entire ring so for the best result we select the part of image that contains my goal(ring) the set the seed on the ring , then do Gaussian filter to sharp the area then segment it and find series time for y_axis=5 and all 28 time, **so the main answer is:**





```
import nibabel as nib
import numpy as np
import matplotlib.pyplot as plt
import scipy.ndimage as ndimage
import cv2

# read the nifti image as 2d array
img = nib.load('/home/razieh/assig3/1401_sbtfe_bh.nii.gz')
img_data = img.get_fdata()
# print(img_data)
# plot the 4d image
# plt.imshow(img_data[100:150,5,95:145,12])
# plt.show()

# sharpen the edge of image by gaussian filter
def img_denoise(img):
    # denoise the image_data   TO sharpen the boundaries of the image
    img_data_denoise = ndimage.gaussian_filter(img, sigma=2)
    # # plot the denoised image
    plt.imshow(img_data_denoise[:,5,:,10])
    plt.show()
    return img_data_denoise


# write function to check the homogeneity of the segmented image
def homeg(avg_old_point, new_point):
    return np.abs(avg_old_point - new_point)


# segmentation function by using region growing
def segmnetation_region_growing(img, img_segmentation, tresh, seed):
    # img_t=np.zeros(img.shape)
    # specify the seed point and set the seed point and evry point that is
same as the ssed point equal to 1
    x = seed[0];
    y = seed[1];
    z = seed[2]
    img_segmentation[x, y, z] = 1

    # calculate the average of pixels that specified the same as seed point to
compare with threshold value
    avg = np.mean(img[np.where(img_segmentation == 1)])

    # check matrix border and homegenous criterion for the 4-neighborhood
    if (x + 1 < img.shape[0] and img_segmentation[x + 1, y, z] == 0 and
homeg(avg, img[x + 1, y, z]) <= tresh):
        # if(x+1 < img.shape[0] and img_t[x+1,y,z]==0 and
homeg(avg,img[x+1,y,z])<=tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x + 1, y,
z])

    if (x - 1 >= 0 and img_segmentation[x - 1, y, z] == 0 and homeg(avg, img[x
- 1, y, z]) <= tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x - 1, y,
z])

    if (y + 1 < img.shape[1] and img_segmentation[x, y + 1, z] == 0 and
homeg(avg, img[x, y + 1, z]) <= tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x, y + 1,
z])

    if (y - 1 >= 0 and img_segmentation[x, y - 1, z] == 0 and homeg(avg,
img[x, y - 1, z]) <= tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x, y - 1,
z])

    if (z + 1 < img.shape[2] and img_segmentation[x, y, z + 1] == 0 and
homeg(avg, img[x, y, z + 1]) <= tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x, y, z +
1])

    if (z - 1 >= 0 and img_segmentation[x, y, z - 1] == 0 and homeg(avg,
img[x, y, z - 1]) <= tresh):
        segmnetation_region_growing(img, img_segmentation, tresh, [x, y, z -
1])

    # return img_segmentation


# create img_segmentation by size img_data.shape[0] to hold the segmented
image
print(img_data.shape)
img_segment = np.zeros([60, 7, 60])

# call the function to segment the image and save the segmented image in
```

```
img_segment
# set the seed point manually from the image to show more clearly the
segmentation

# select the part of image that contains my goal(ring) the set the seed on the
ring , then do gaussian filter to sharp the area then segment it
segmnetation_region_growing(img_denoise(img_data)[100:145, :, 95:140, 12],
img_segment, 150, [20, 5, 40])

# plot the segmented image
plt.figure(figsize=(12, 6))
plt.subplot(121);
plt.imshow(img_data[:, 5, :, 12])  # ; plt.axis('off')
plt.subplot(122);
plt.imshow(img_segment[:, 5, :] * 255)  # ; plt.axis('off')



# function to calculate the time series of the segmented image in the last
axis with value 28 as time in x axis and value of number of pixels in y axis
def time_series():
    thred = 150
    time_series_x = np.zeros(np.shape(img_data)[3])
    for i in range(np.shape(img_data)[3]):
        print(i)
        try:
            img_segment = np.zeros([60, 7, 60])
            segmnetation_region_growing(img_denoise(img_data)[100:145, :,
95:140, i], img_segment, thred, [20, 5, 40])
        except:
            img_segment = np.zeros([60, 7, 60])
            try:
                thred = thred - 2
                segmnetation_region_growing(img_denoise(img_data)[100:145, :,
95:140, i], img_segment, thred,
                                            [20, 5, 40])
            except:
                thred = thred - 4
                segmnetation_region_growing(img_denoise(img_data)[100:145, :,
95:140, i], img_segment, thred,
                                            [20, 5, 40])
        time_series_x[i] = np.sum(img_segment[:, 5, :])
        # plot the img_segment
        plt.imshow(img_segment[:, 5, :])
        plt.show()
        # print (time_series_x[i])

    return time_series_x


# call the function to calculate the time series of the segmented image and
plot the time series
time_series_x = time_series()
```

```
plt.plot(time_series_x)
plt.show()
```