Preprocessing and Segmentation Results Marcio

May 10, 2018

In [1]:
```
import cv2
import matplotlib.pyplot as plt
from immas import MammogramImage , preprocessing , segmentation, classification
from immas.io import read_dataset
from immas.basic_functions import show_image_plt
from immas import preprocessing

#Openkernel , Openkernel1 and Openkernel2 are tuples
def Segmentation(img, OpenKernel=(10,10), CloseKernel=(10,10)):
    img = segmentation.multithresholding(img)
    img = segmentation.thresh_to_binary(img)
    img1 = preprocessing.open(img,OpenKernel)
    img2 = preprocessing.close(img1,CloseKernel)
    return (img1,img2)

#mean 1&2 are numbers
def Segmentation2(img, mean1=30,mean2=30, OpenKerne2=(20,20)):
    img = segmentation.mean_shift(img,mean1,mean2)
    img = segmentation.multithresholding(img)
    img = segmentation.thresh_to_binary(img)
    img = preprocessing.open(img,OpenKerne2)
    return img

from immas import preprocessing

def testPreProcessing (img, MorphoKernel = 20, ClaheKernel = 10.0, OpenKernel=(5,5)):
    #img = preprocessing.resize(img, 0.5, 0.5)
    #img = preprocessing.close(img)
    #img = preprocessing.dilate(img)
    #img = preprocessing.clahe(img,10.0)
    img = preprocessing.open(img,OpenKernel)
    img = preprocessing.morphoEnhancement(img,MorphoKernel, ClaheKernel)
    img = preprocessing.waveletTransform(img)
    #img = preprocessing.erode(img,(3,3))
    return img
```

In [2]:
```
import cv2
import numpy as np
```

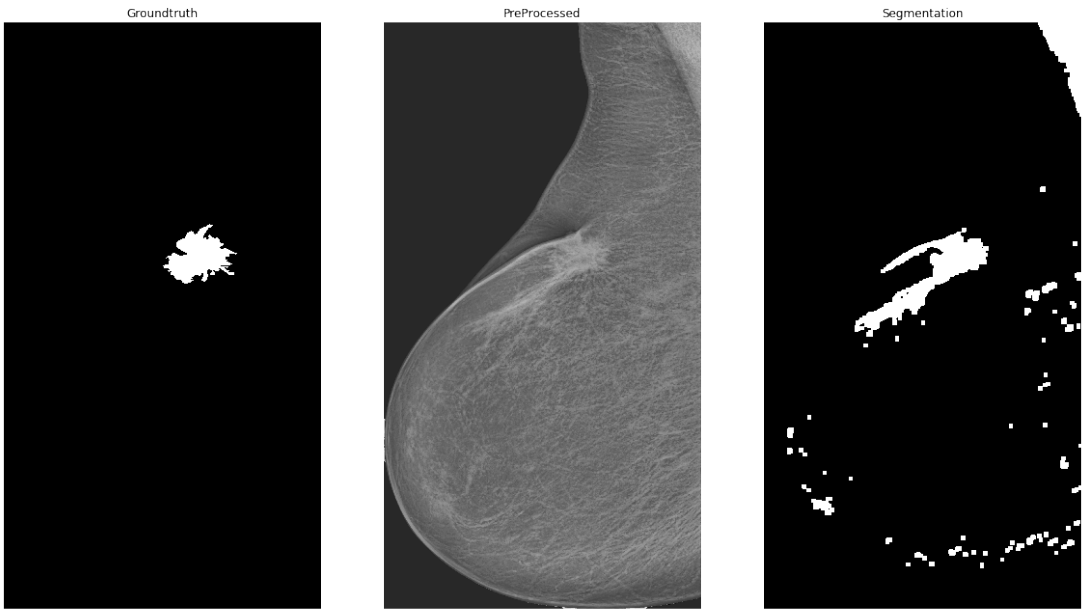
```
import matplotlib.pyplot as plt
from immas import MammogramImage , preprocessing , segmentation
from immas.io import read_dataset
from immas.basic_functions import show_image_plt
from immas import preprocessing

#def Plot_Result(img, Morpho_kernel_size, Morpho_clahe_kernel, PreOpenKernel, OpenKern
def Plot_Result(img):

    jaccard = np.zeros(len(img))
    dice = list()
    for index, m in enumerate(img, start=0):
        m.read_data()
        #m.image_data = testPreProcessing(m.image_data, Morpho_kernel_size, Morpho_cla
        m.image_data = preprocessing.fullPreprocessing(m.image_data)
        preprocessed = m.image_data

        #segment using both methods
        #result1, result2 = Segmentation(m.image_data, OpenKernel, CloseKernel)
        m.image_data = segmentation.fullSegmentation(m.image_data)
        #result3 = Segmentation2(m.image_data, mean1,mean2, Openkernel2)

        #show both results
        plt.figure(figsize=(20, 40))
        plt.subplot(1,3,1)
        plt.axis("off")
        plt.imshow(m.cropped_ground_truth, interpolation="nearest", cmap=plt.cm.gray)
        plt.title('Groundtruth')

        plt.subplot(1,3,2)
        plt.axis("off")
        plt.imshow(preprocessed, interpolation="nearest", cmap=plt.cm.gray)
        plt.title('PreProcessed')

        plt.subplot(1,3,3)
        plt.axis("off")
        plt.imshow(m.image_data, interpolation="nearest", cmap=plt.cm.gray)
        plt.title('Segmentation')

        #plt.subplot(1,4,4)
        #plt.axis("off")
        #plt.imshow(result2, interpolation="nearest", cmap=plt.cm.gray)
        #plt.title('Segmentation2')
        plt.show()
        jaccard[index] = segmentation.jaccard_index(m.image_data,  m.cropped_ground_tru
        dice.append(classification.find_match(m, visual_result = "yes"))
        print(jaccard[index], dice[index])
        #jac2 = segmentation.jaccard_index(result2,  m.cropped_ground_truth)
```

```
            #av_jaccard2 = av_jaccard2 + jac2
        return jaccard, dice
```

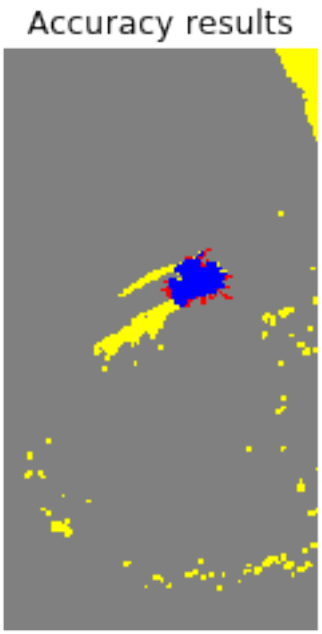
```
In [3]: data_set = read_dataset(image_folder="../dataset/masses_examples",
                    mask_folder="../dataset/masks",
                    results_folder="../dataset/groundtruth",
                    train_set_fraction=1)

        print("Number of images for training is {0}, number of images for testing is {1}".forma
            len(data_set["train"]), len(data_set["test"])))

        img = data_set["train"]

        jaccard, dice = Plot_Result(img)
```

```
Reading list of files...
Reading mamograms images and all additional data...
All data have been successfully loaded.
Number of images for training is 107, number of images for testing is 0
```

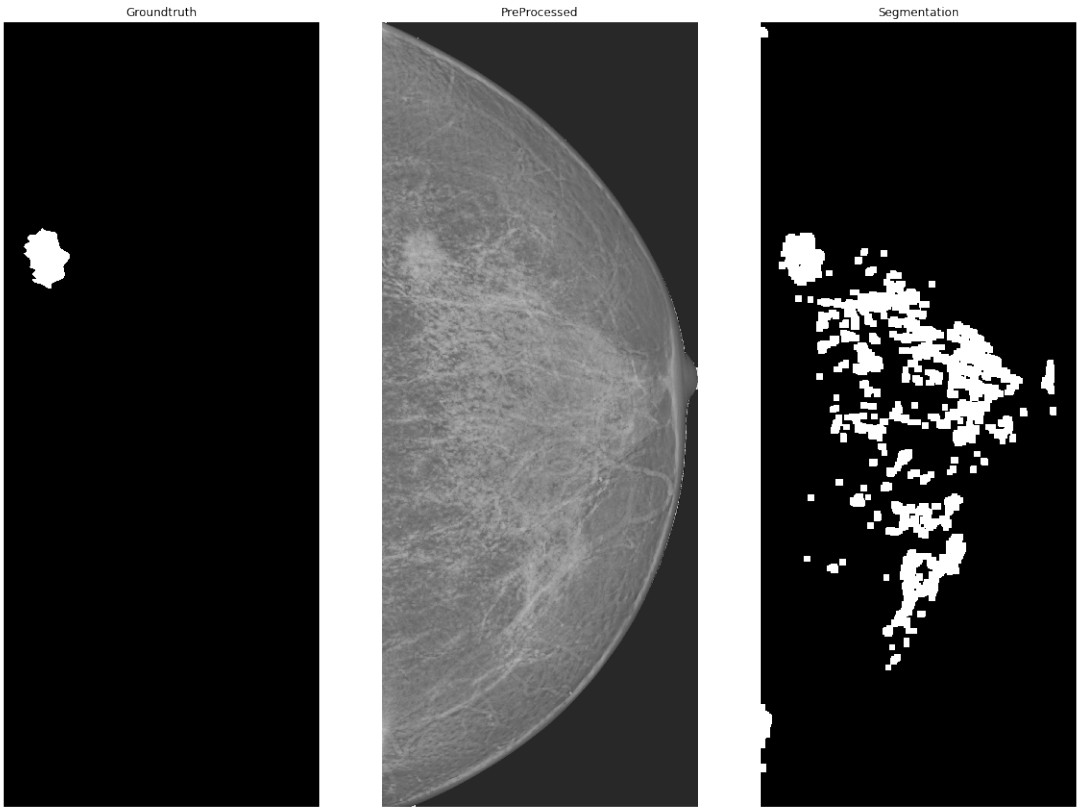
```
C:\ProgramData\Anaconda3\lib\site-packages\immas-1.0-py3.6.egg\immas\segmentation.py:51: Runtin
```

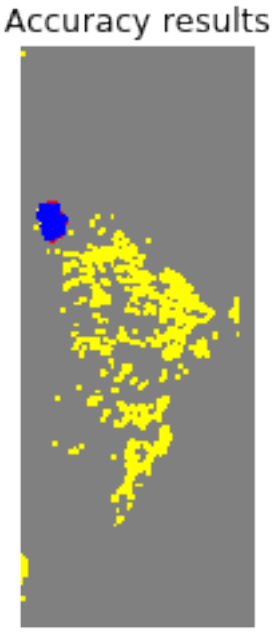



```
C:\ProgramData\Anaconda3\lib\site-packages\immas-1.0-py3.6.egg\immas\classification.py:18: Runt
C:\ProgramData\Anaconda3\lib\site-packages\immas-1.0-py3.6.egg\immas\basic_functions.py:80: Ru
```

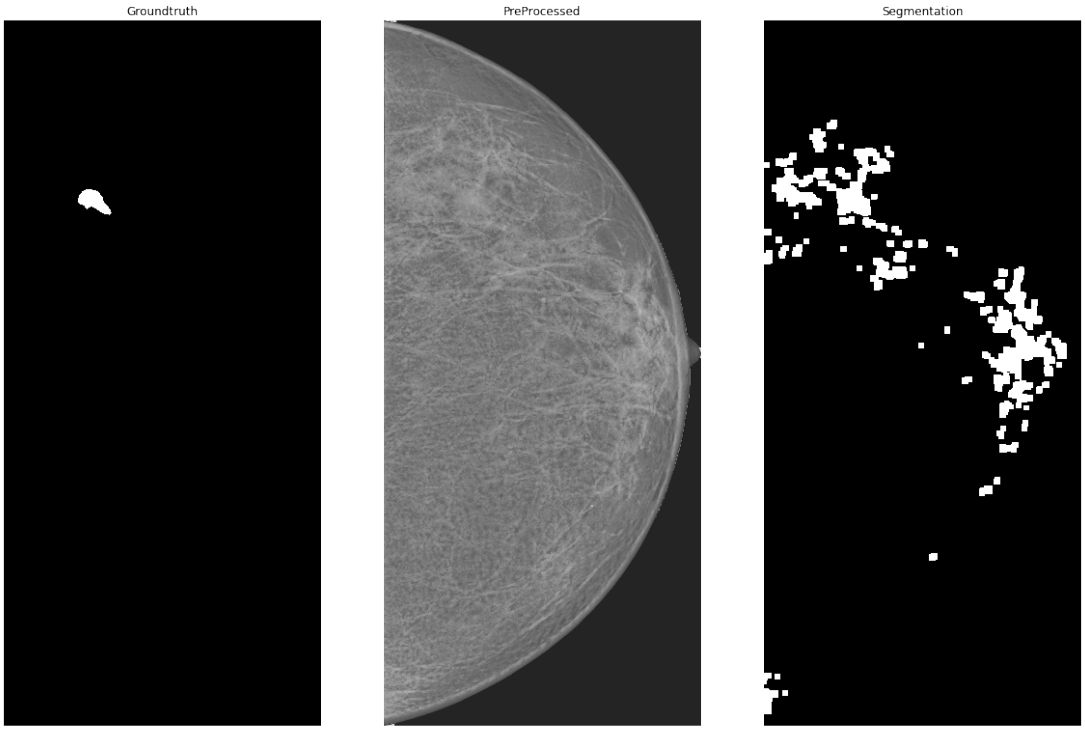


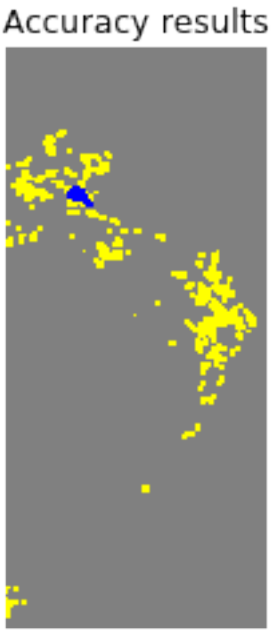
0.41879528048023185 (1, 52, 1)



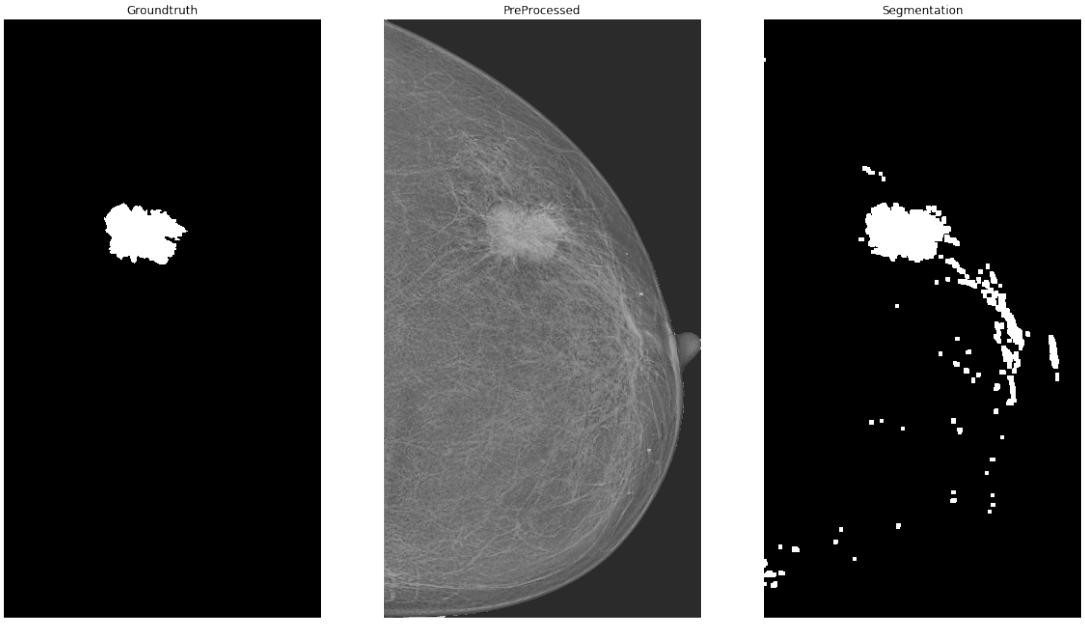


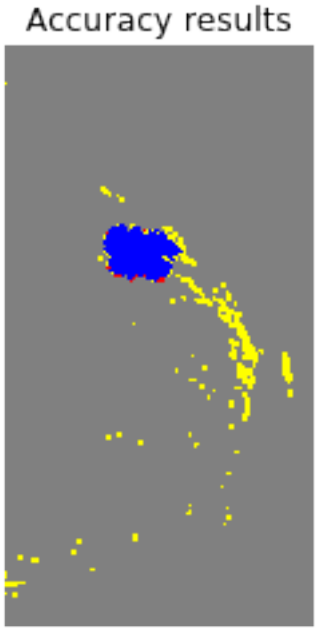

```
0.8572313748359819 (1, 62, 1)
```


Groundtruth
PreProcessed
Segmentation

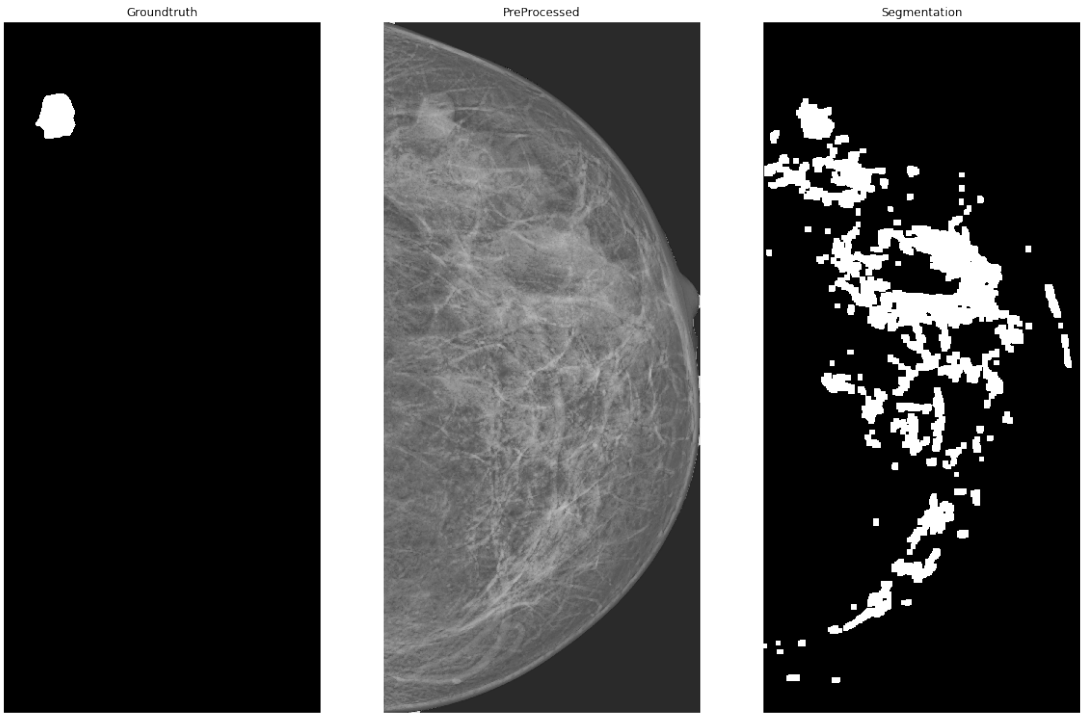

Accuracy results

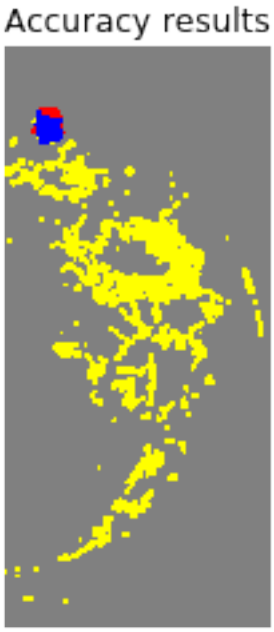
```
0.2665249451936491 (1, 51, 1)
```



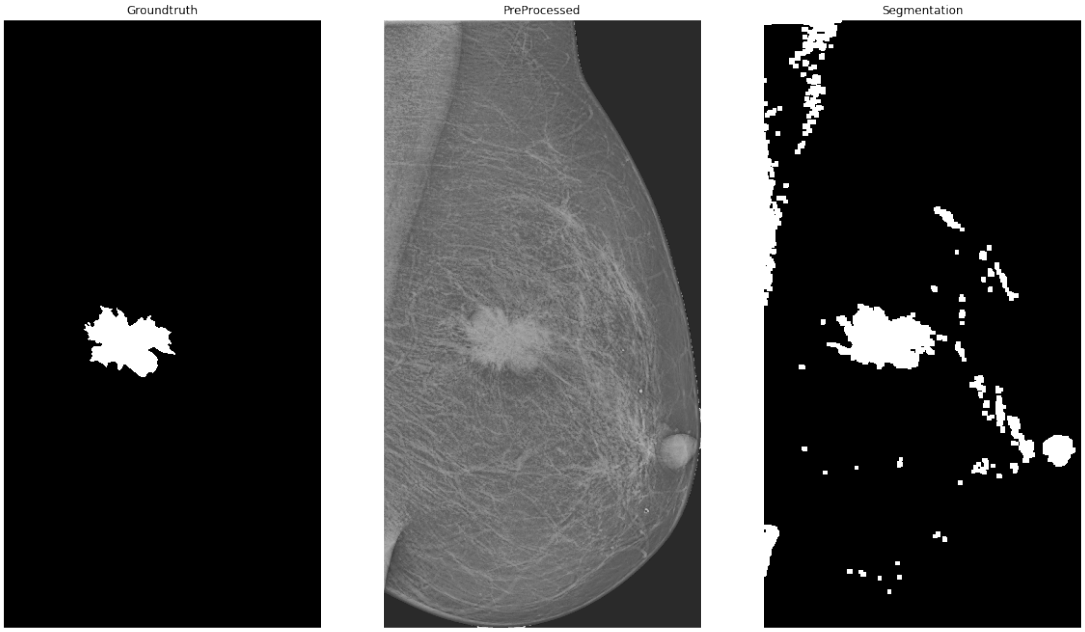


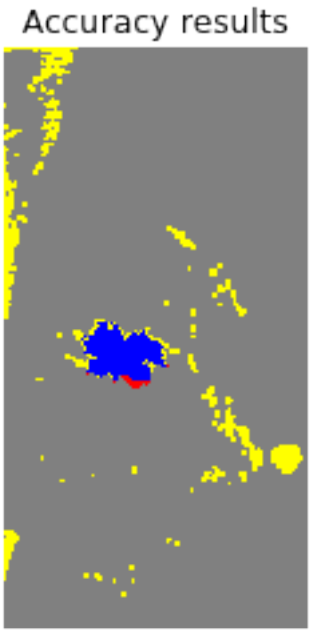
```
0.8512160963198147 (1, 54, 1)
```



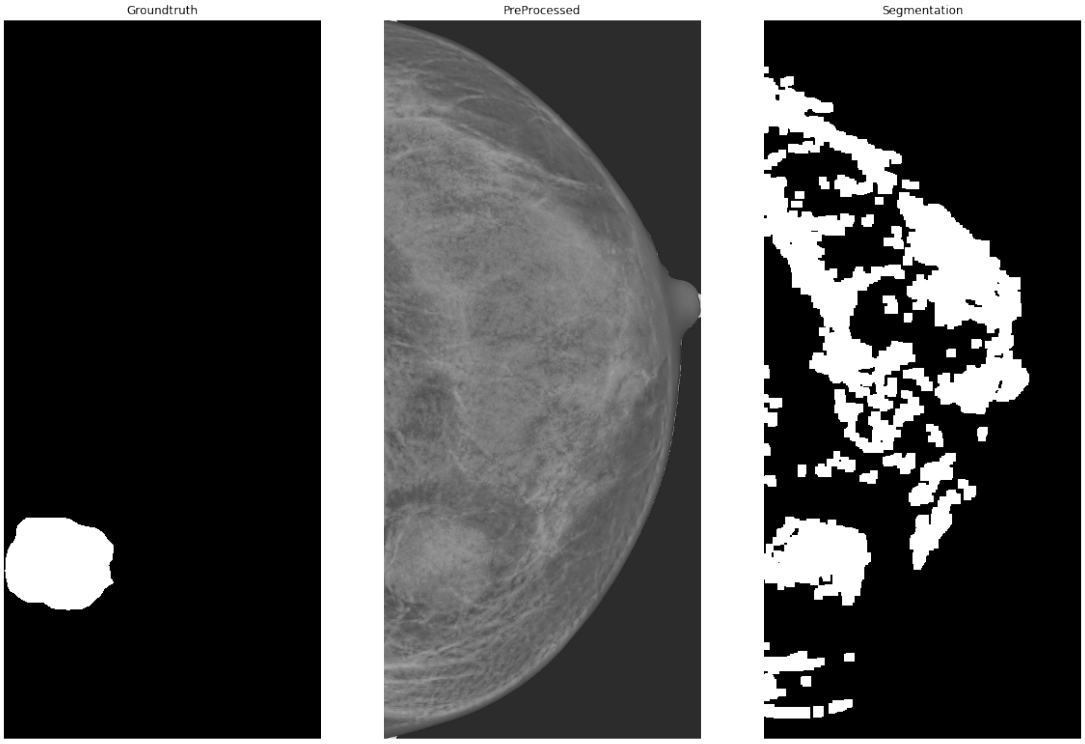


```
0.6945900572172543 (1, 62, 1)
```

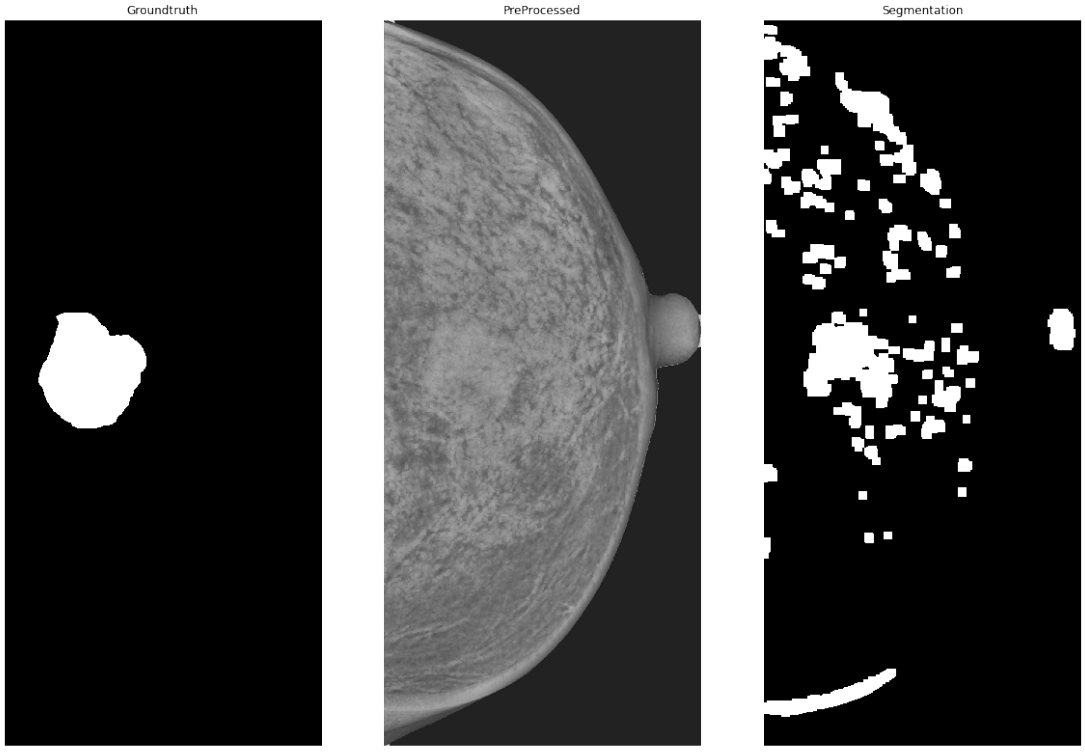


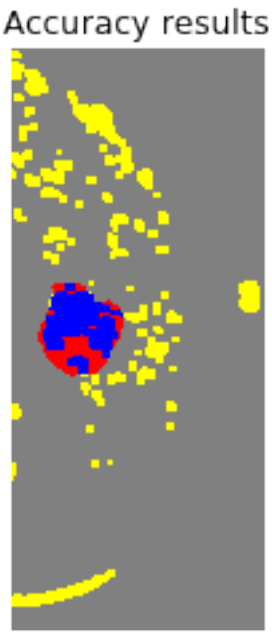


```
0.7623581054632892 (1, 59, 1)
```

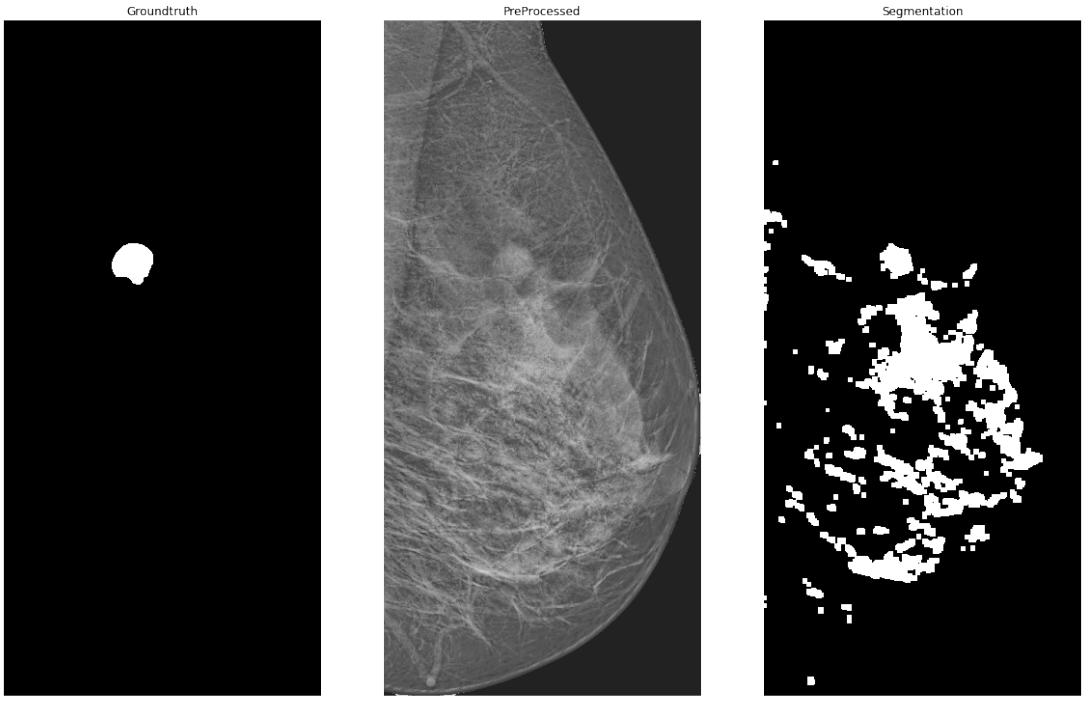


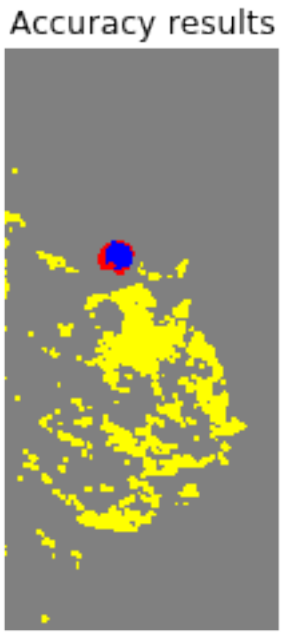
```
0.7982593281055729 (1, 38, 1)
```



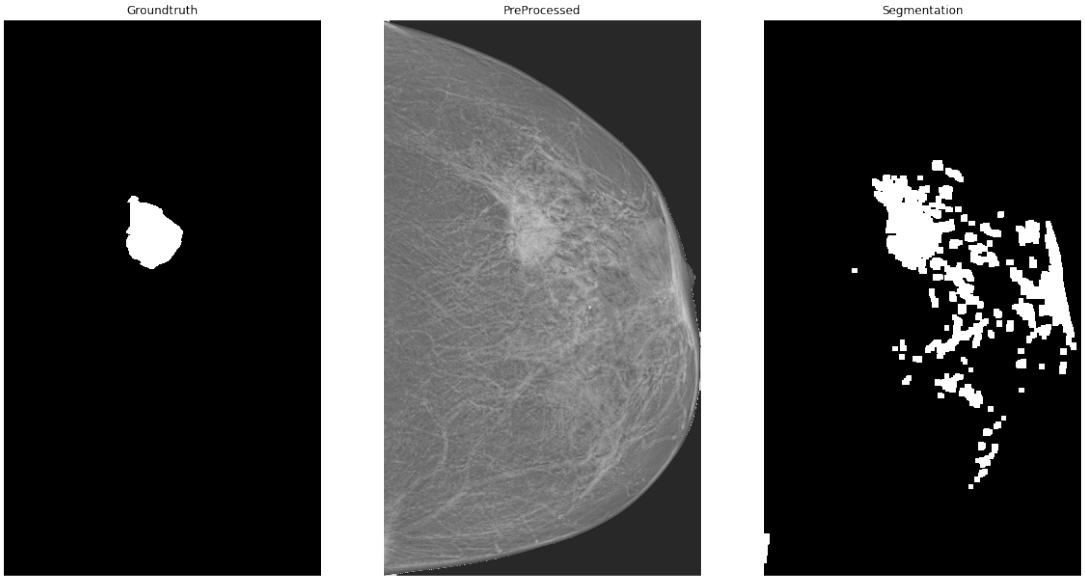


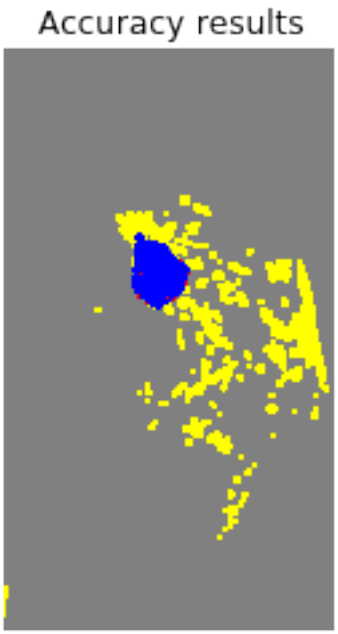

```
0.6183948893683632 (1, 54, 1)
```



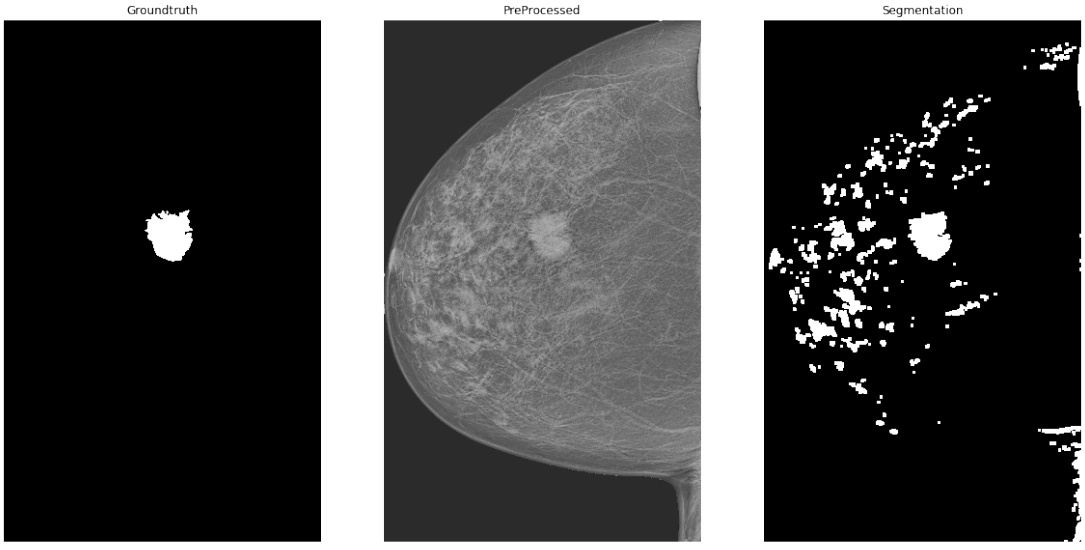


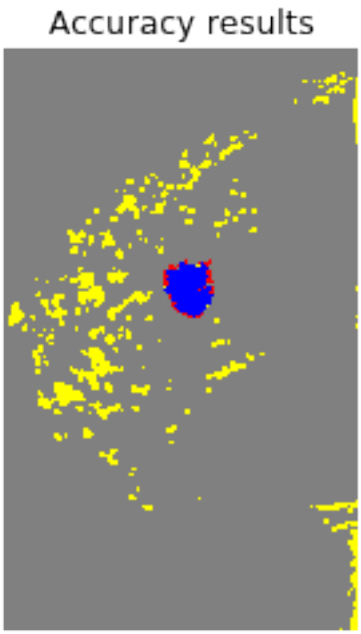

```
0.6259837312347067 (1, 77, 1)
```



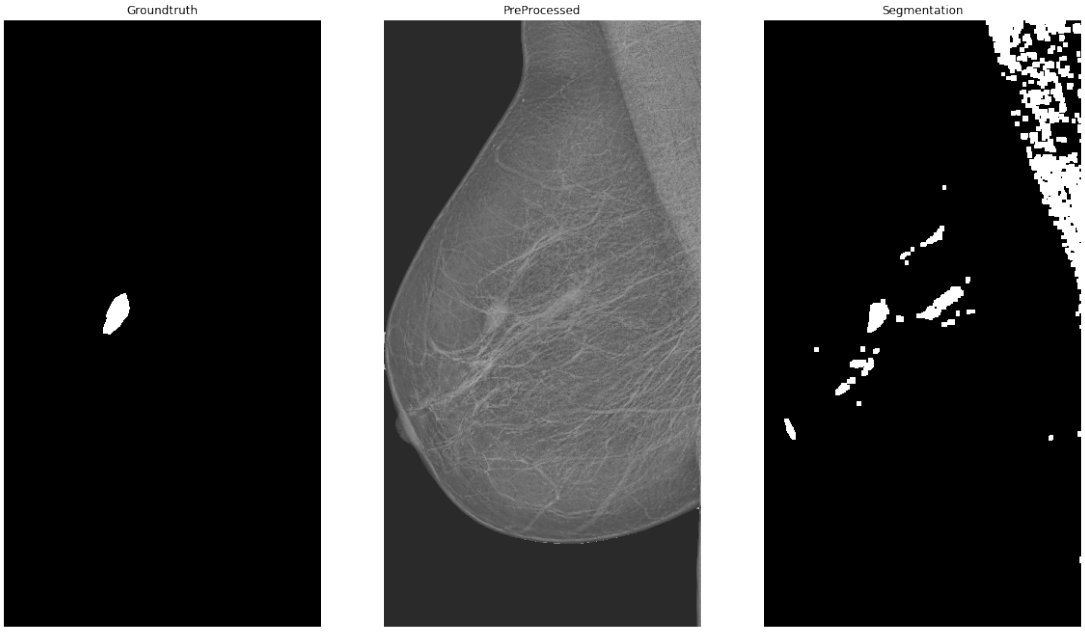


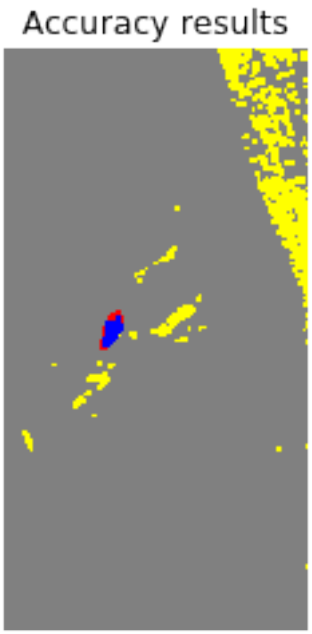

```
0.5535222579164755 (1, 50, 1)
```



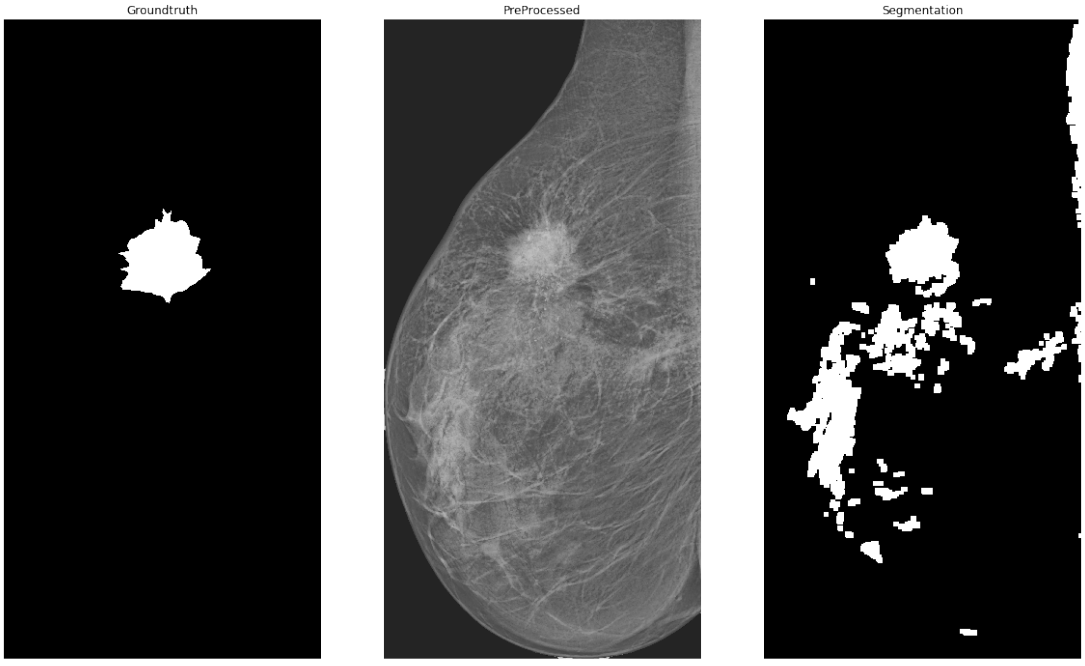


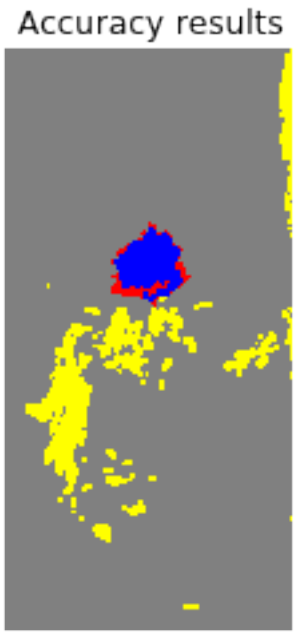

```
0.8341466570546591 (1, 149, 1)
```



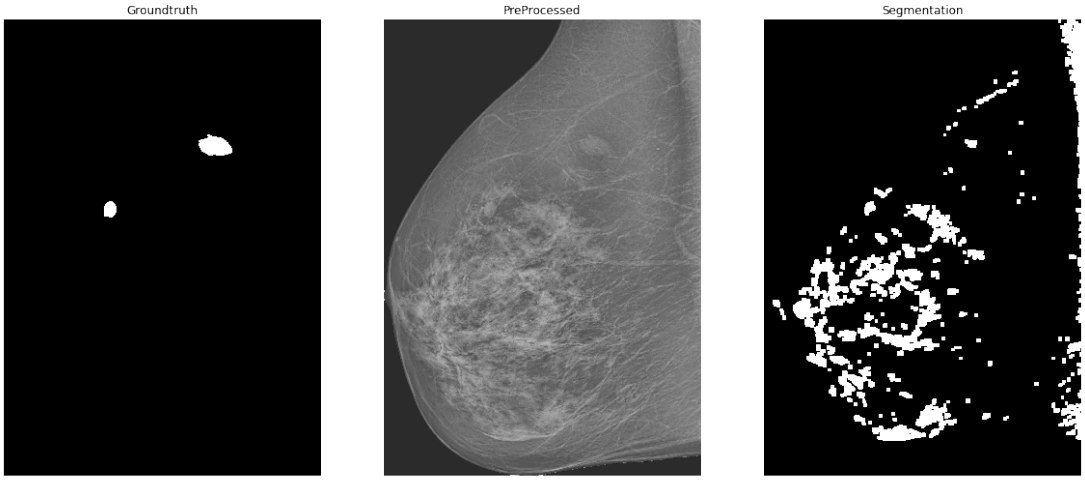


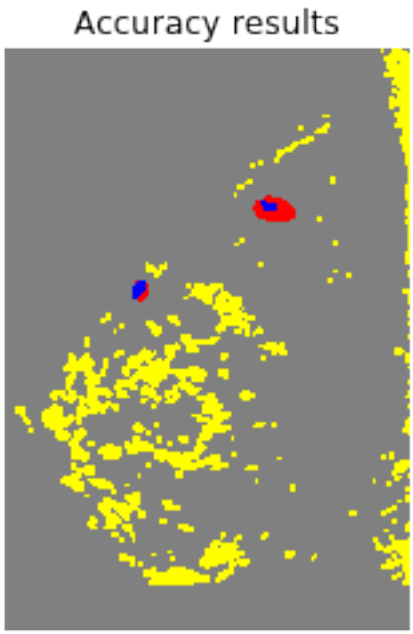

```
0.6812050359712231 (1, 52, 1)
```



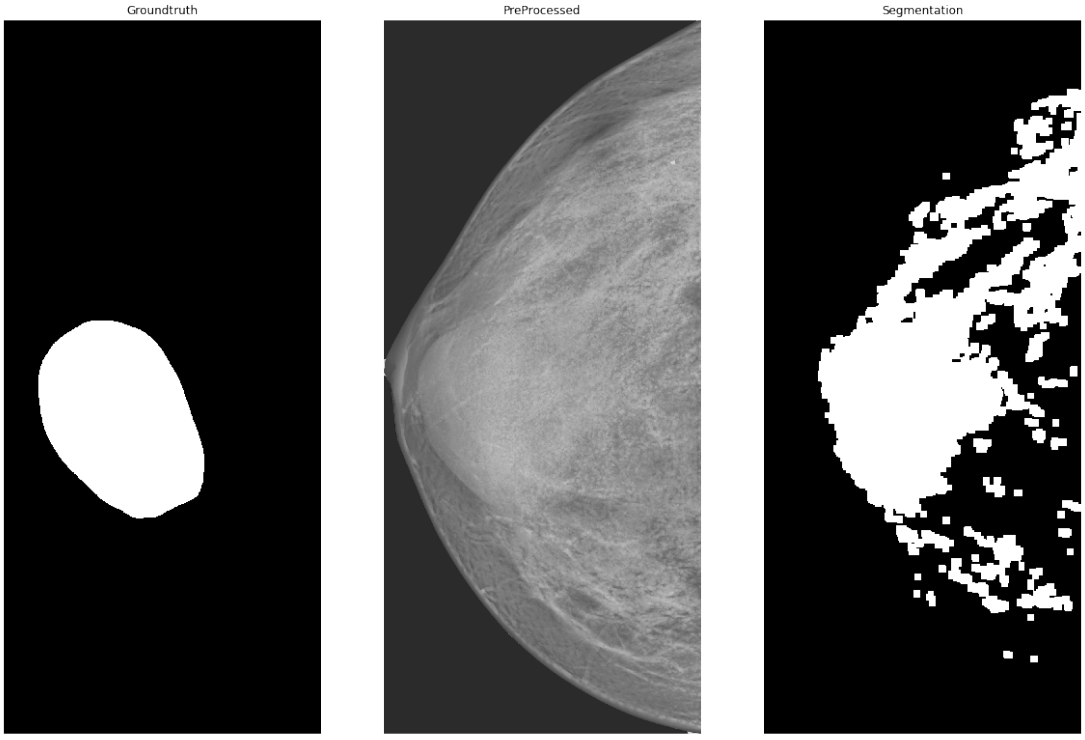


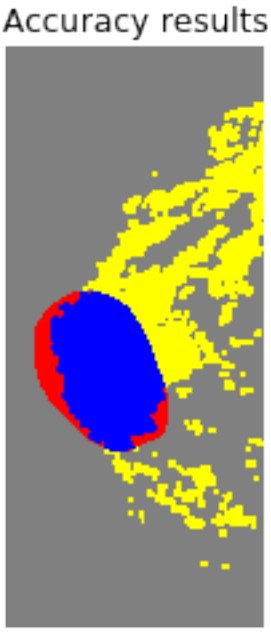

```
0.7676494944963942 (1, 34, 1)
```



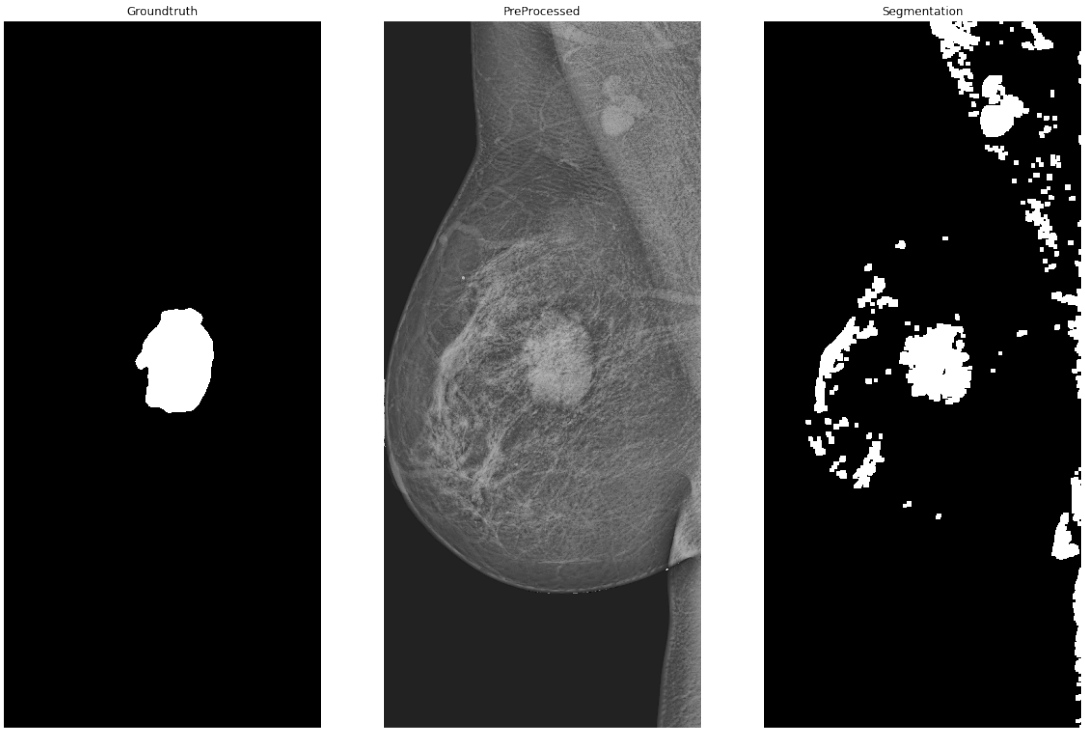


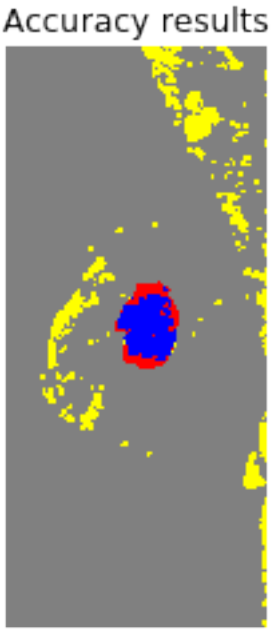
0.6385598686525156 (2, 124, 2)



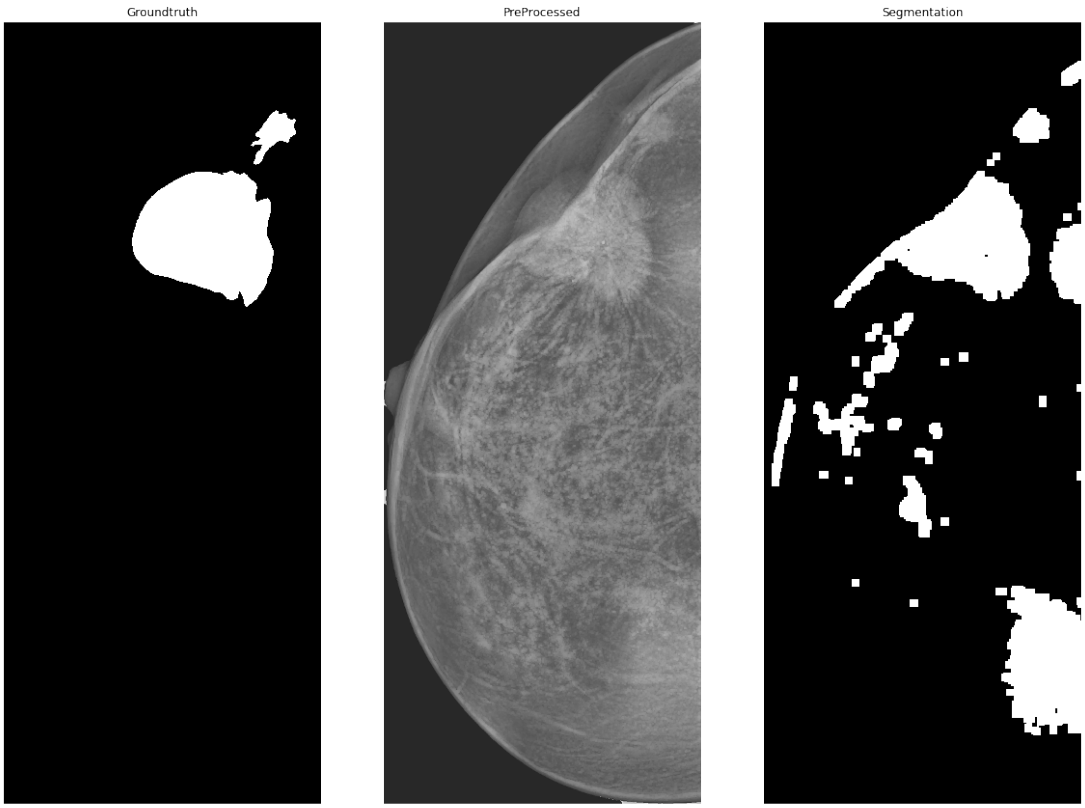


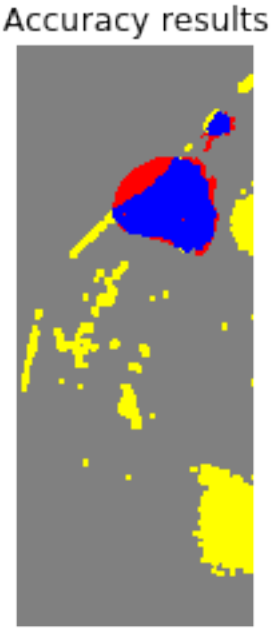

```
0.33930989906072107 (1, 34, 1)
```


Groundtruth
PreProcessed
Segmentation

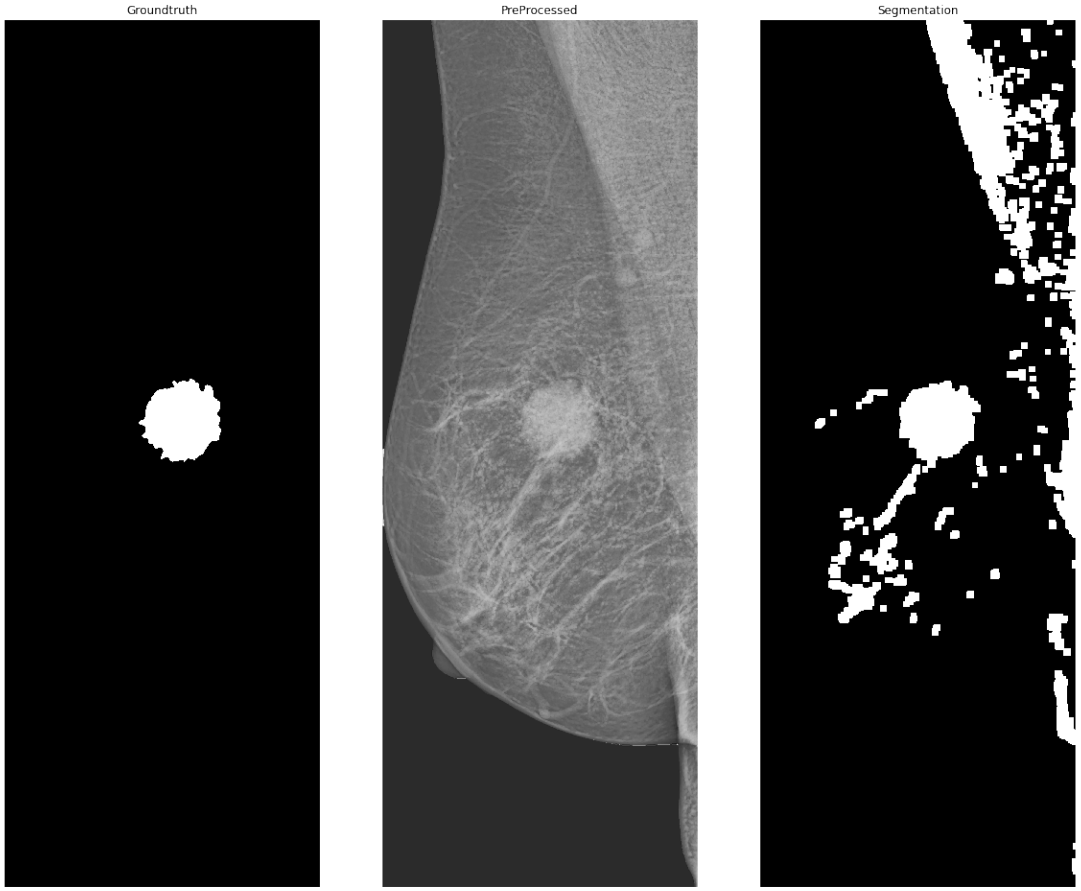

Accuracy results

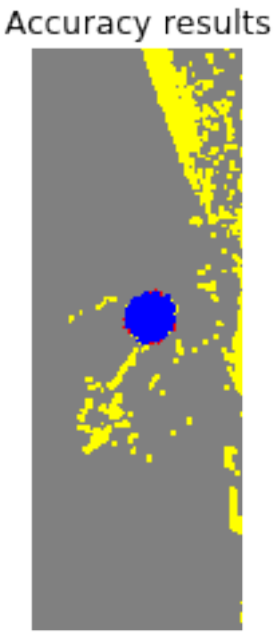
```
0.6768911242899682 (1, 91, 1)
```



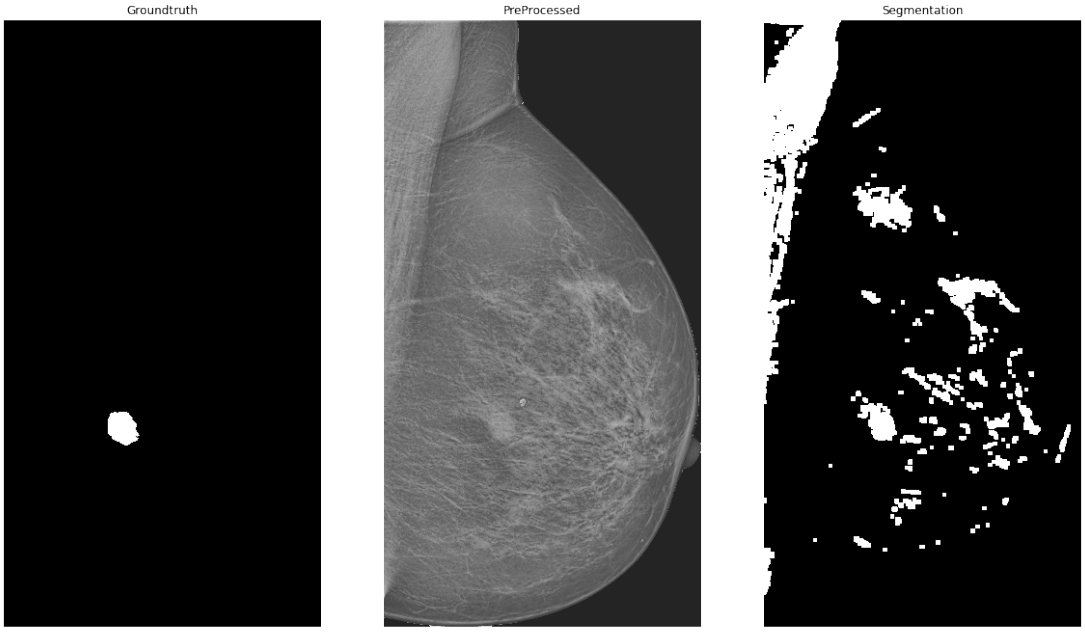


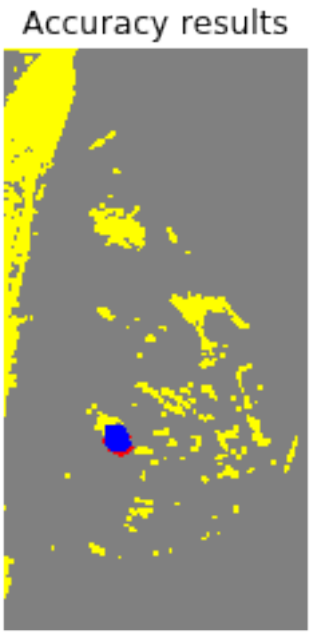
0.7097588194735682 (2, 30, 2)



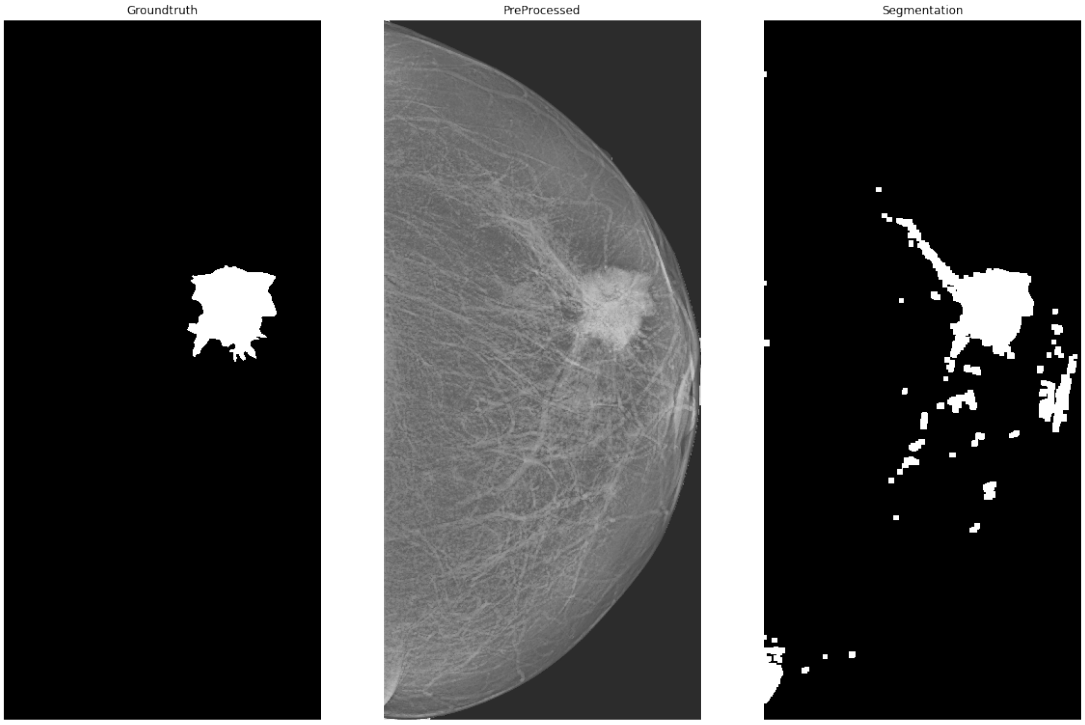


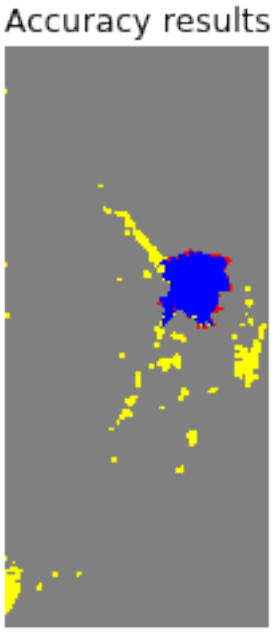

```
0.9112034534231653 (1, 82, 1)
```



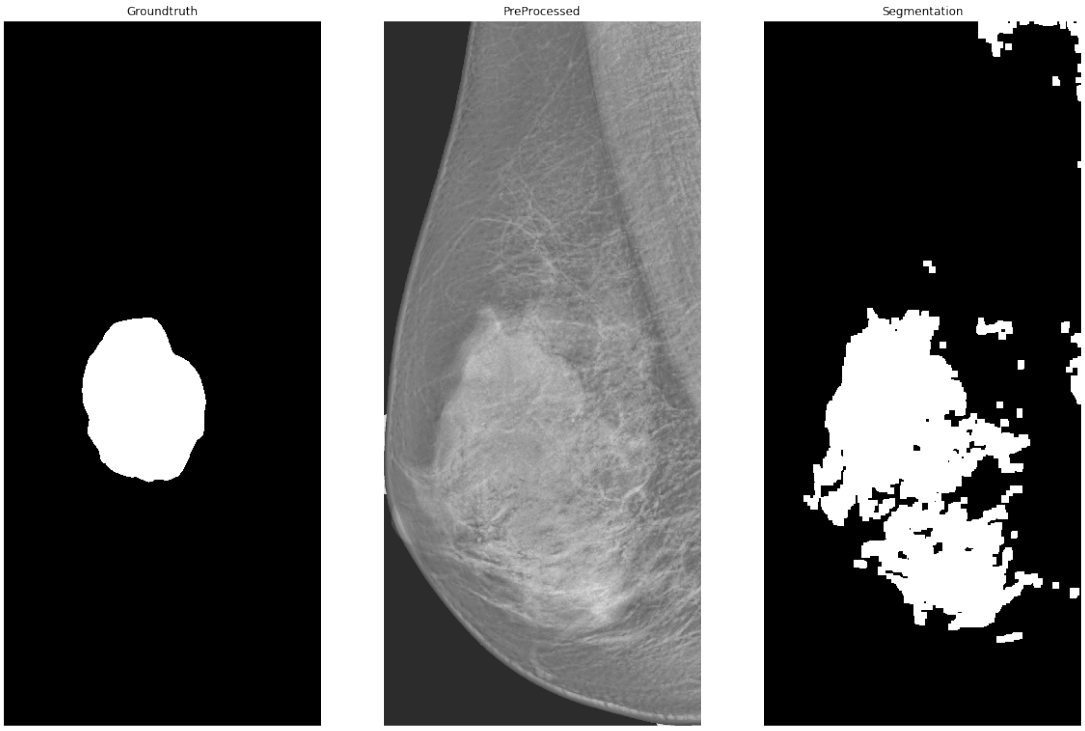


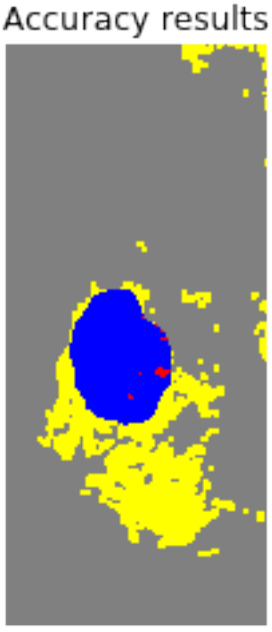
0.5832329014763483 (1, 88, 1)



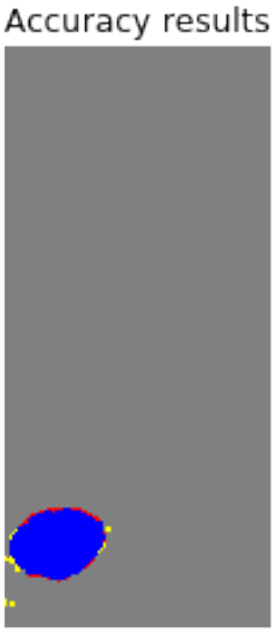



```
0.7811227320082804 (1, 37, 1)
```

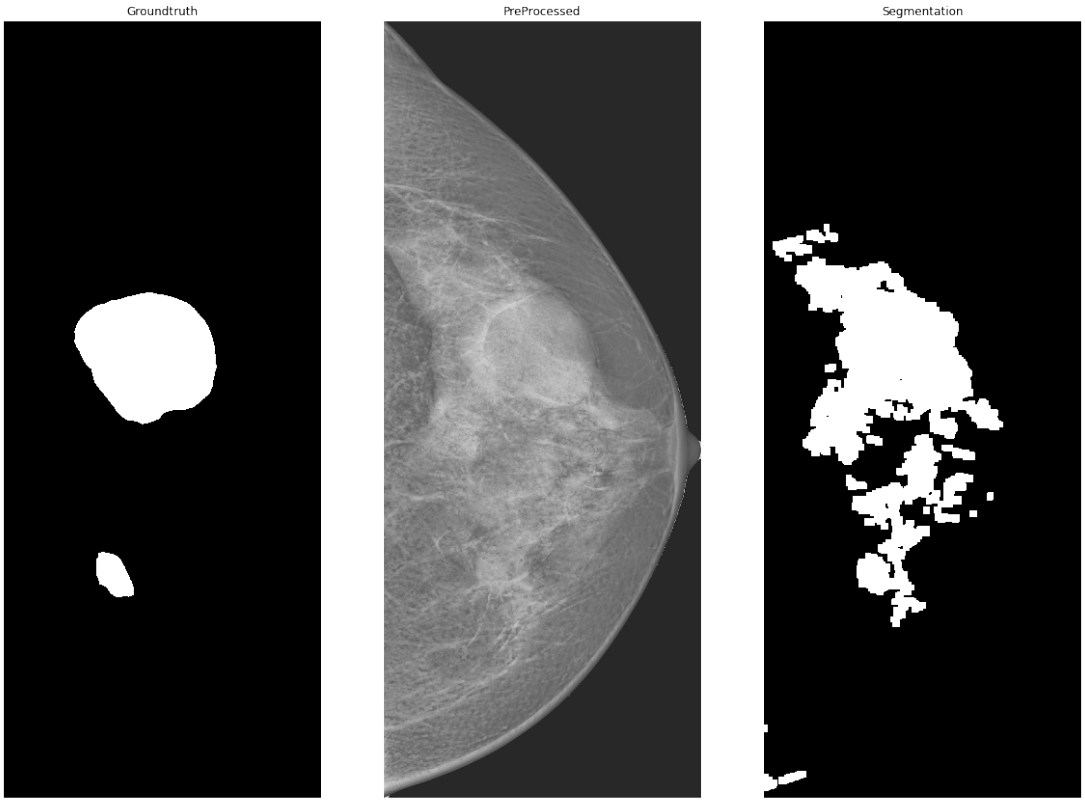

Groundtruth
PreProcessed
Segmentation

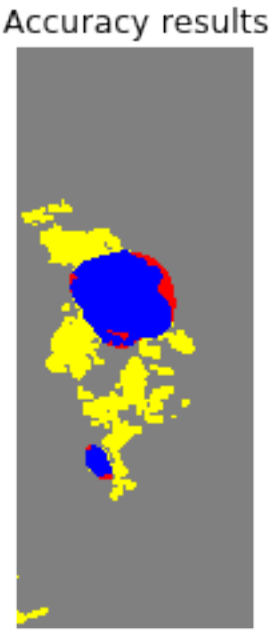

Accuracy results

```
0.37921643858025034 (1, 26, 1)
```

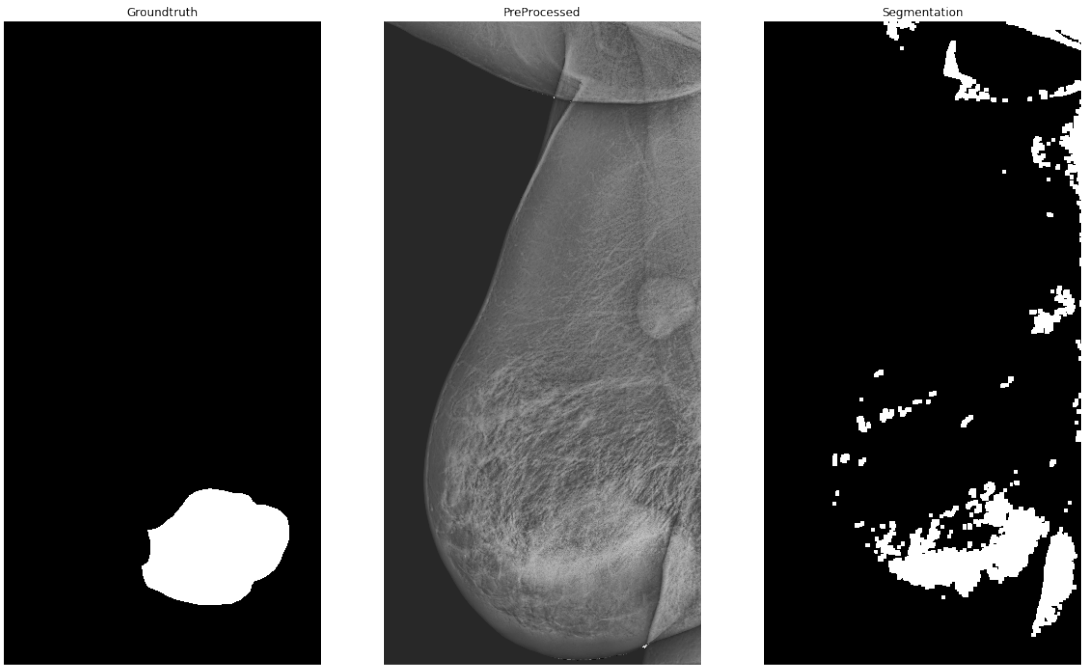


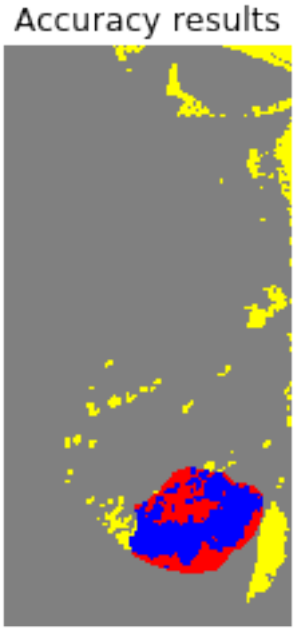
0.9092516205067767 (1, 3, 1)



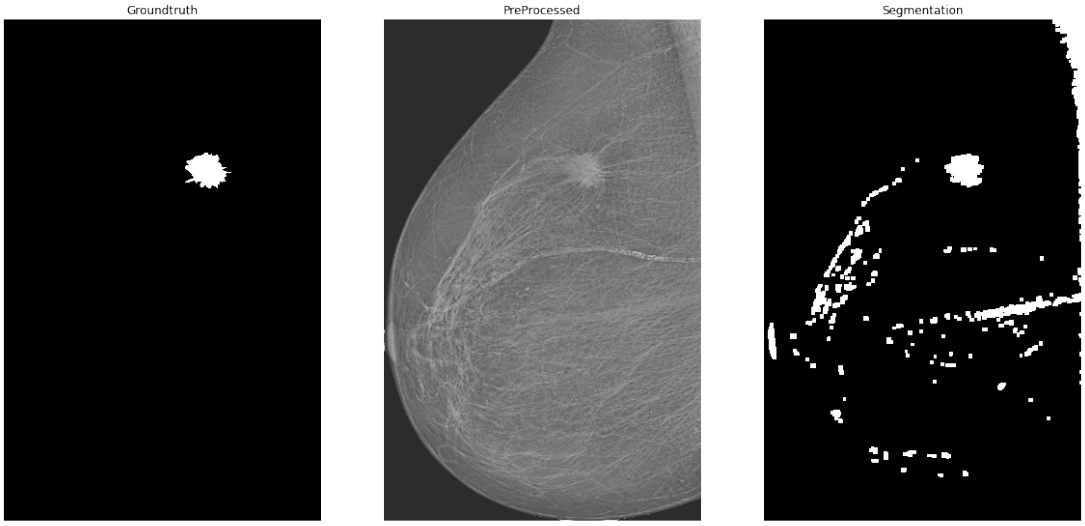


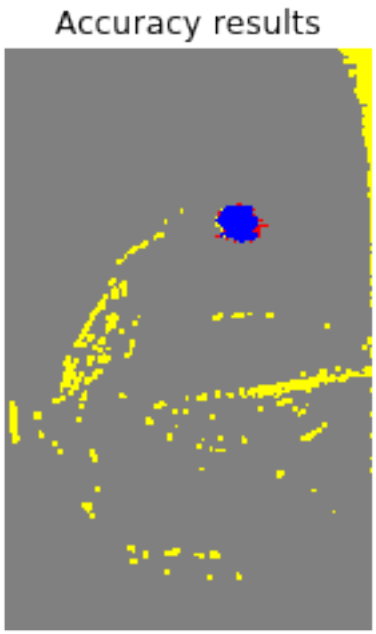
```
0.534633244060808 (2, 12, 2)
```



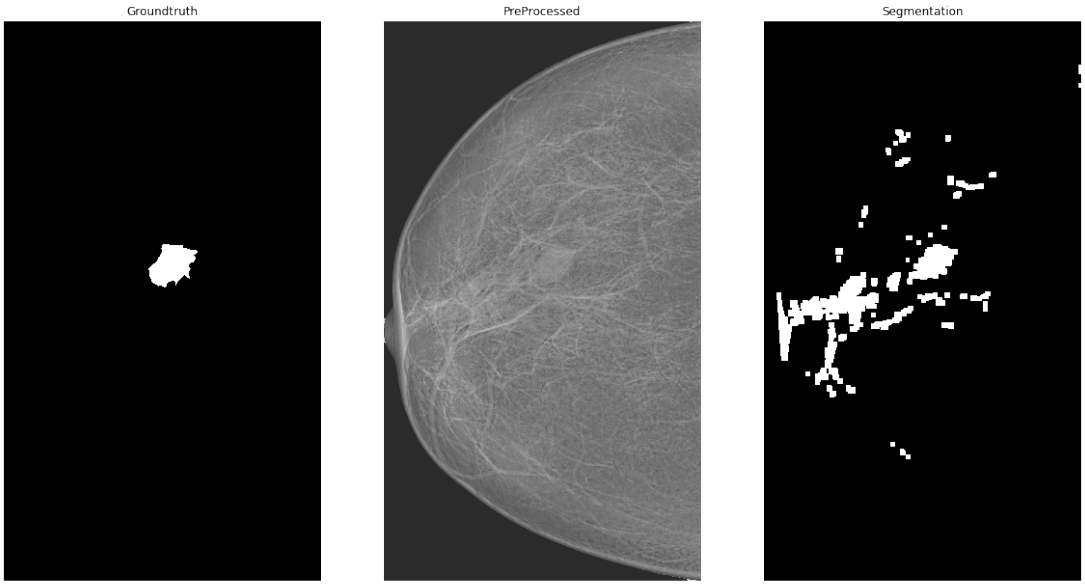


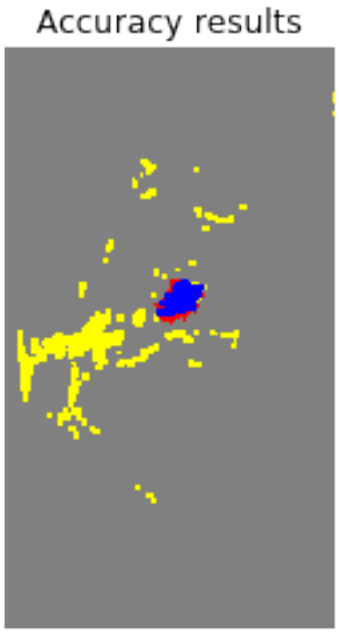
```
0.5522073085268794 (1, 76, 1)
```



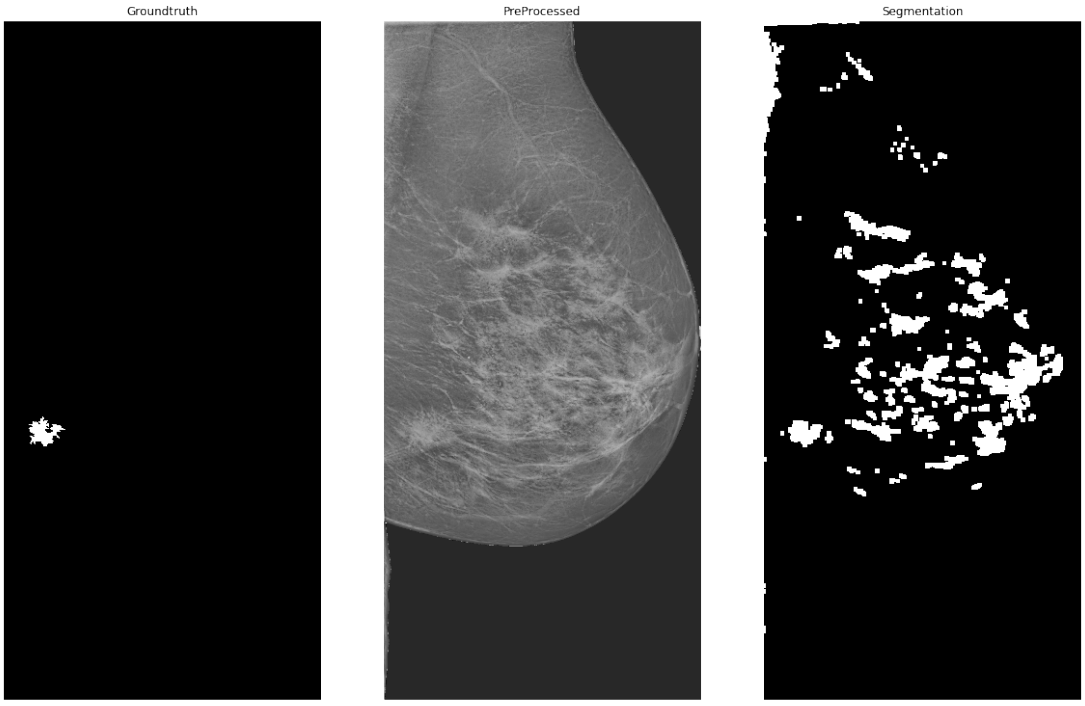


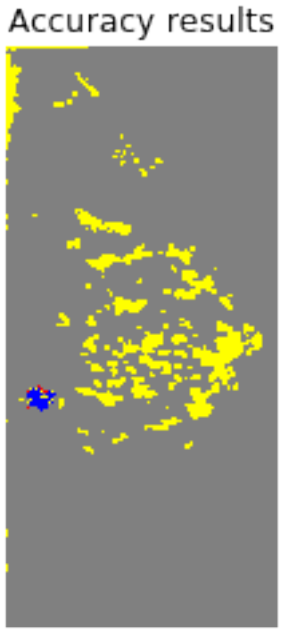
0.8273920368892436 (1, 98, 1)



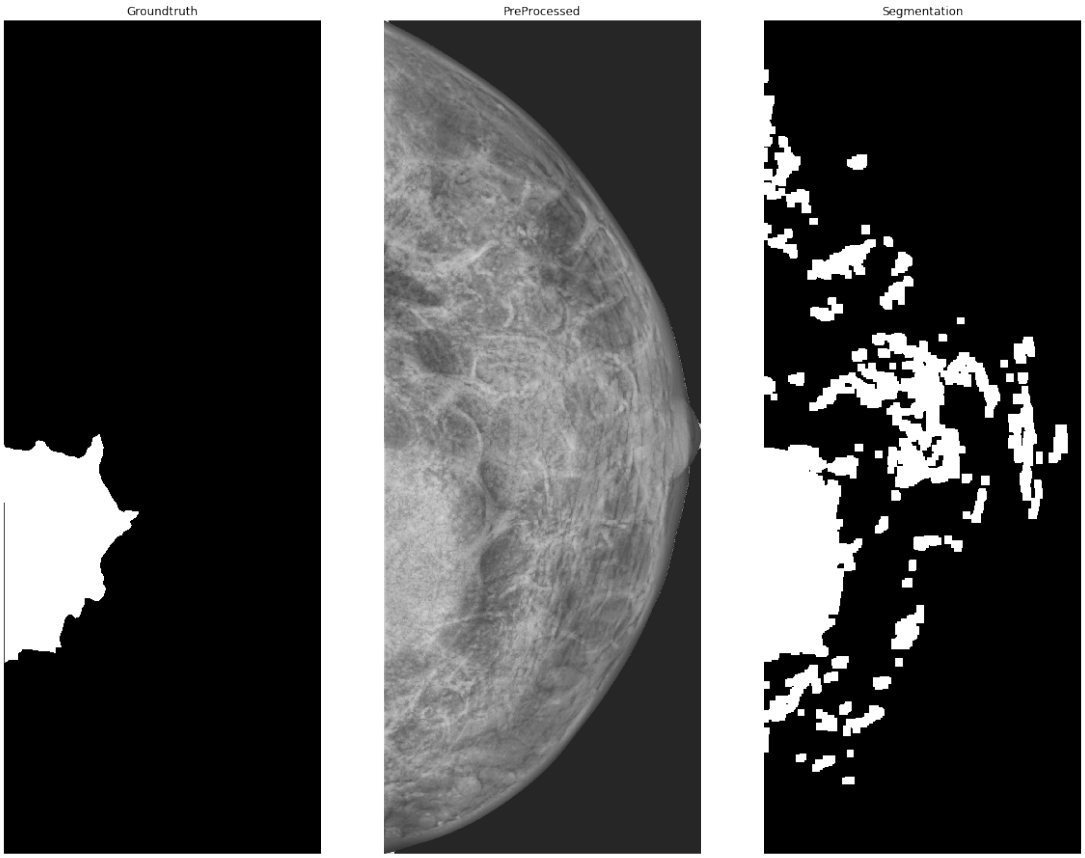


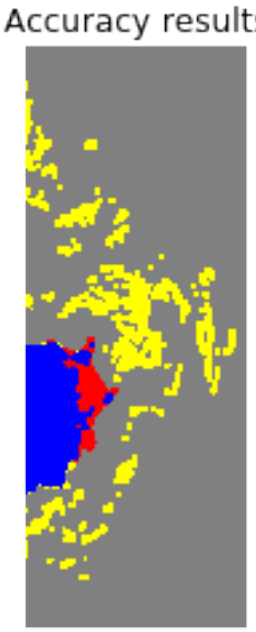
0.7240802675585284 (1, 38, 1)



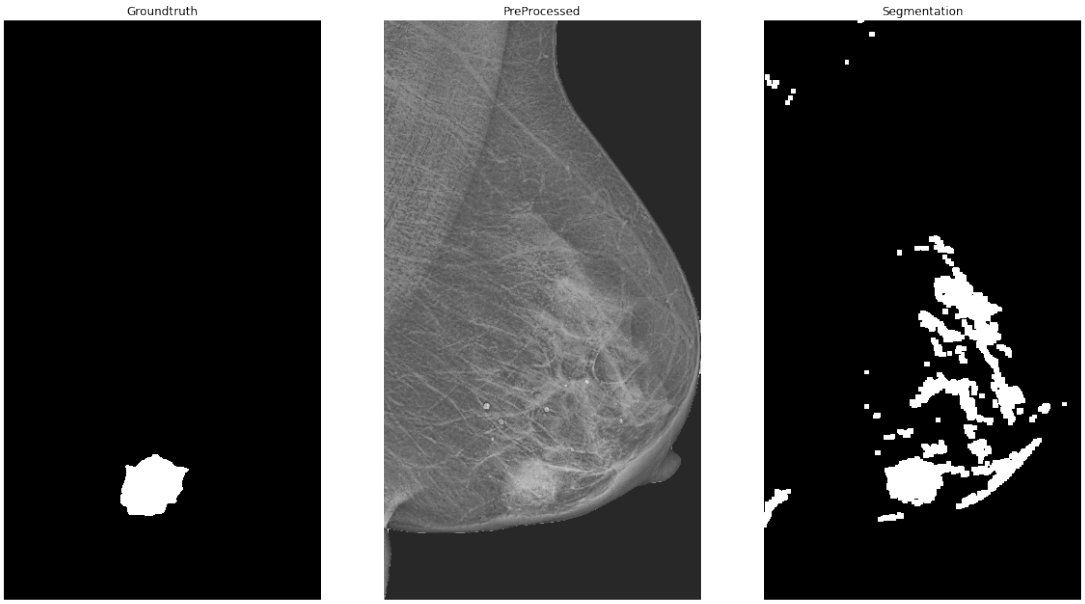


0.7413055687562103 (1, 99, 1)

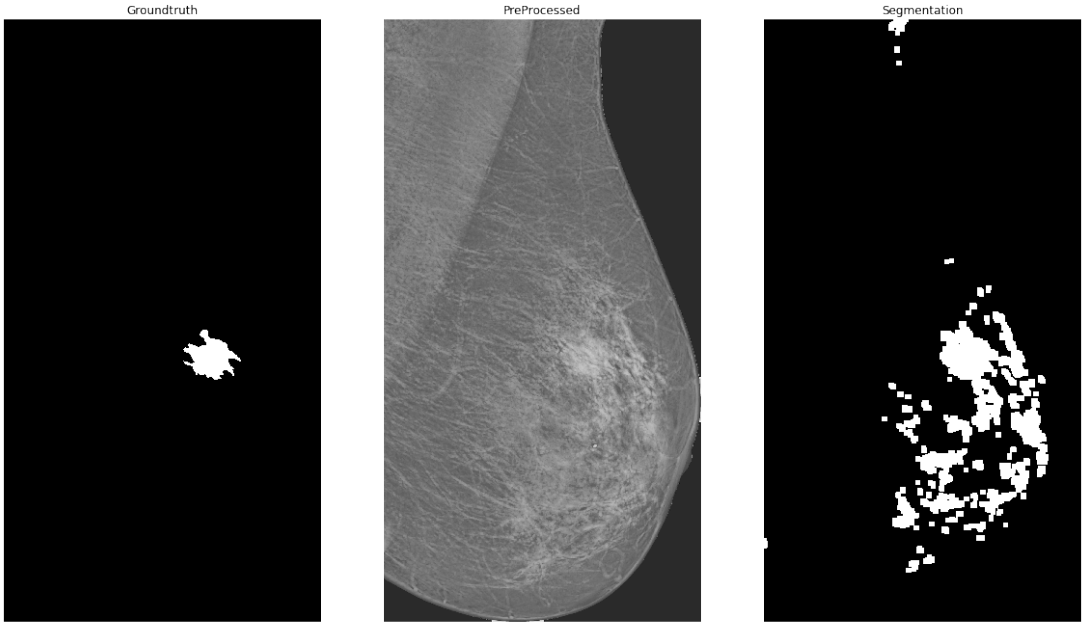

Groundtruth
PreProcessed
Segmentation

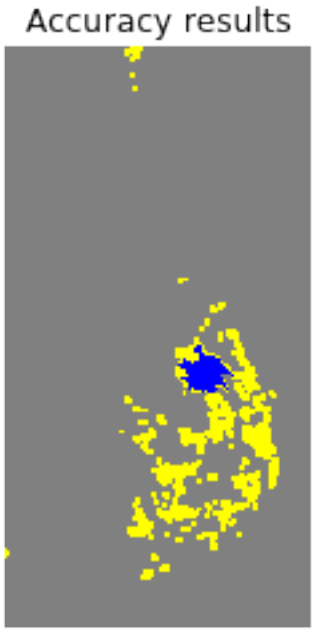

Accuracy results

```
0.7611130532691059 (1, 54, 1)
```

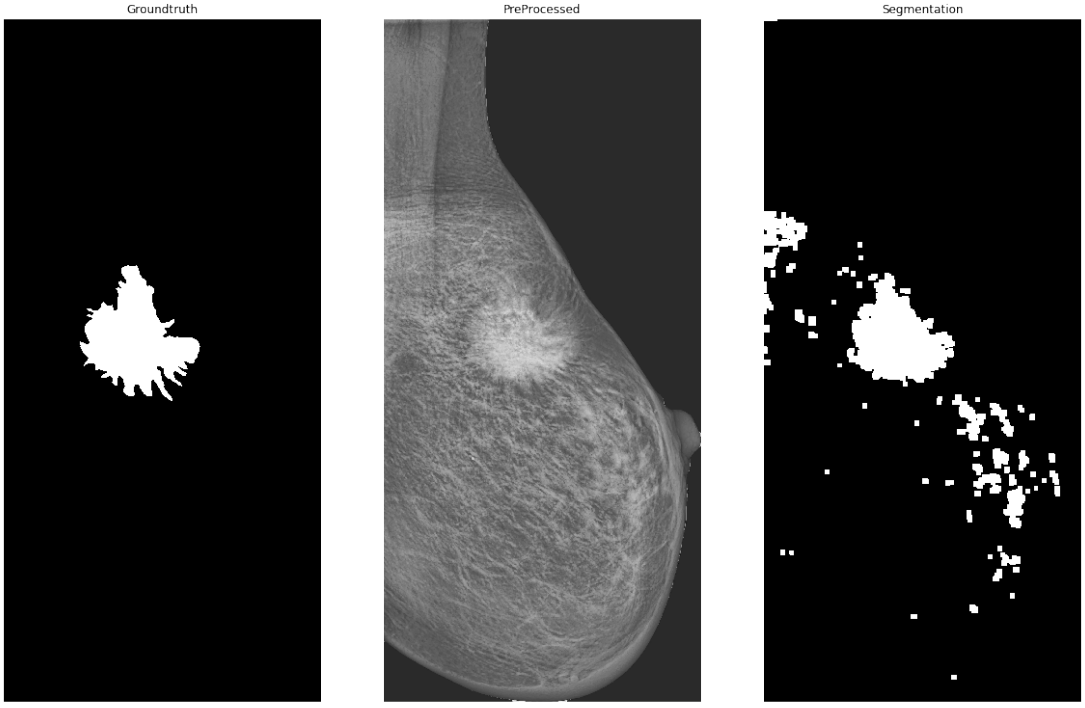


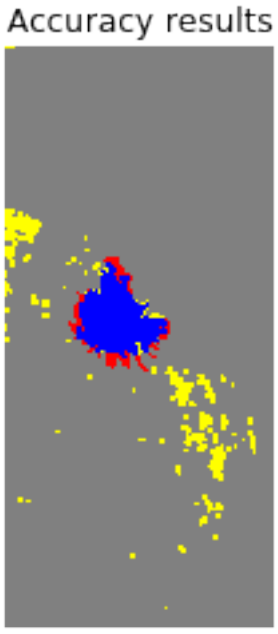
0.6723609183875745 (1, 36, 1)



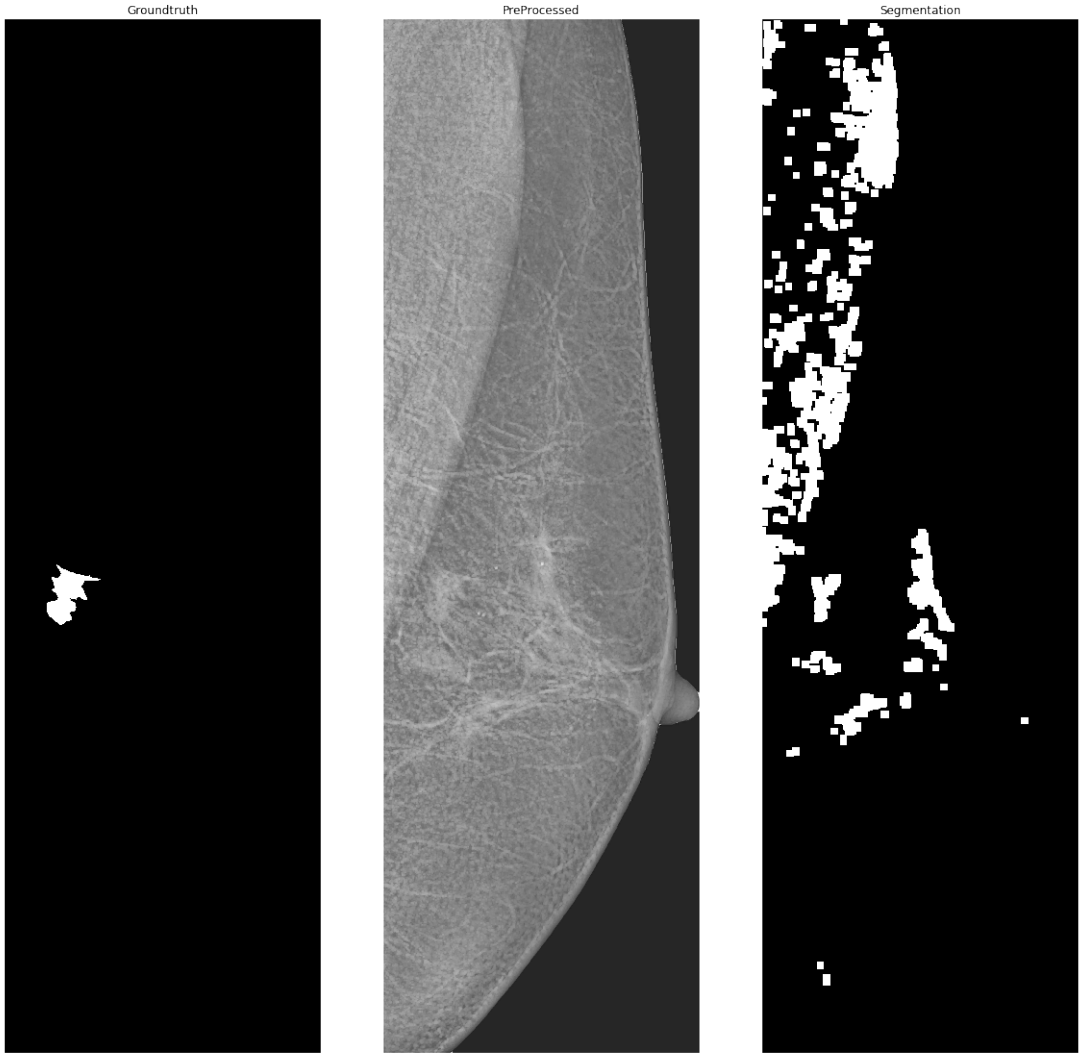


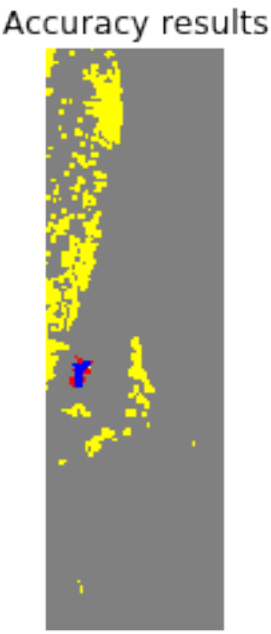
```
0.3361669086802194 (1, 47, 1)
```



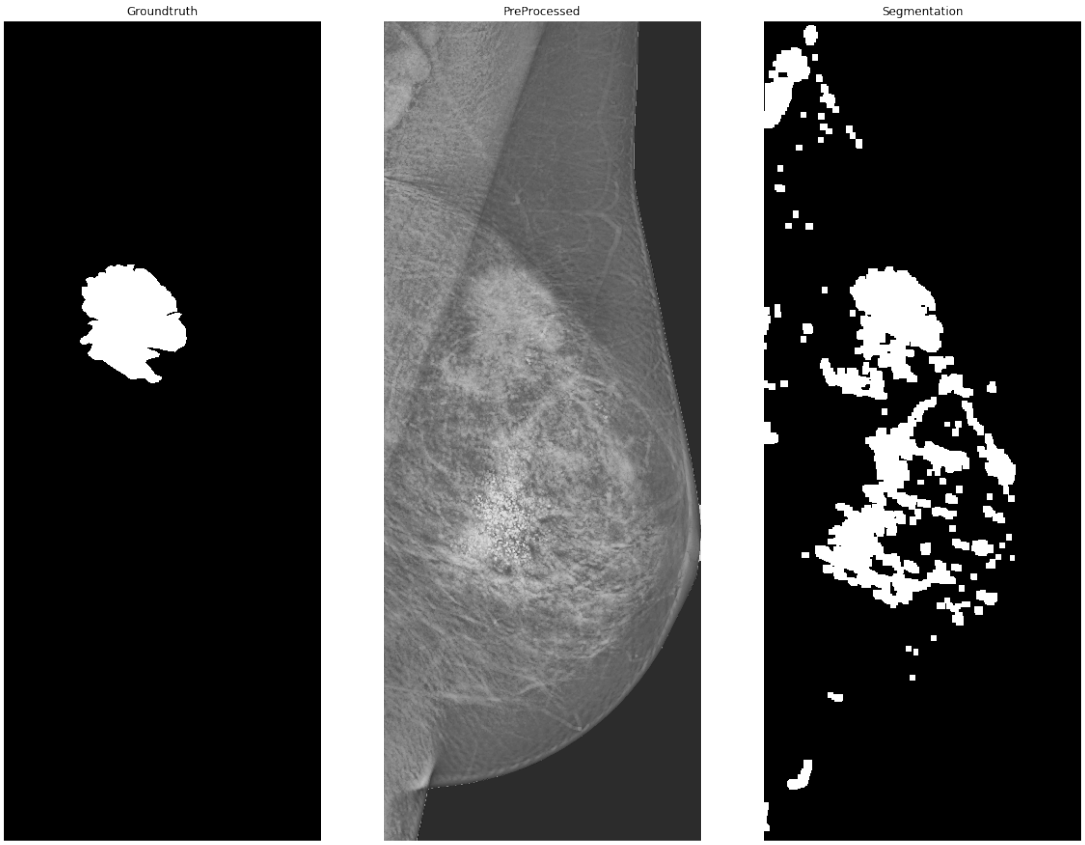


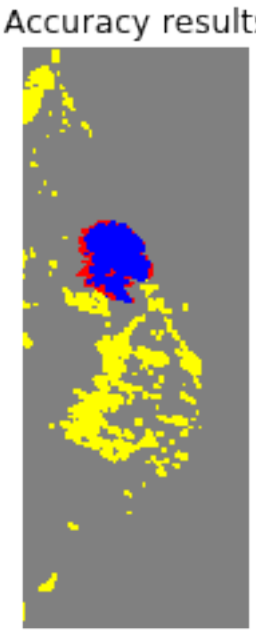
0.7594400802682133 (1, 61, 1)



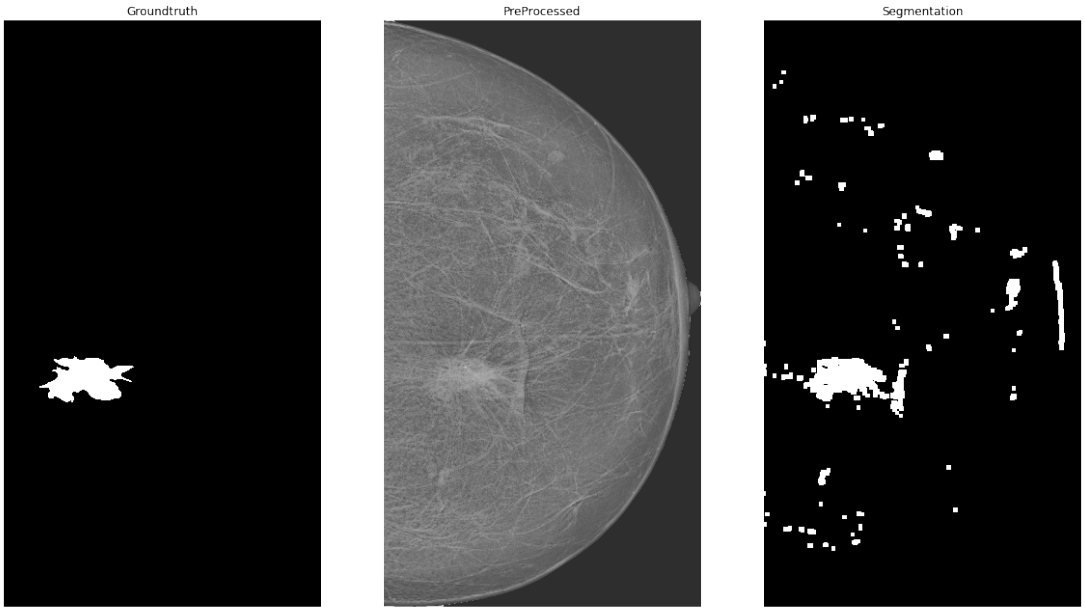


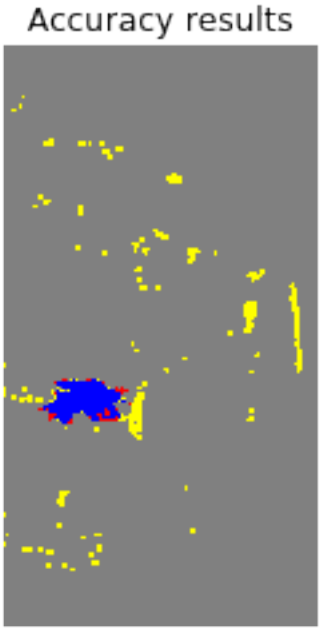

```
0.5970343928056566 (1, 67, 1)
```


Groundtruth
PreProcessed
Segmentation

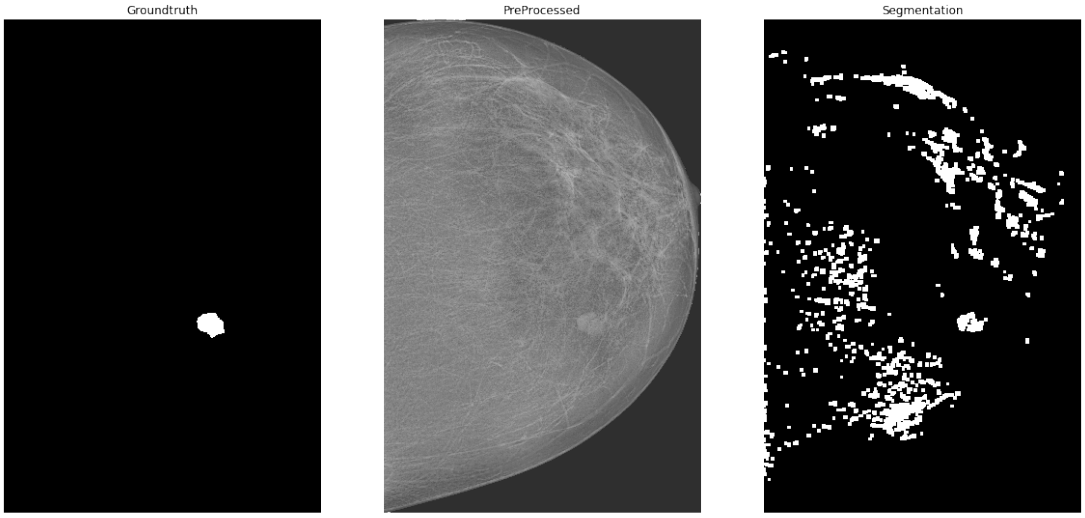

Accuracy results

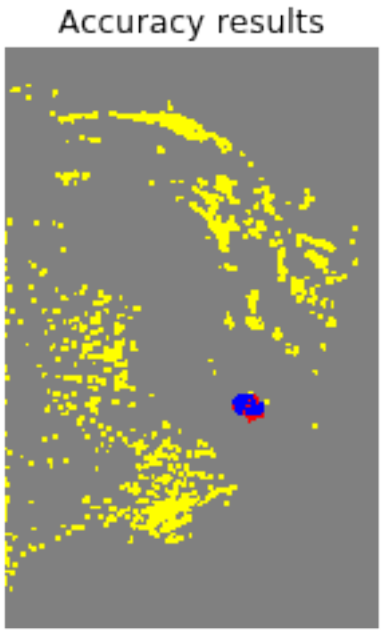
```
0.805221142545934 (1, 65, 1)
```



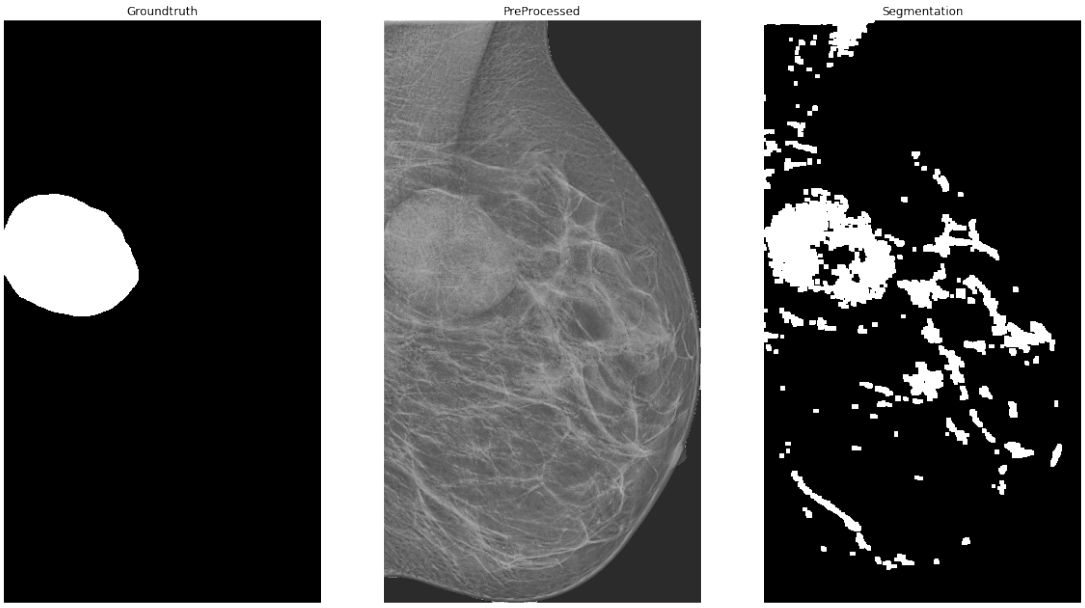


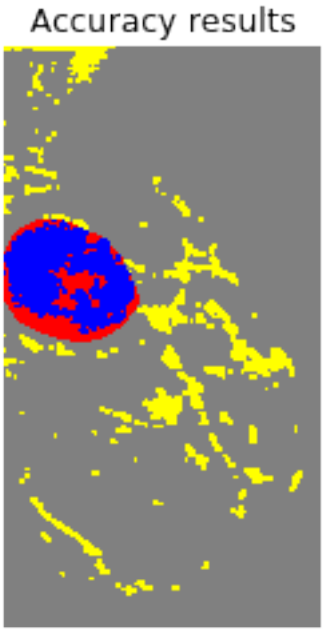
0.7967926135709517 (1, 68, 1)



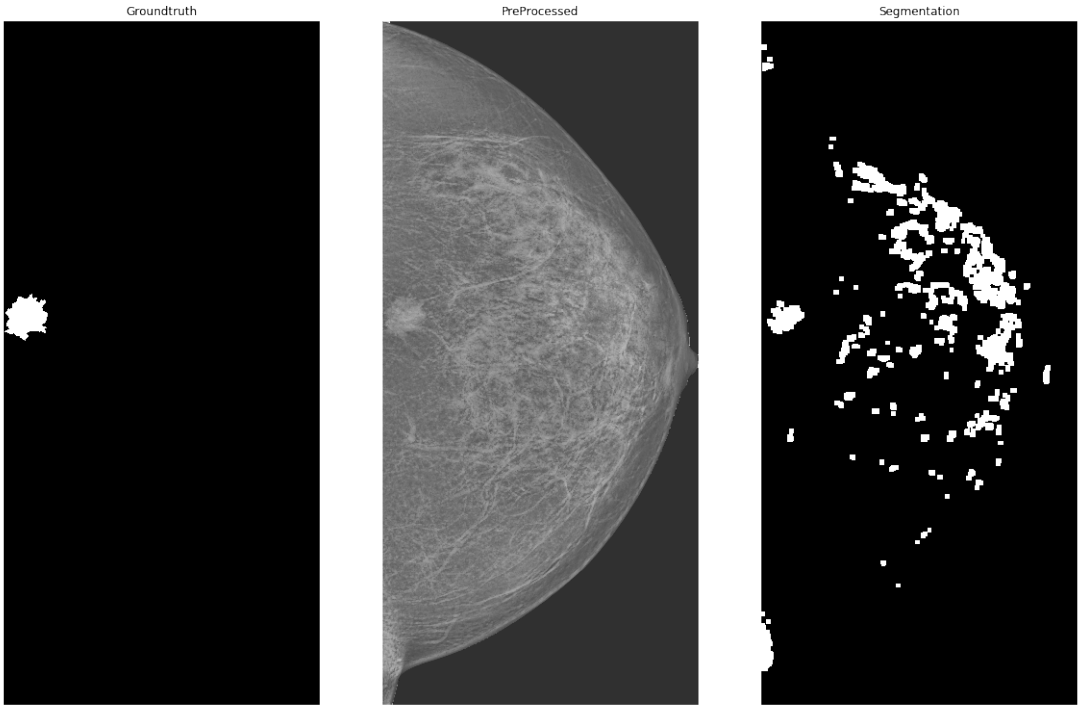


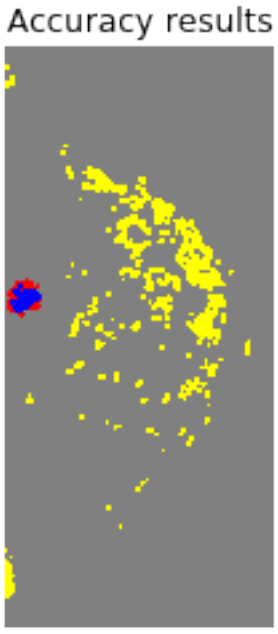
0.7435056953208228 (1, 246, 1)



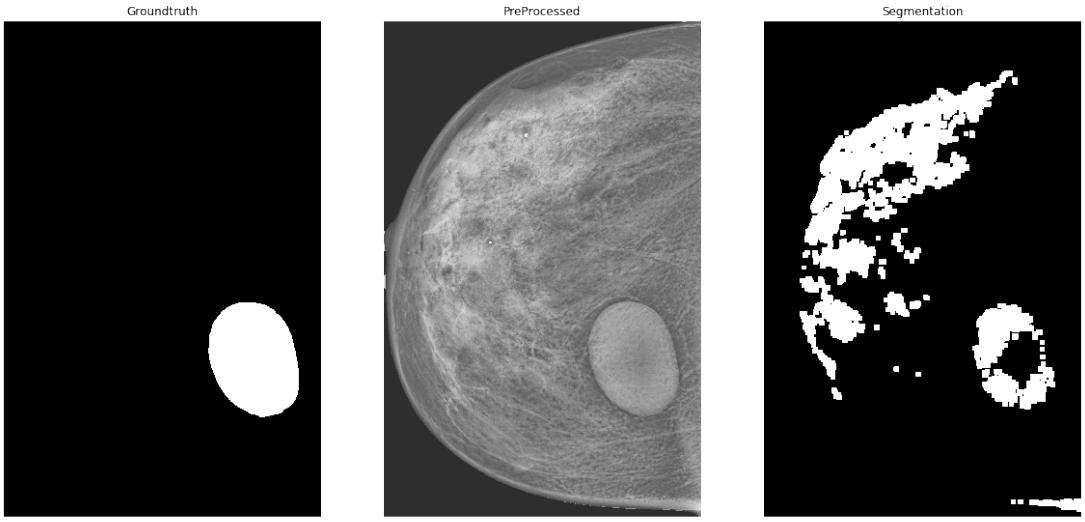


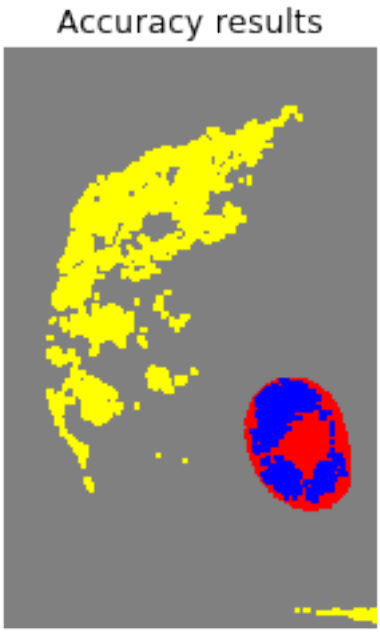
0.7730204346501741 (1, 104, 1)



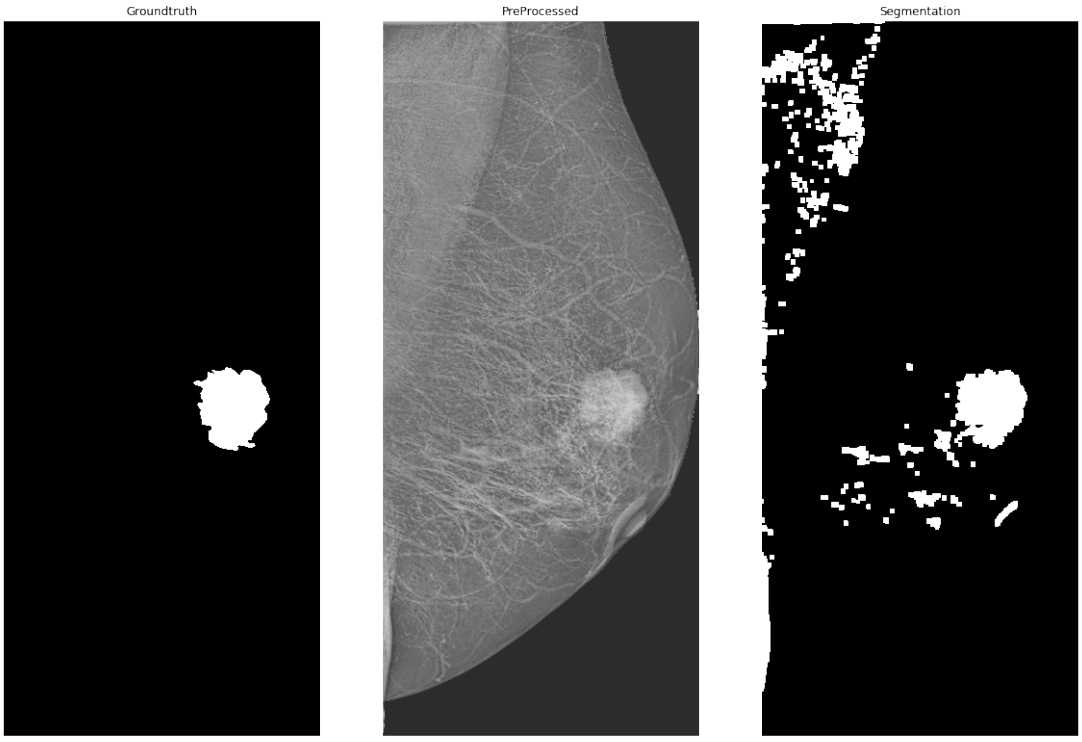


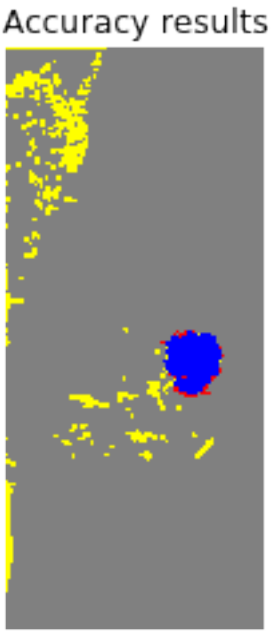
0.5879857408007542 (1, 82, 1)



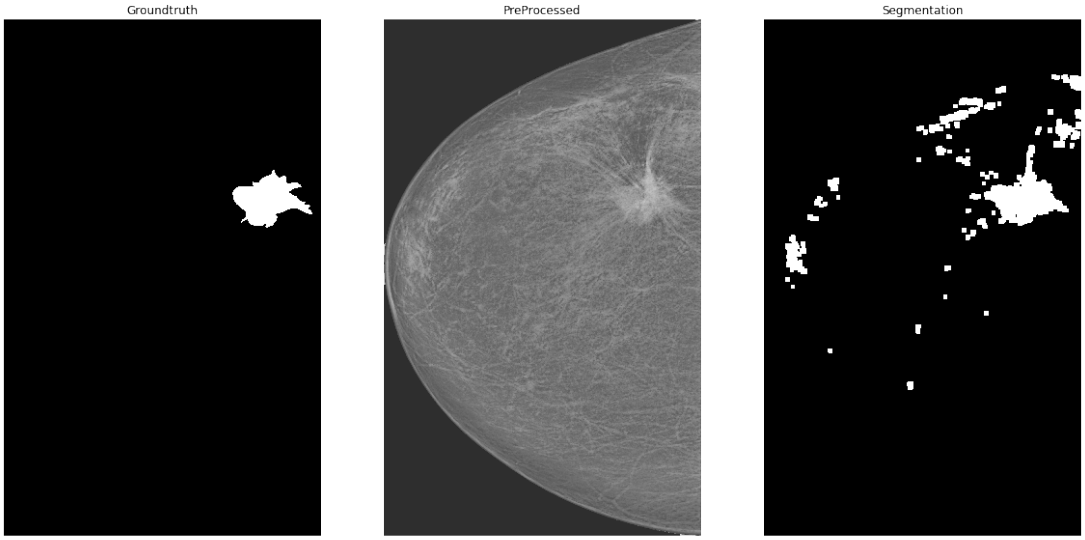


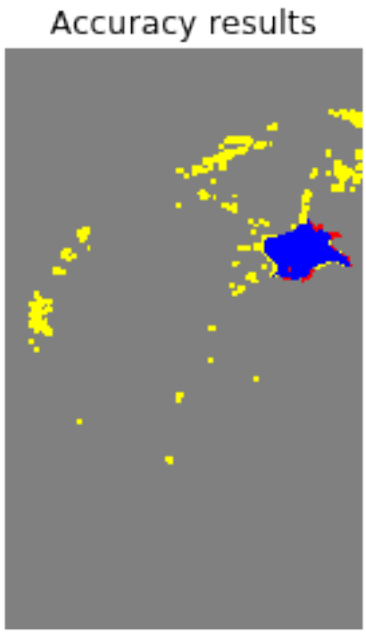
0.28712627972928473 (2, 26, 1)


Groundtruth
PreProcessed
Segmentation

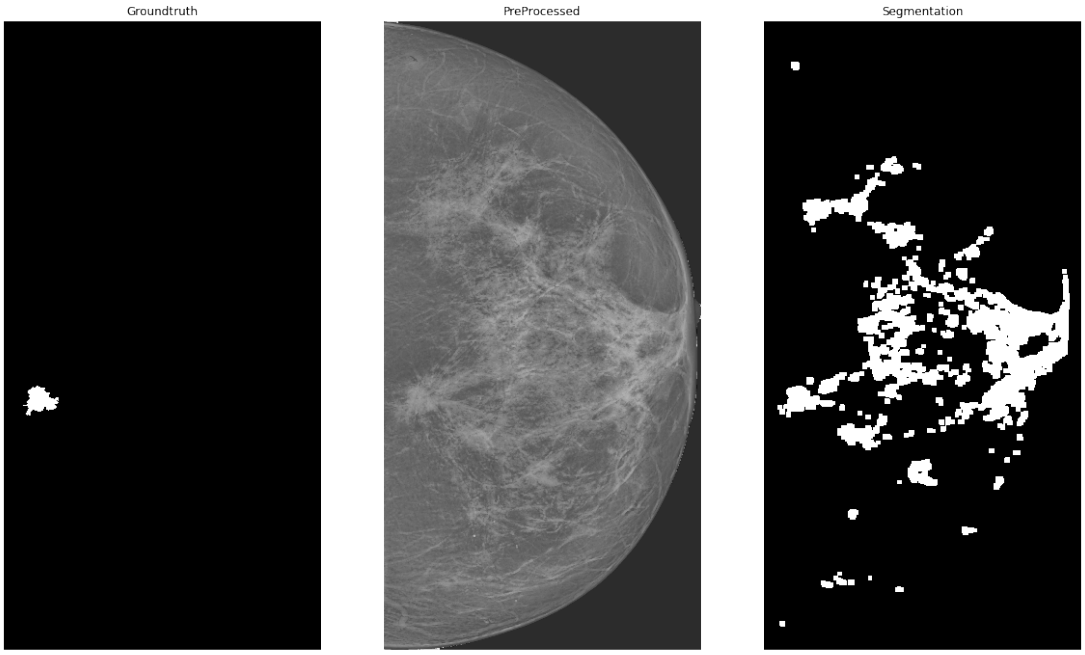

Accuracy results

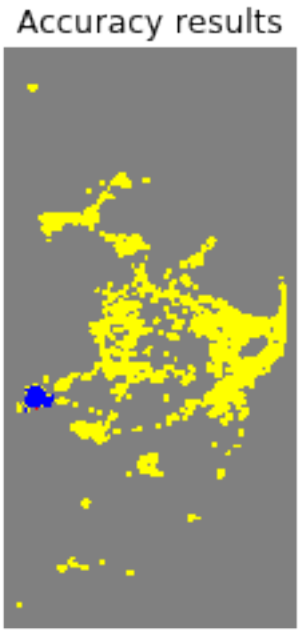
0.8741970238239892 (1, 87, 1)



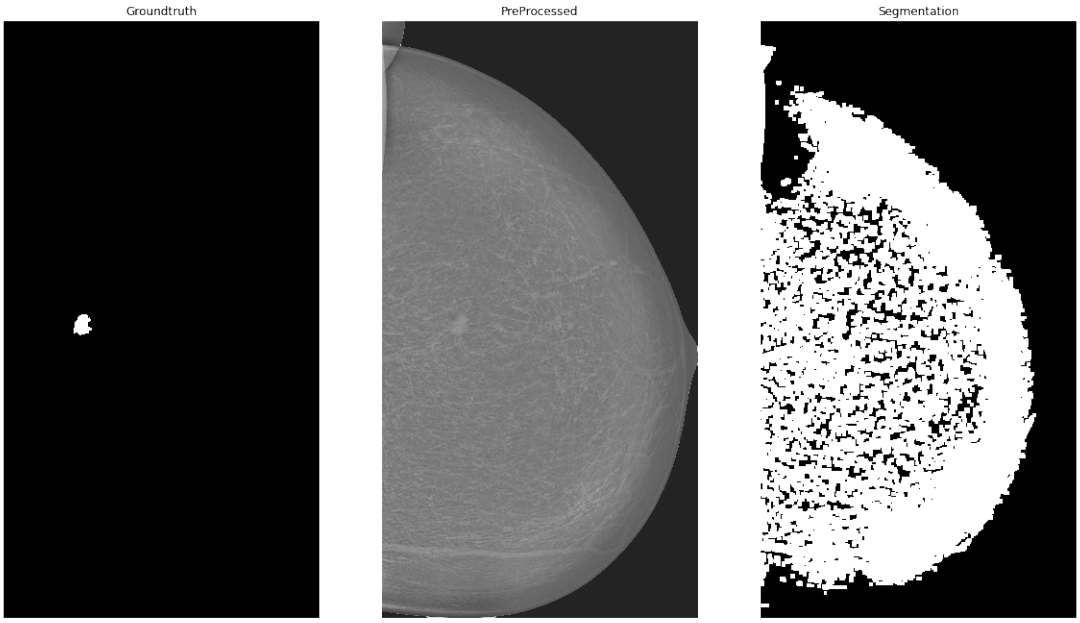


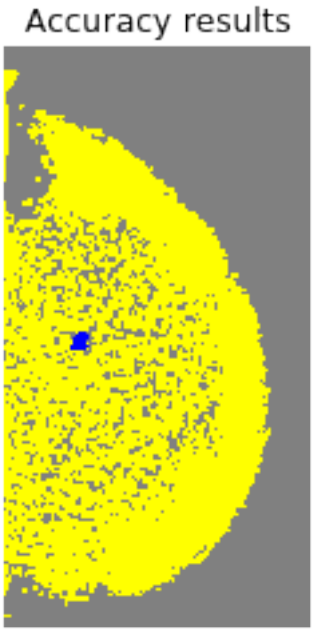
0.7416784166275001 (1, 42, 1)



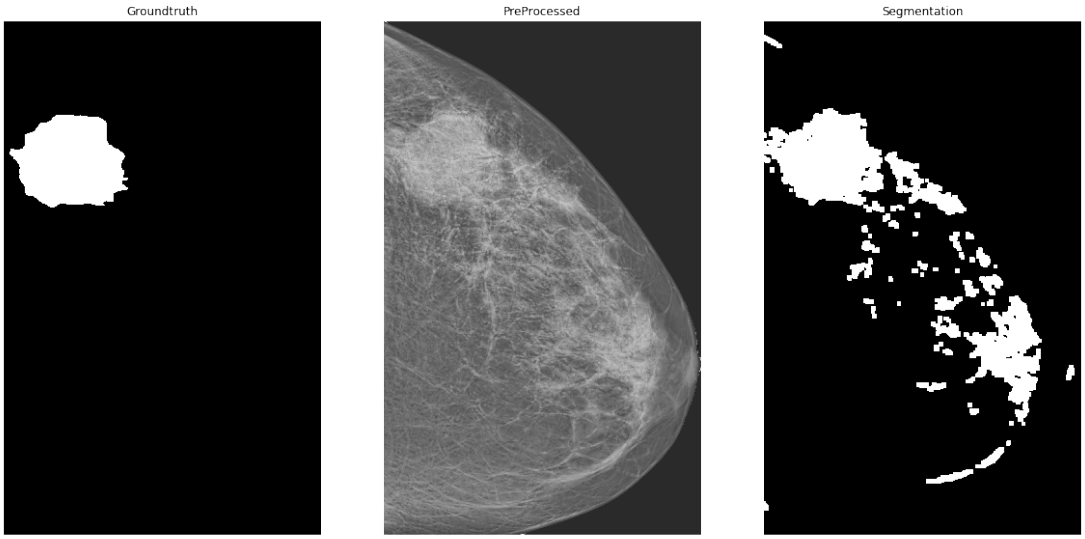


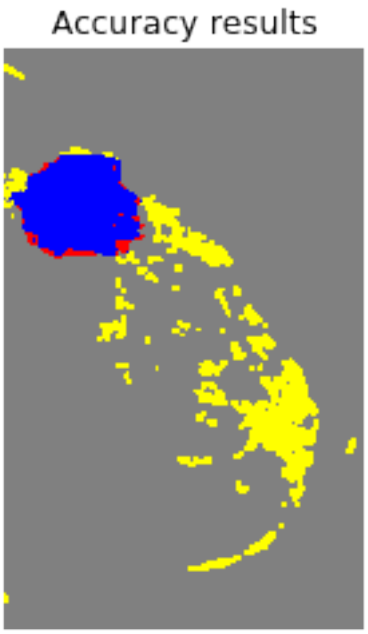
0.6731947164465186 (1, 58, 1)



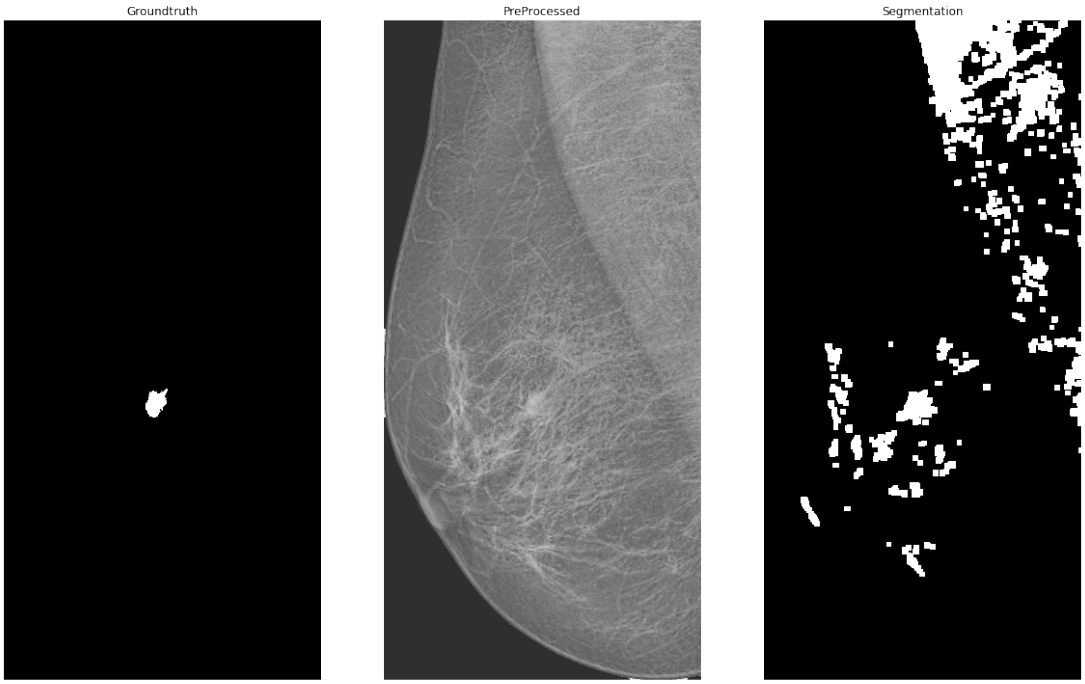


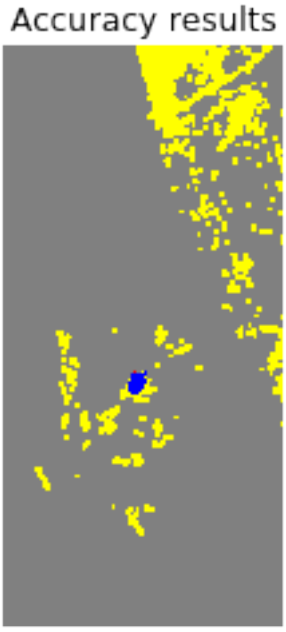
0.0026079151359073177 (0, 17, 1)



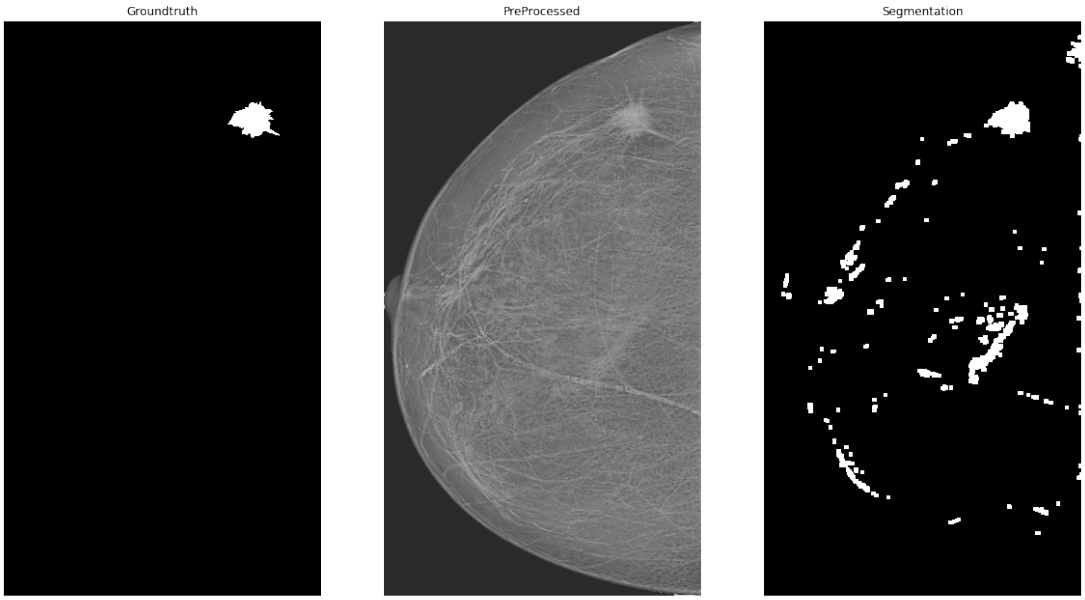


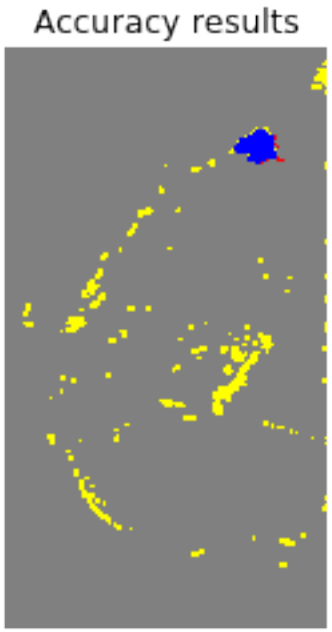
0.8486962632067658 (1, 47, 1)



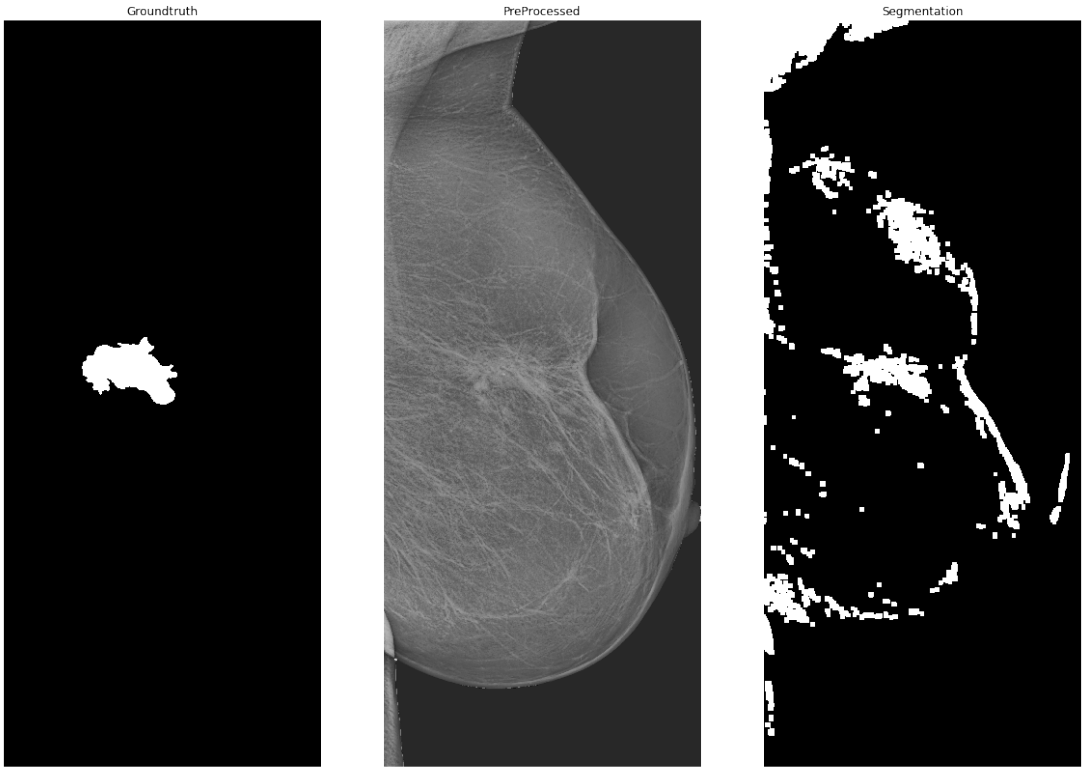


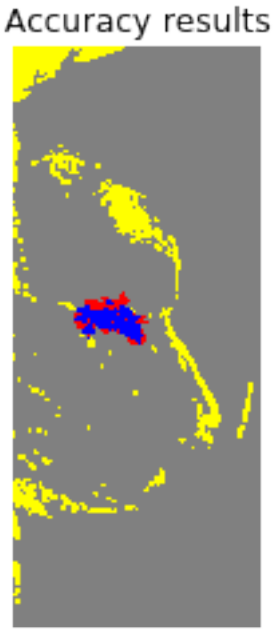
0.4701103364988747 (1, 95, 1)



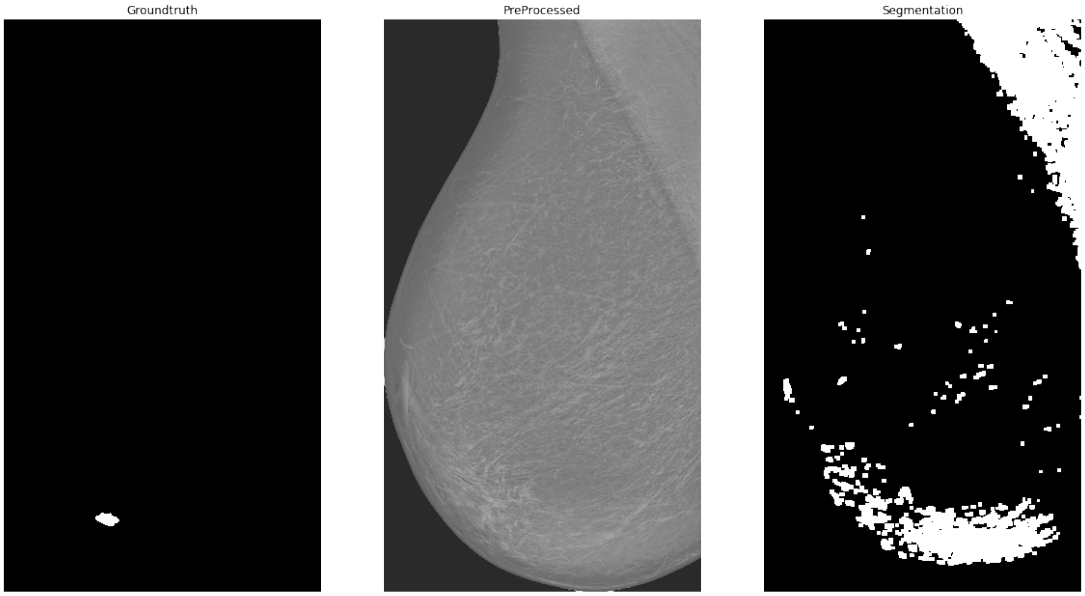


0.8430016361743269 (1, 91, 1)

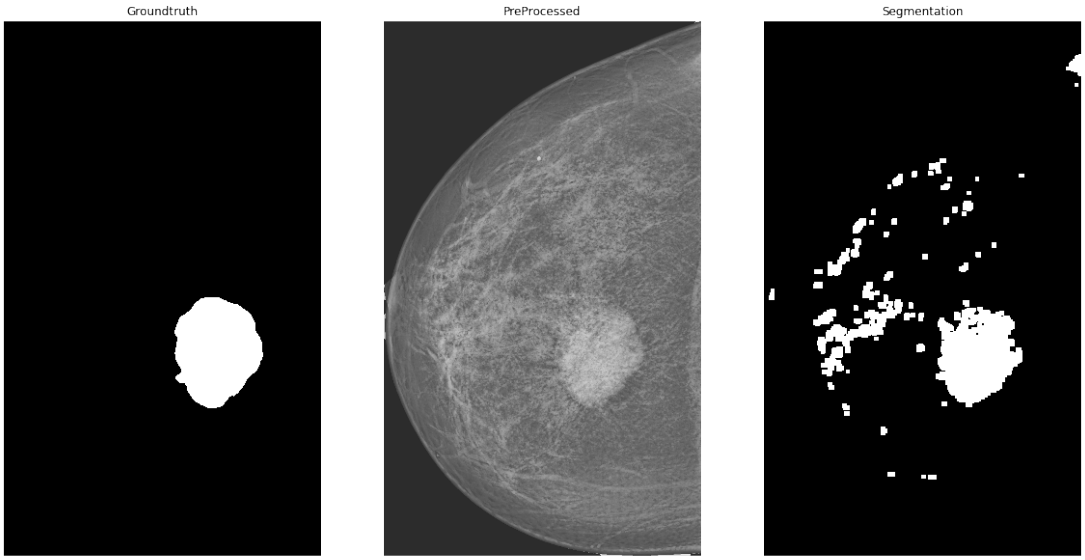

Groundtruth
PreProcessed
Segmentation

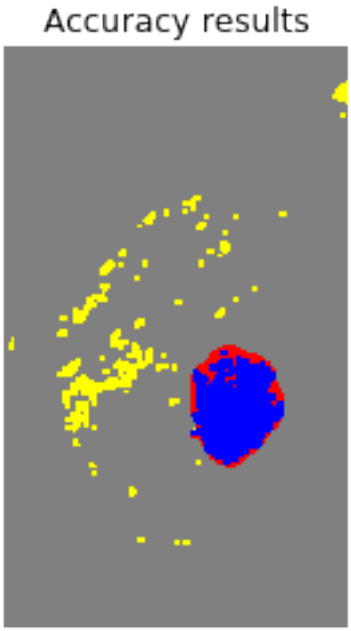

Accuracy results

0.6077063625644672 (1, 85, 1)

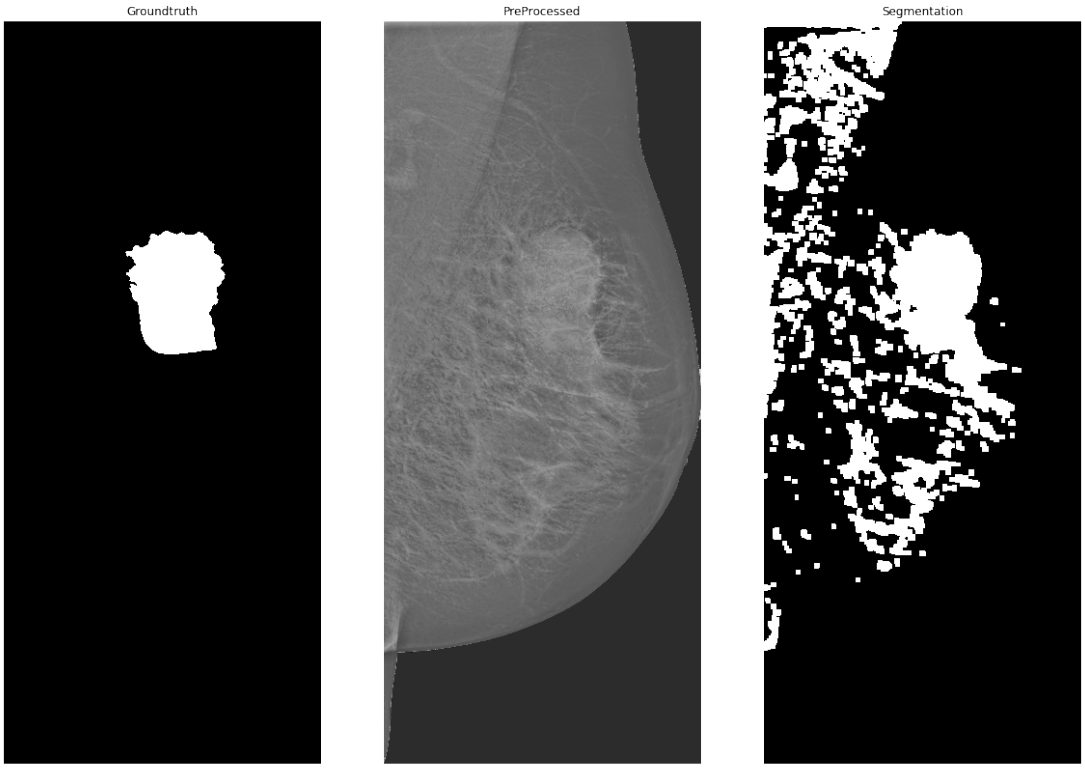


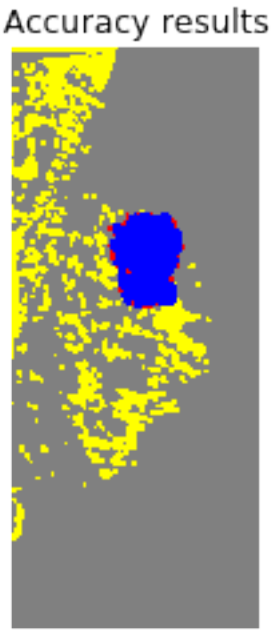
0.2420187080930438 (1, 81, 1)



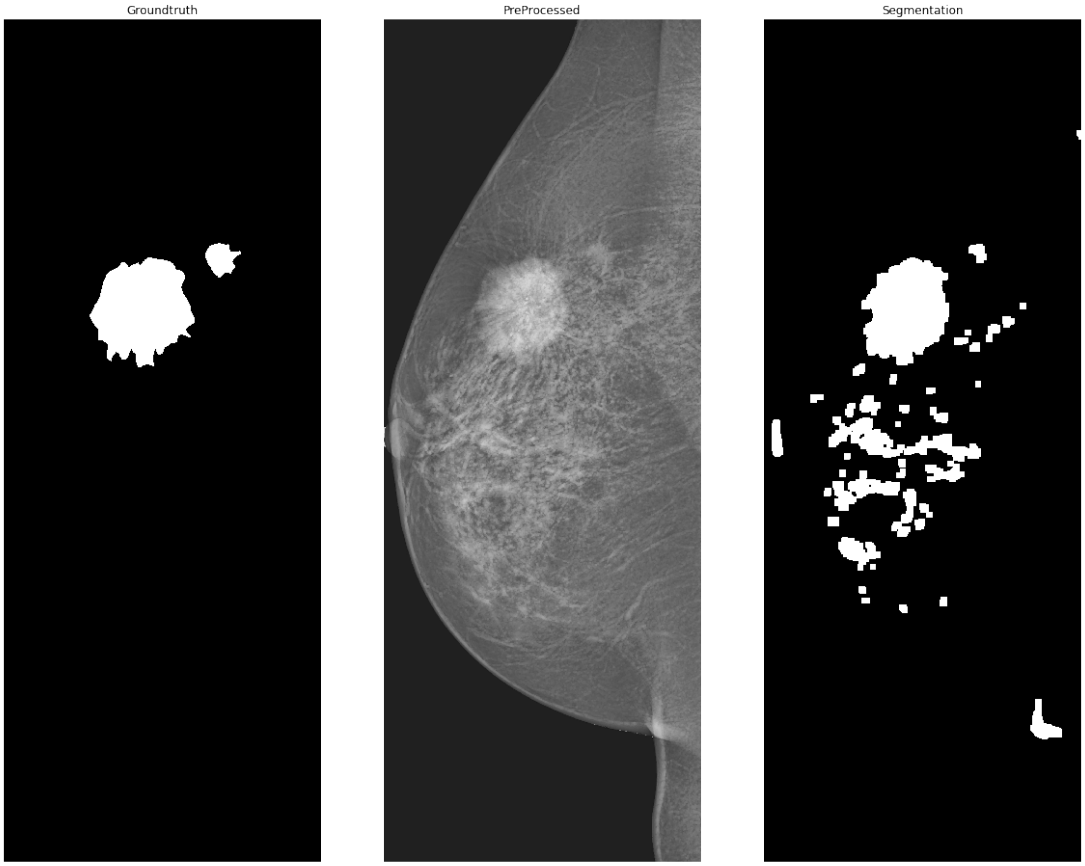


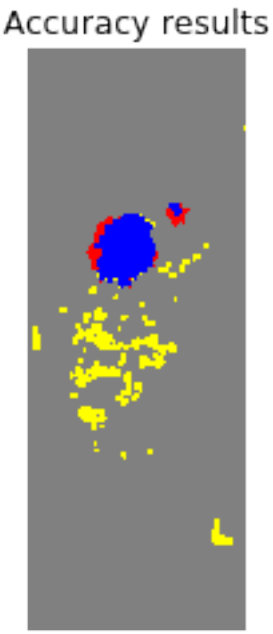
0.8161482349289315 (1, 50, 1)


Groundtruth
PreProcessed
Segmentation

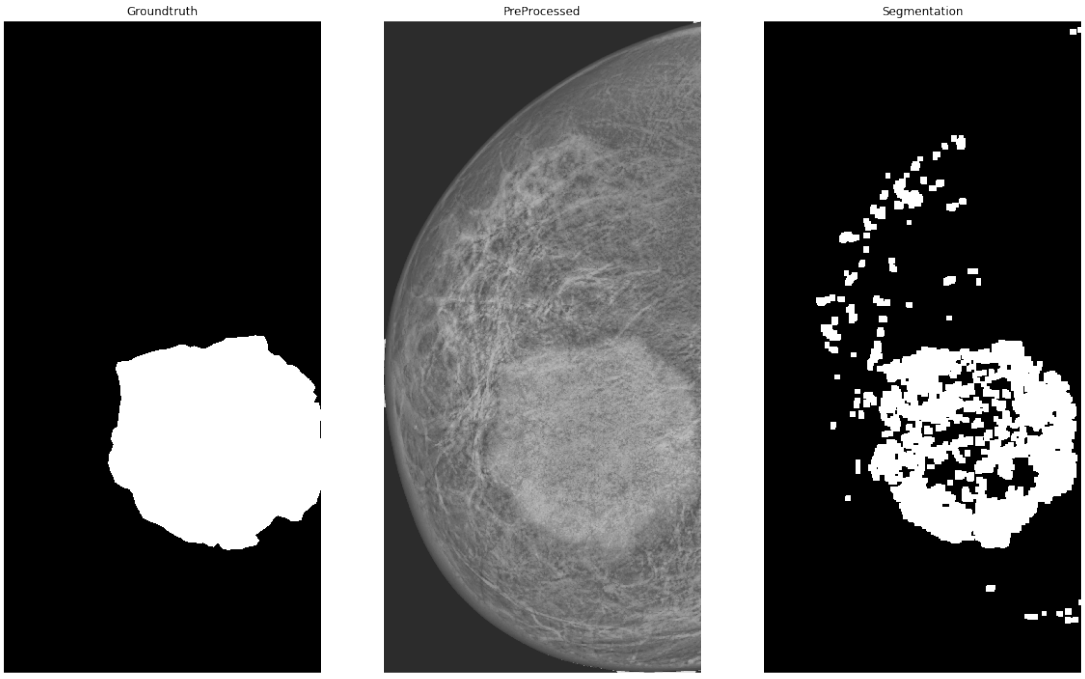

Accuracy results

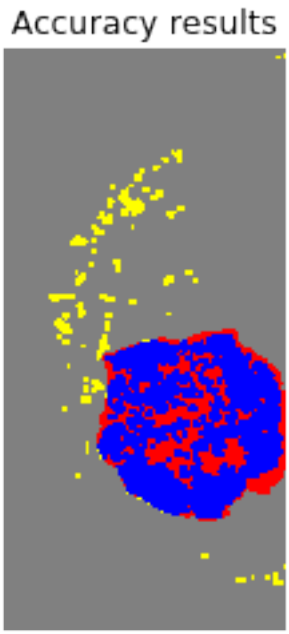
```
0.6934106976123783 (1, 114, 1)
```



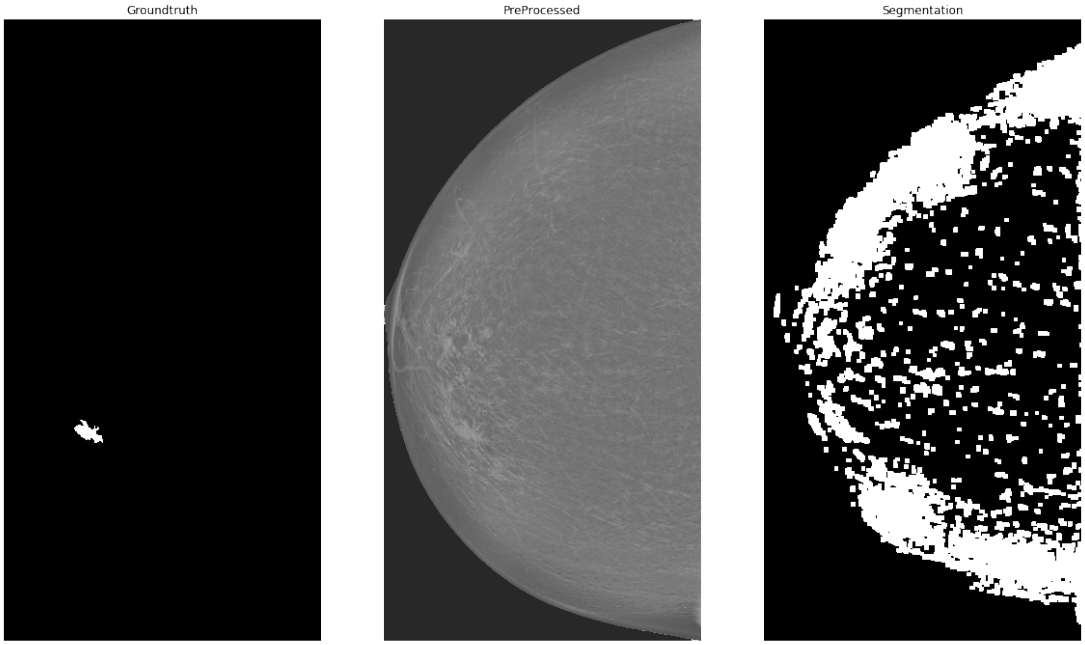


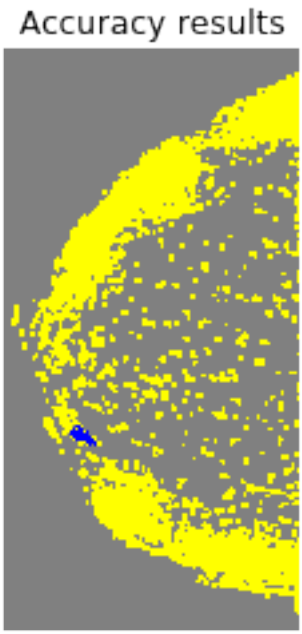

```
0.837897619549189 (2, 36, 2)
```



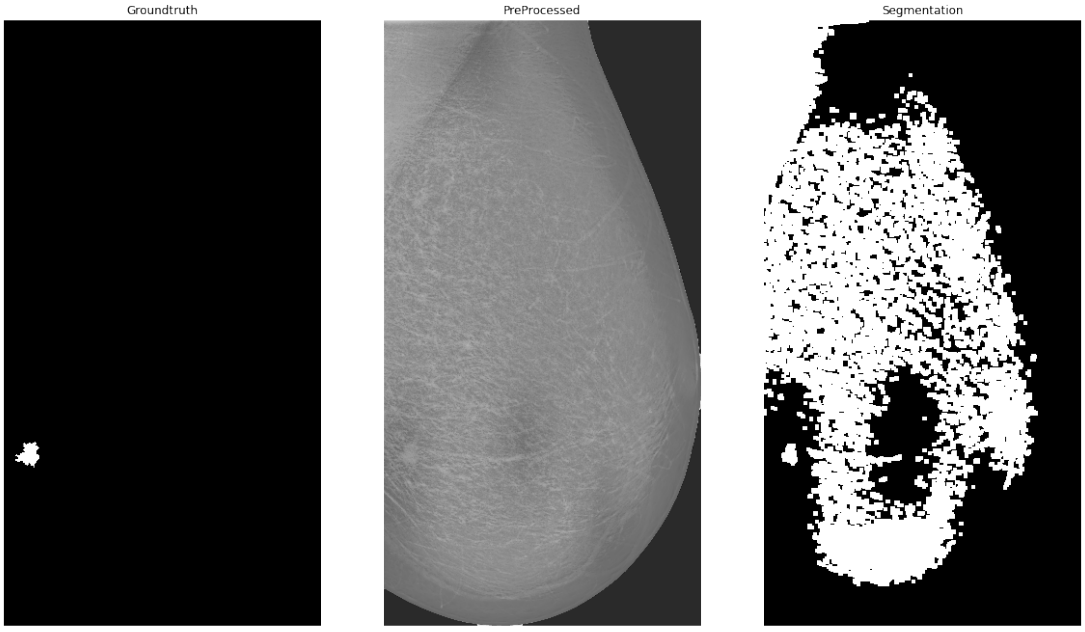


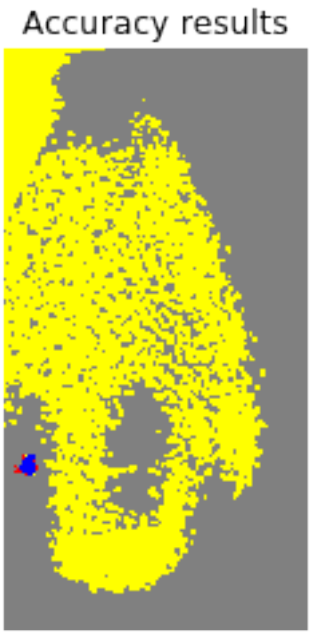
0.8926017198890172 (1, 51, 1)



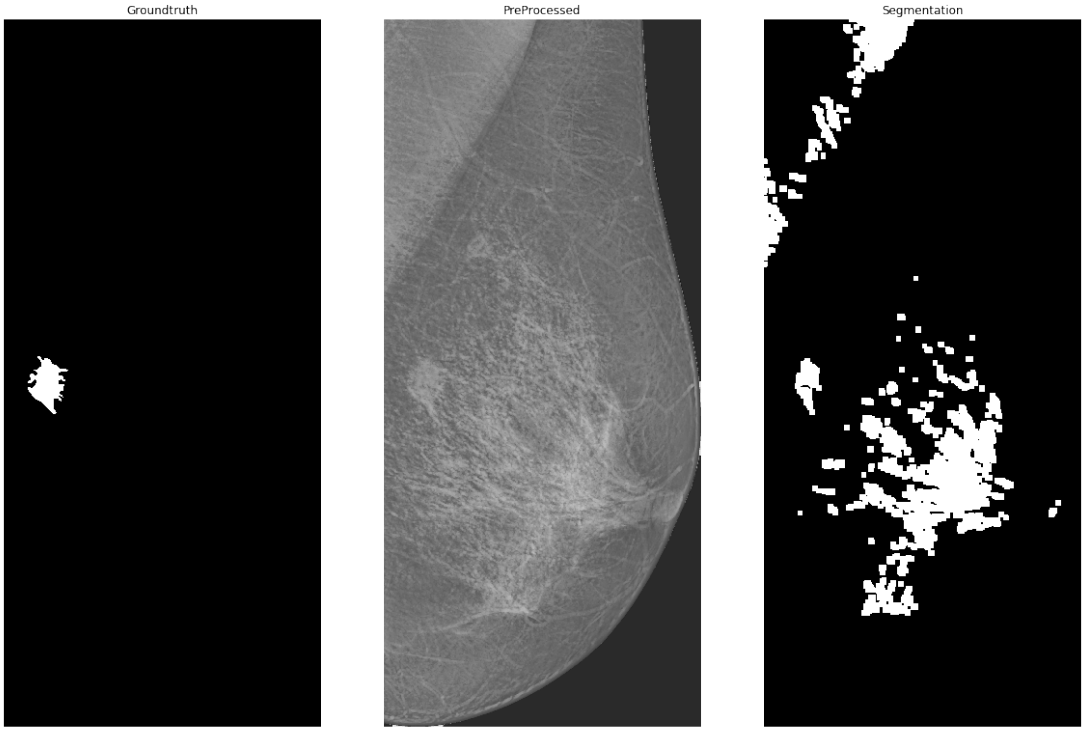


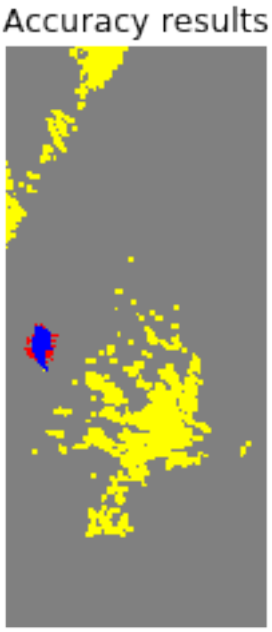

```
0.2564057119708325 (1, 237, 1)
```



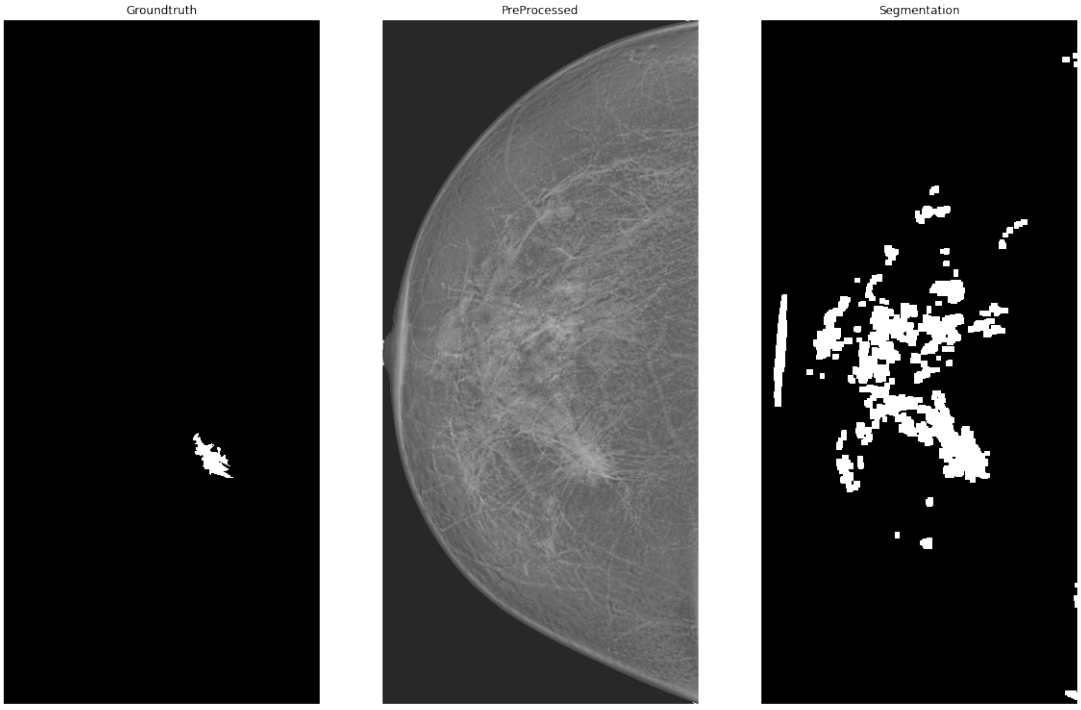


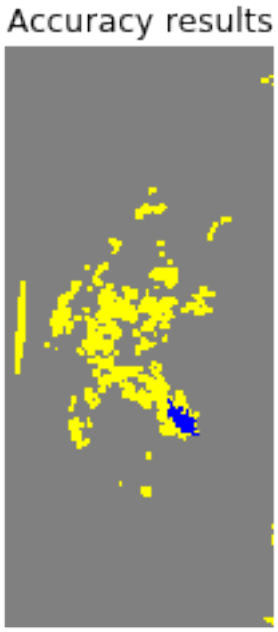

```
0.7231158772713733 (1, 43, 1)
```



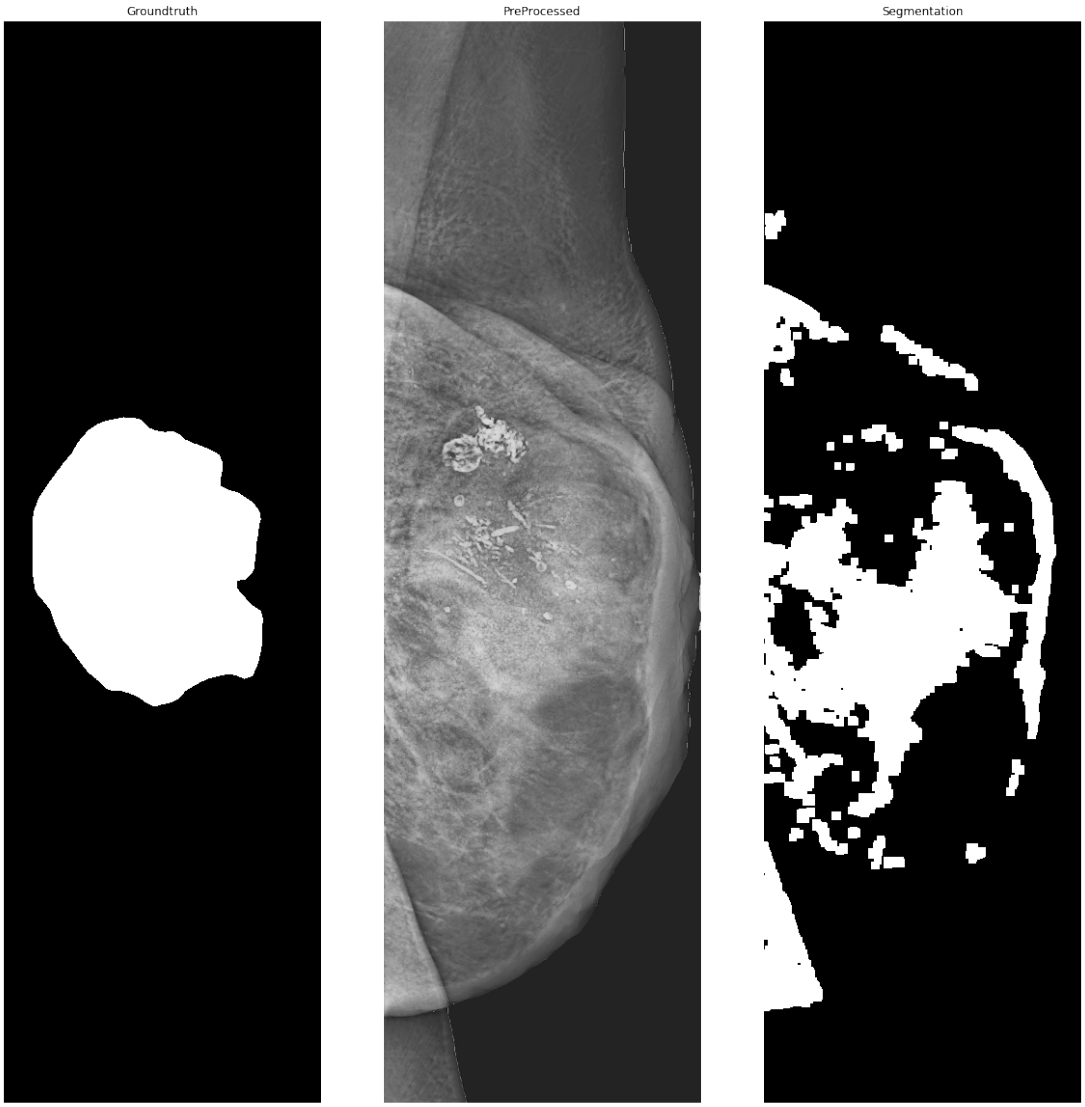


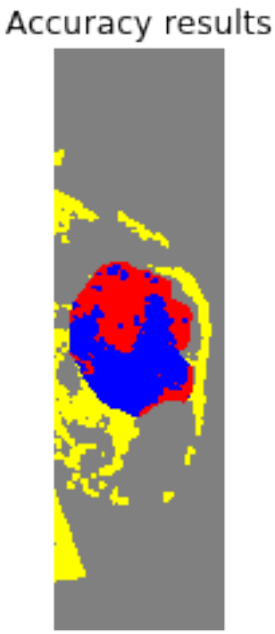

```
0.6914385605866803 (1, 53, 1)
```



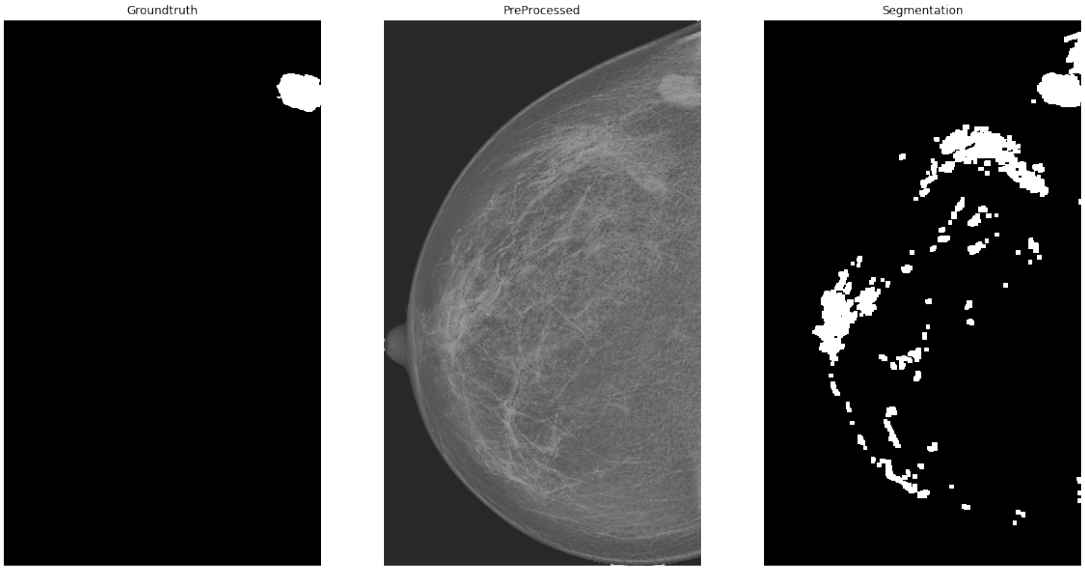



```
0.13209318593926897 (1, 39, 1)
```

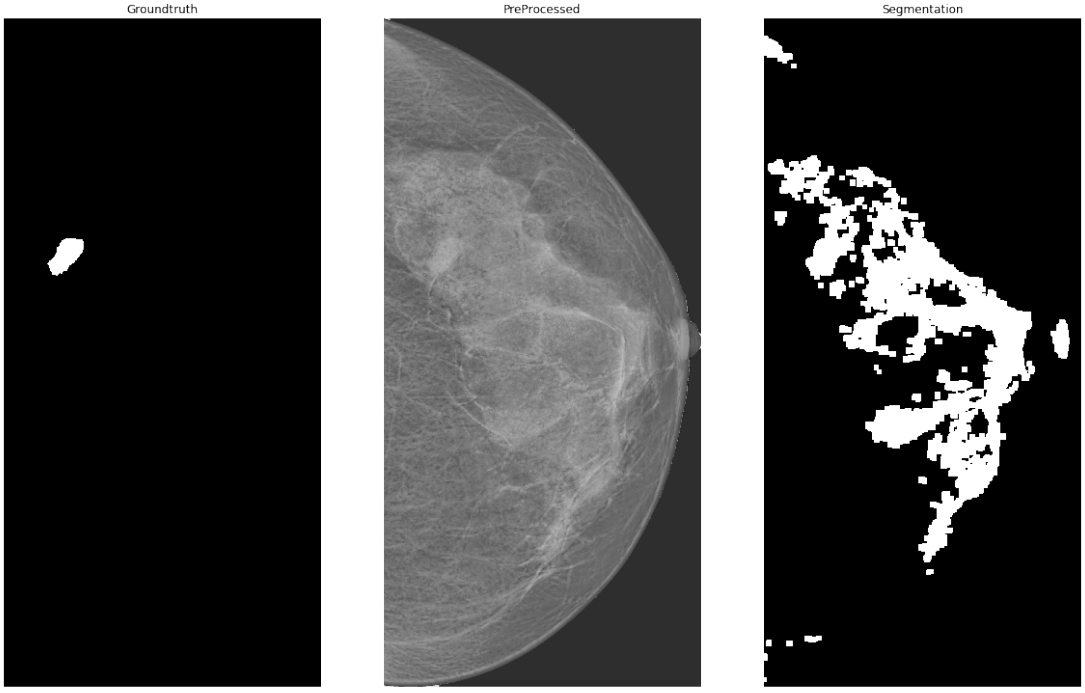

Groundtruth
PreProcessed
Segmentation

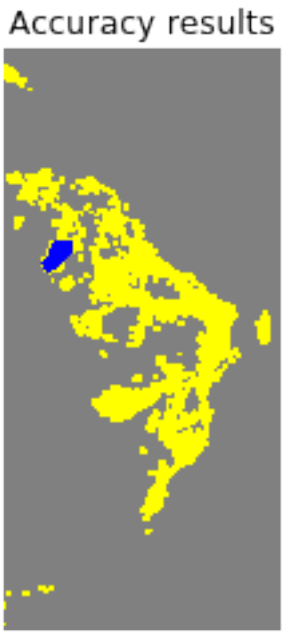



```
0.3784759964737198 (1, 30, 1)
```

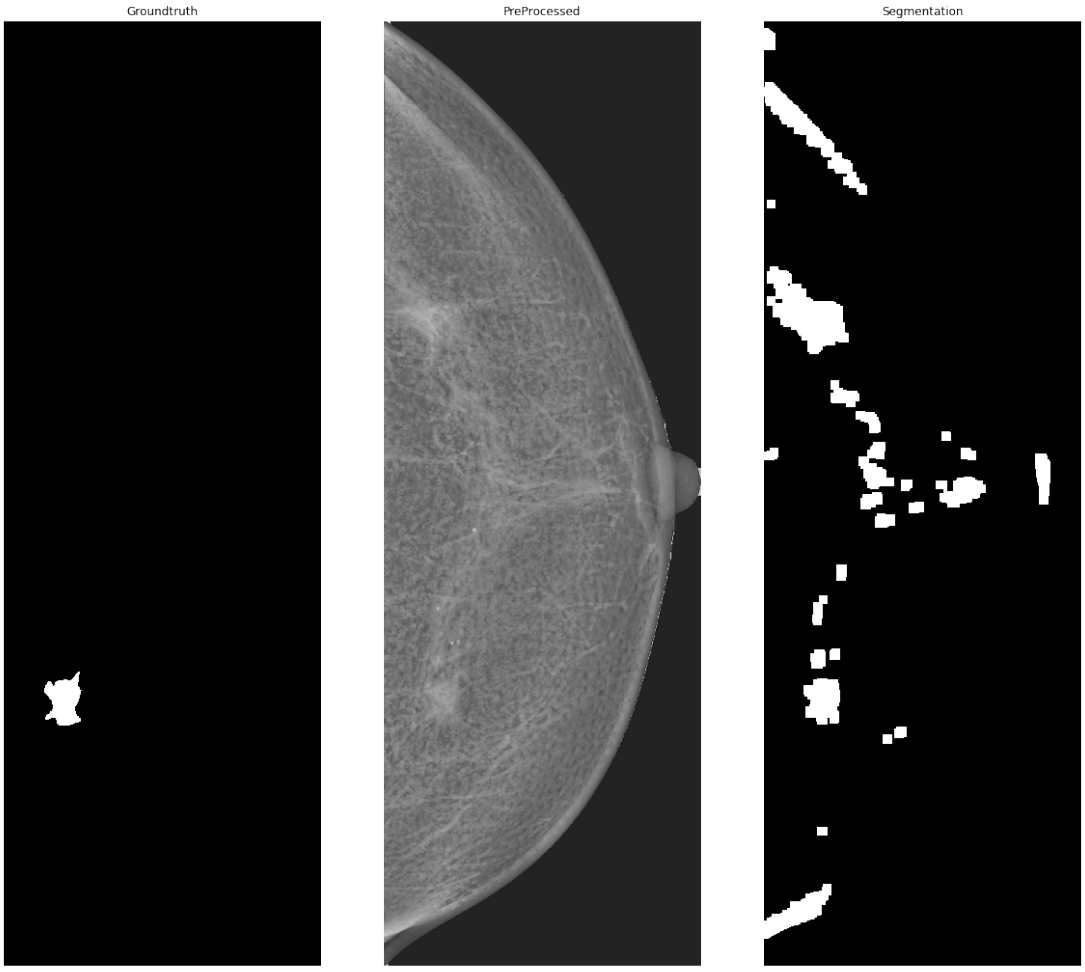



```
0.6512408971274483 (1, 66, 1)
```

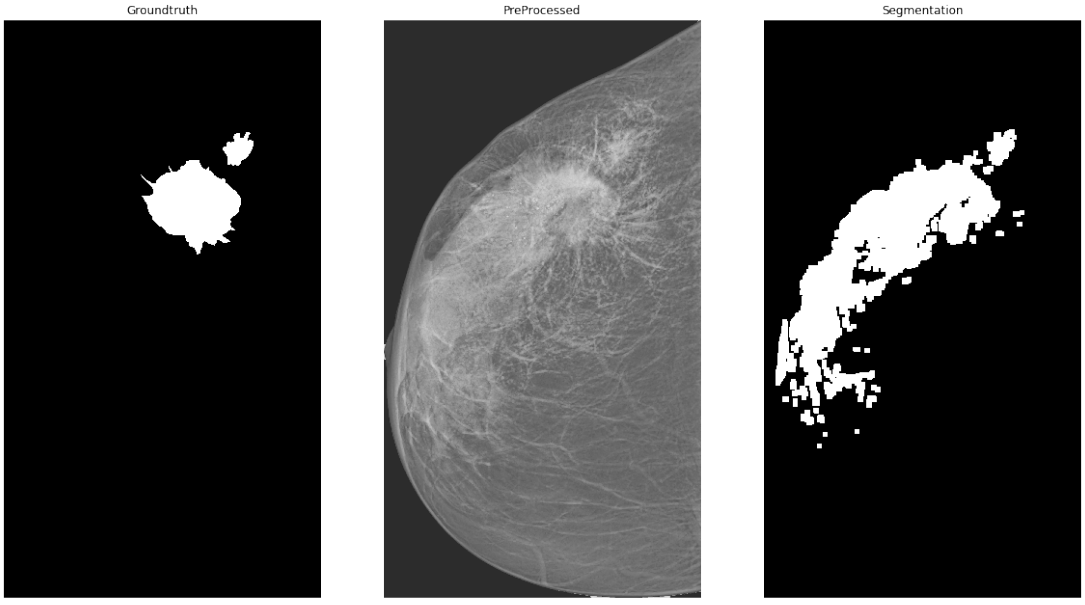


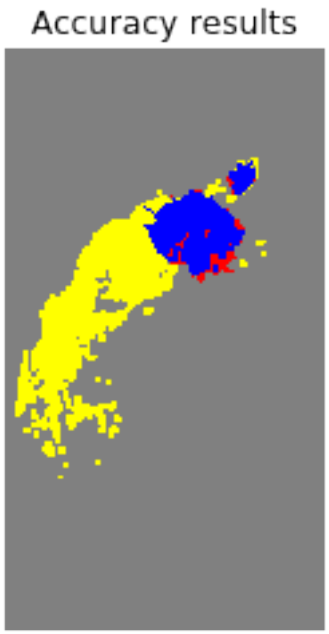



```
0.18808582017658193 (1, 27, 1)
```

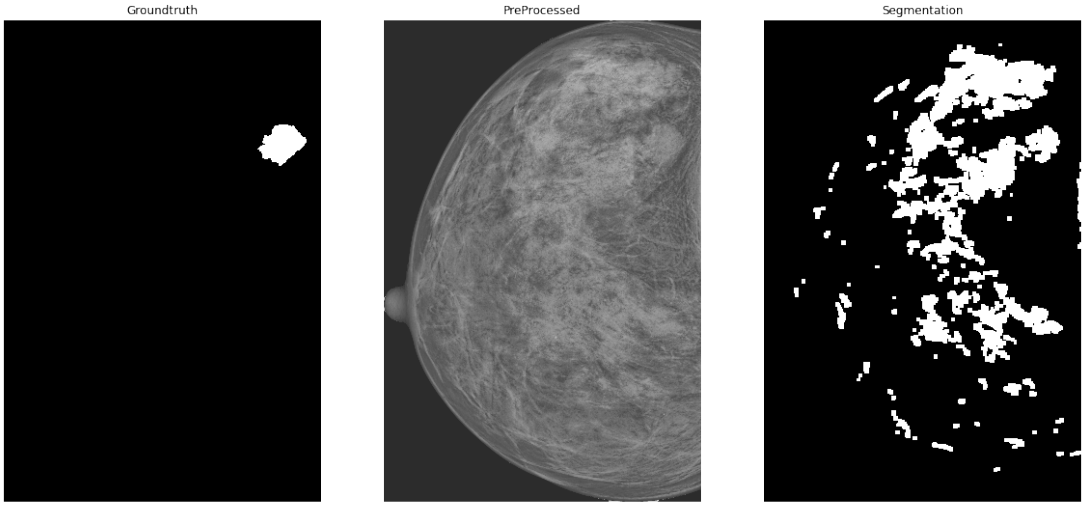

Groundtruth
PreProcessed
Segmentation

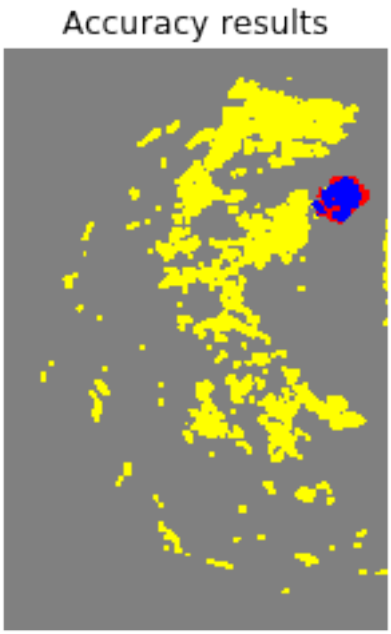
0.8048299191881225 (1, 24, 1)



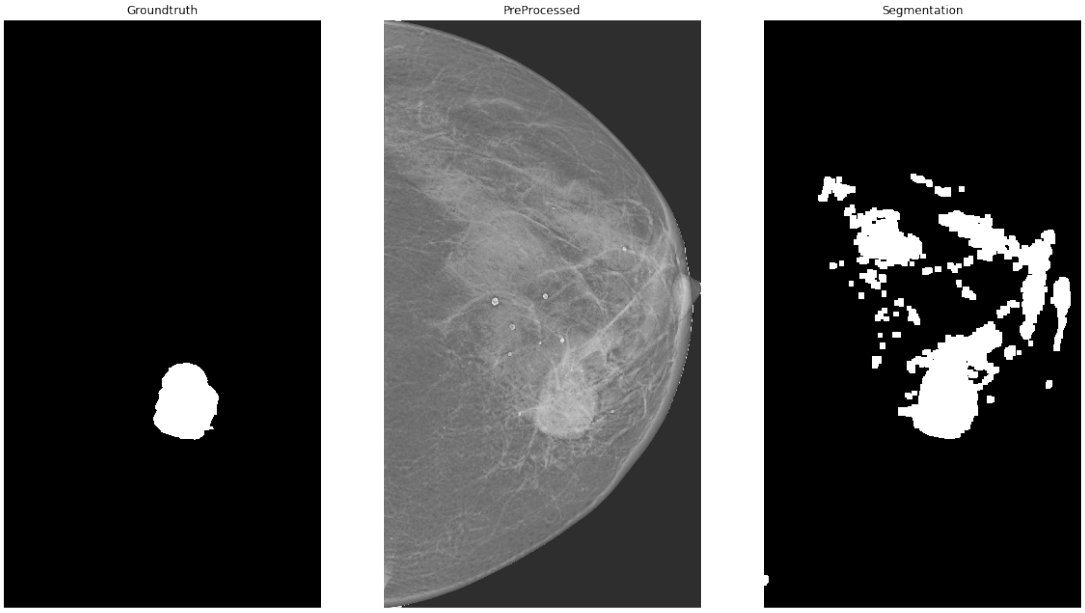


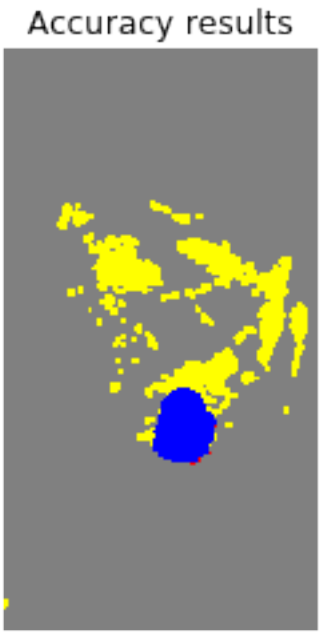
0.7474235326536236 (2, 18, 2)



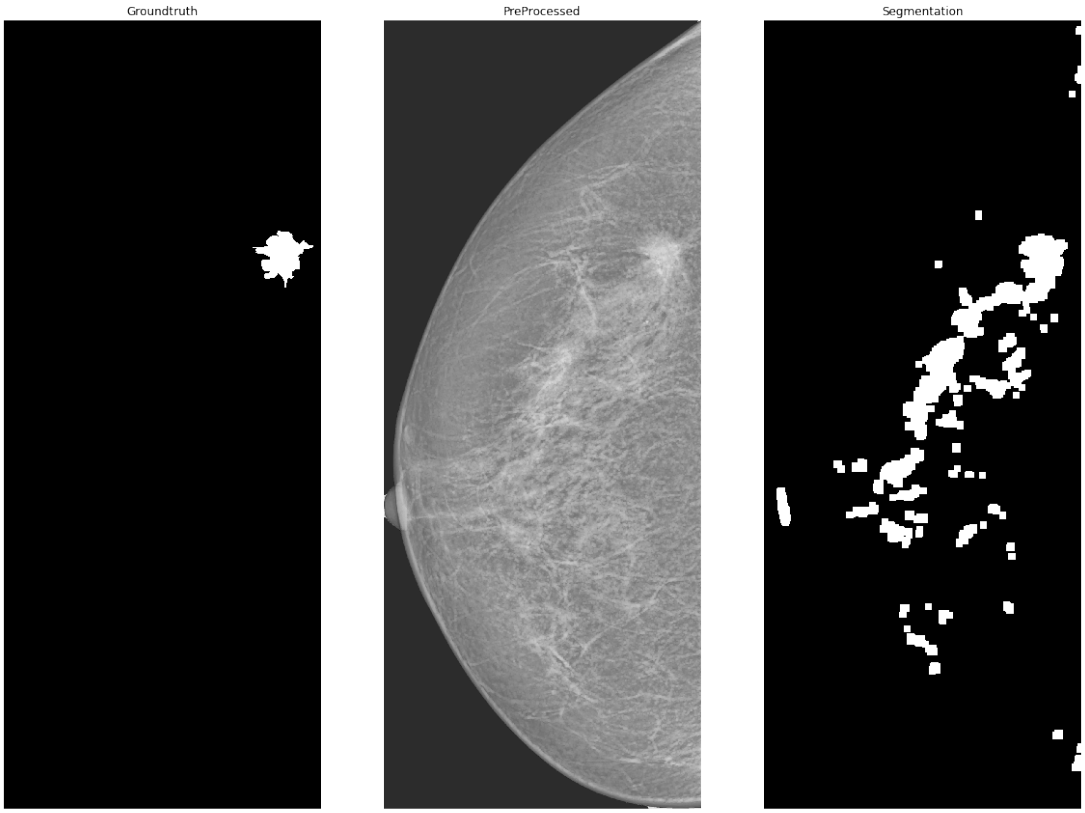


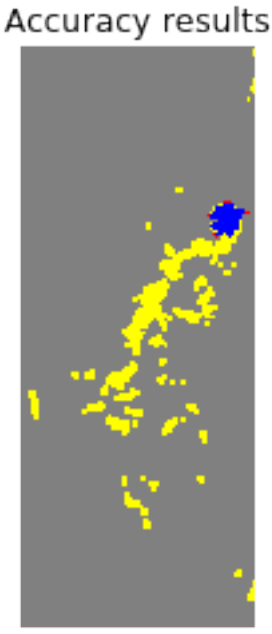

```
0.13411680486786376 (1, 73, 1)
```



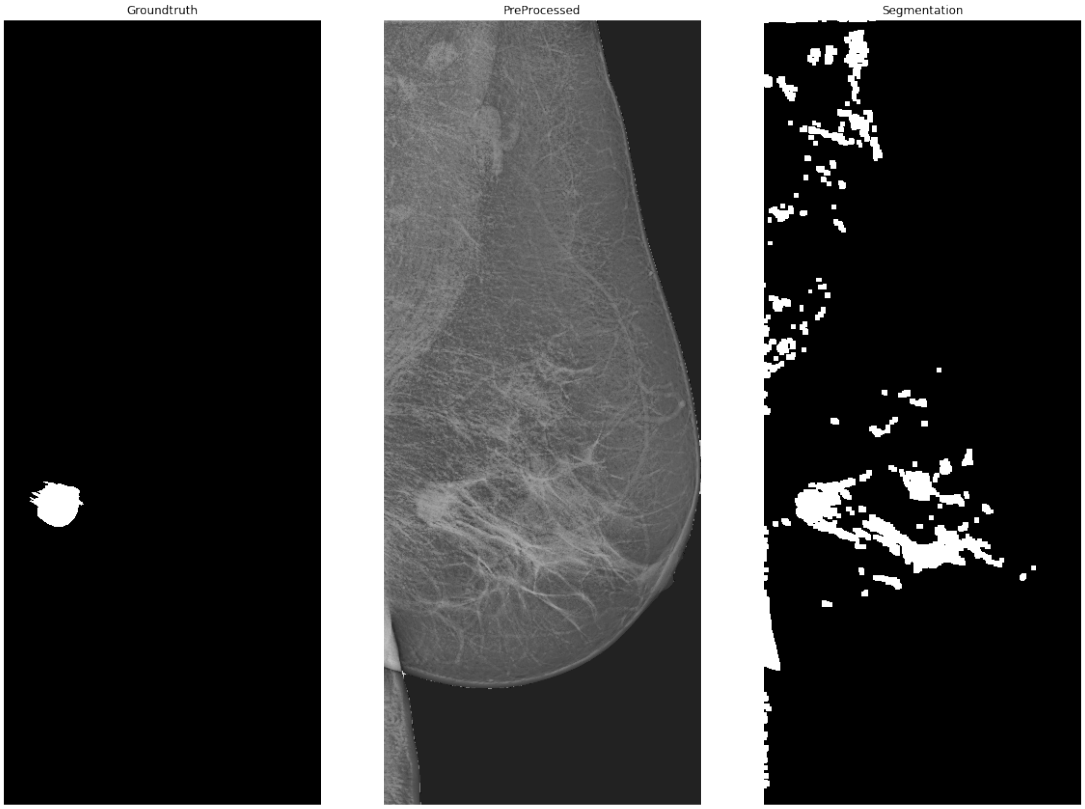


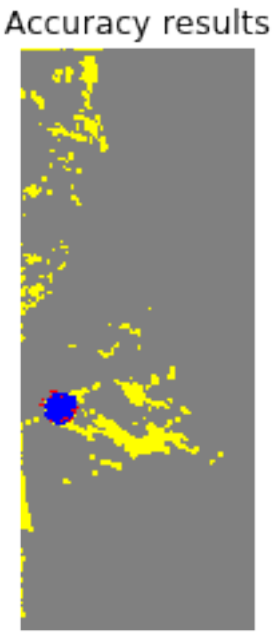

```
0.5329733581844096 (1, 34, 1)
```


Groundtruth
PreProcessed
Segmentation

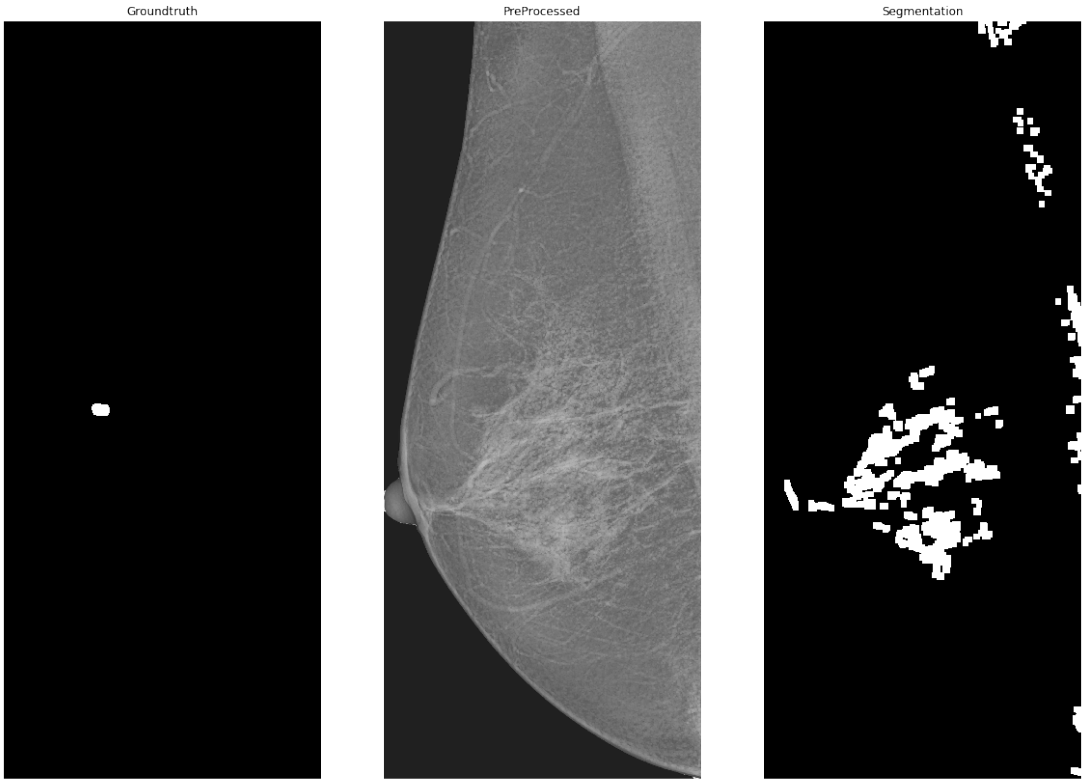

Accuracy results

```
0.20920900496497313 (1, 42, 1)
```



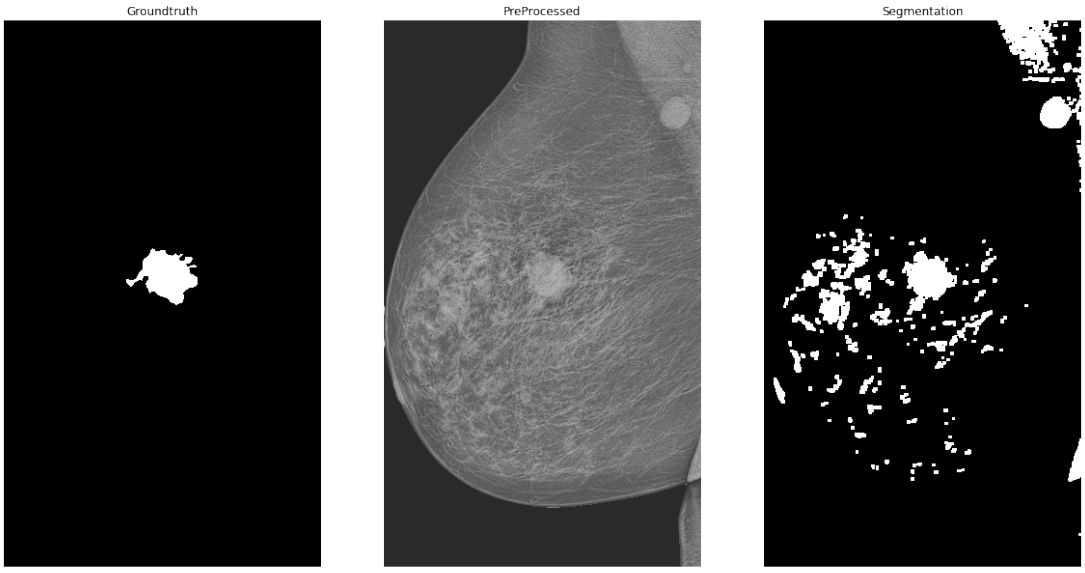


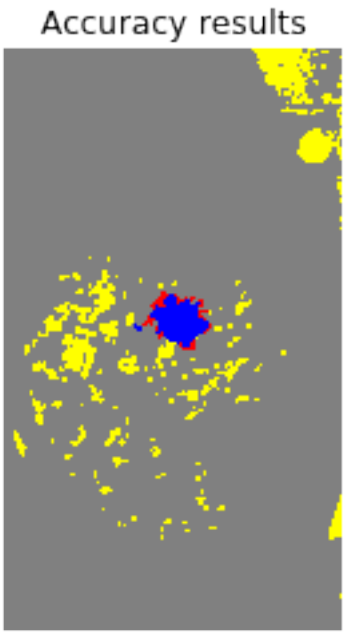
0.5841122886401524 (1, 99, 1)



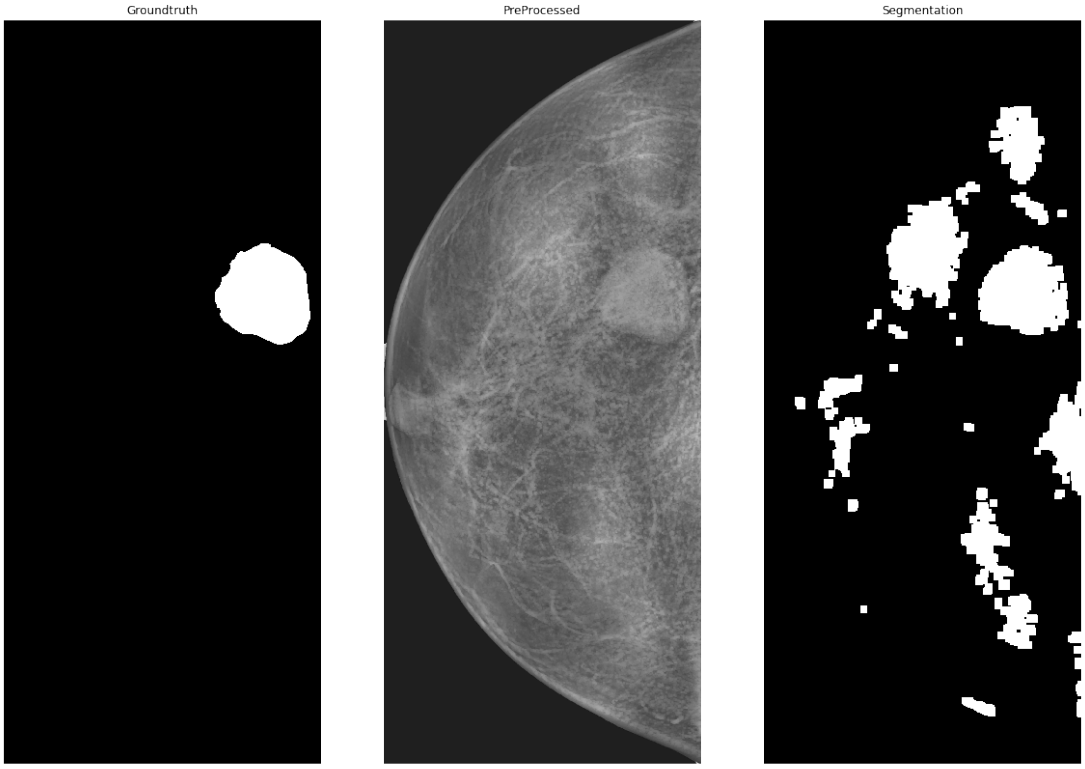

Accuracy results

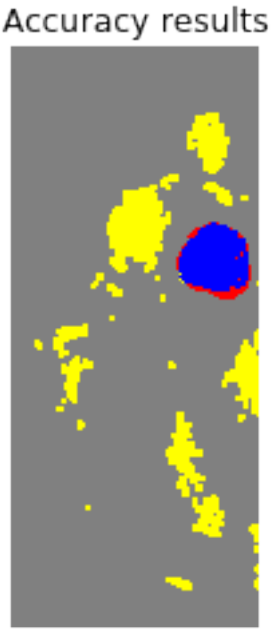
```
0.0 (0, 38, 1)
```


Groundtruth
PreProcessed
Segmentation

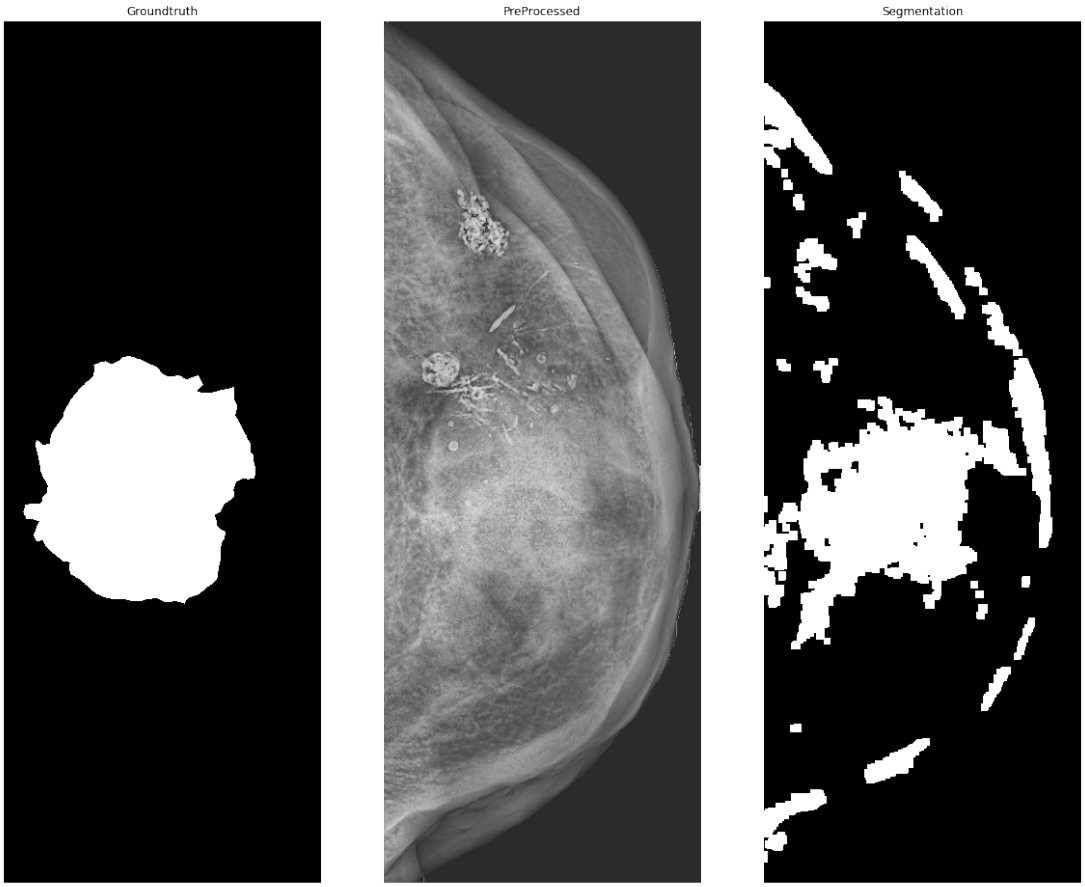


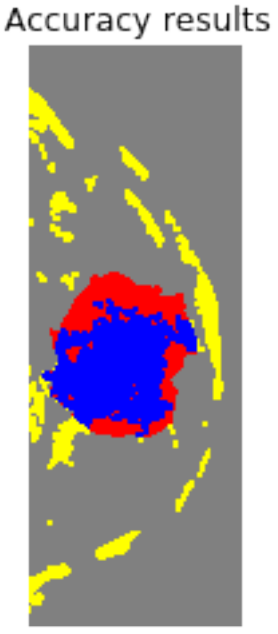

```
0.6883709496520043 (1, 123, 1)
```



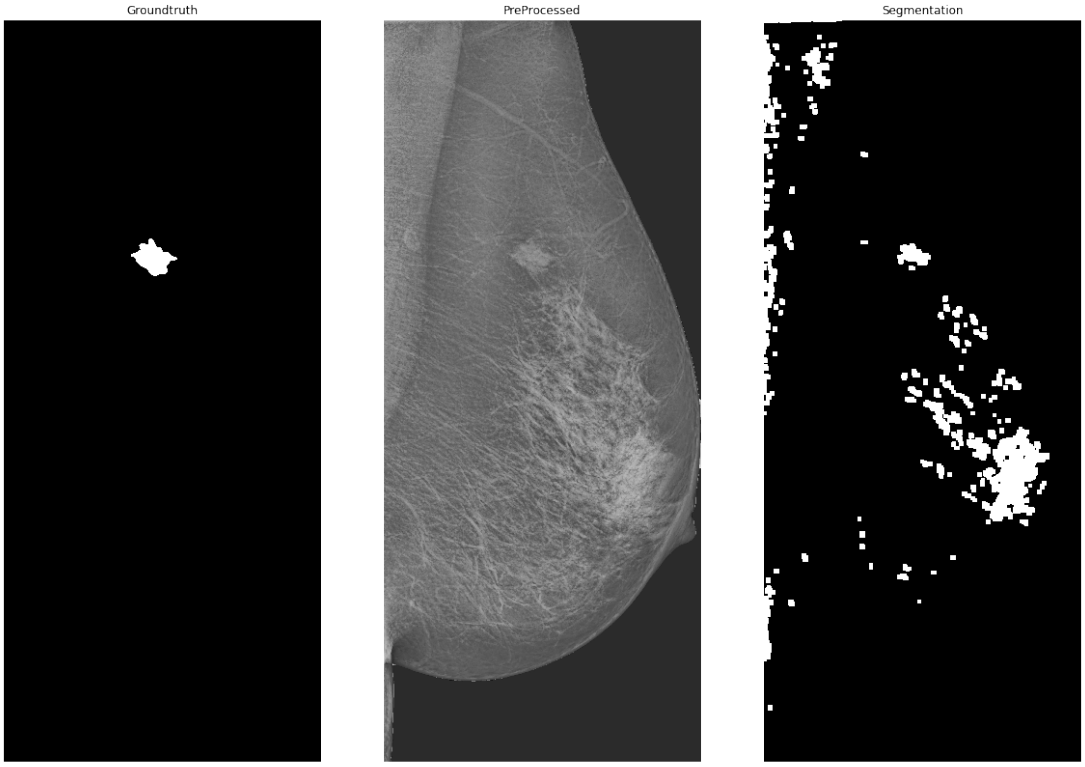


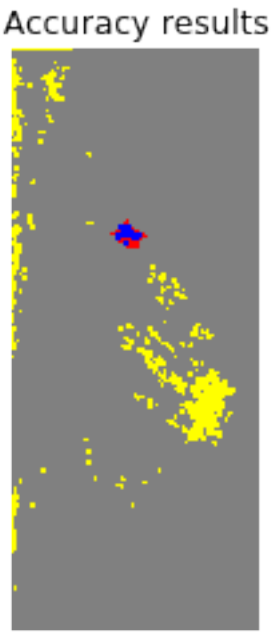

```
0.8464041524497082 (1, 30, 1)
```


Groundtruth
PreProcessed
Segmentation

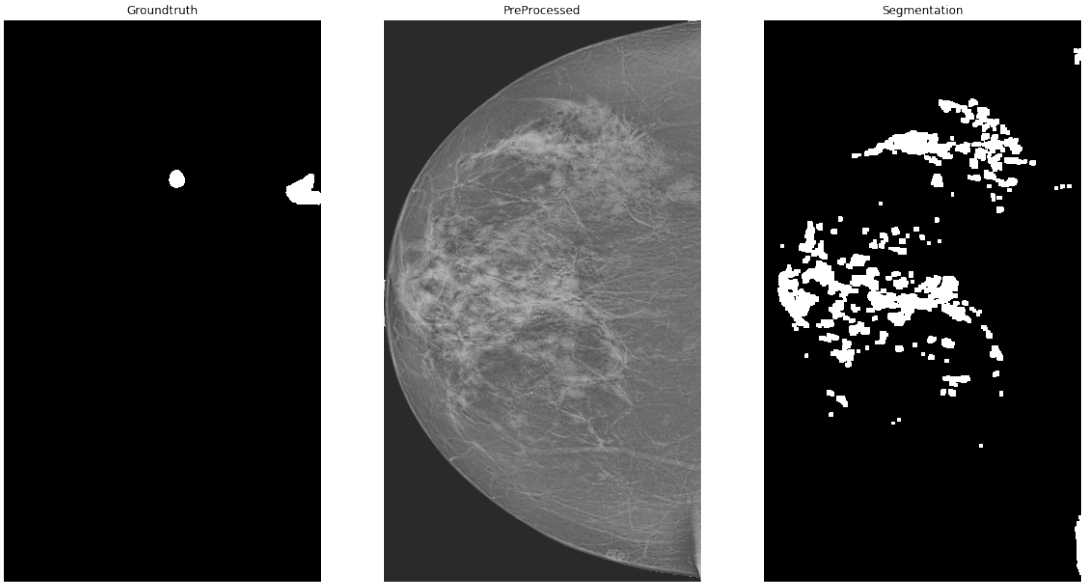

Accuracy results

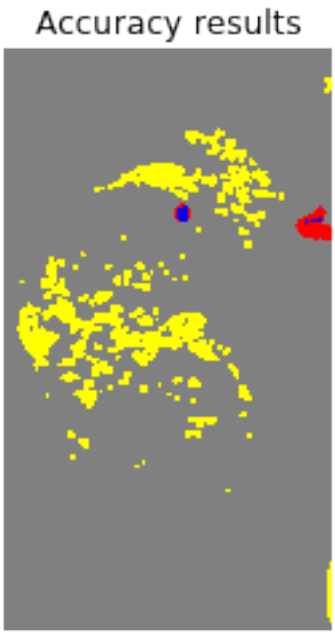
0.6002723251700417 (1, 28, 1)



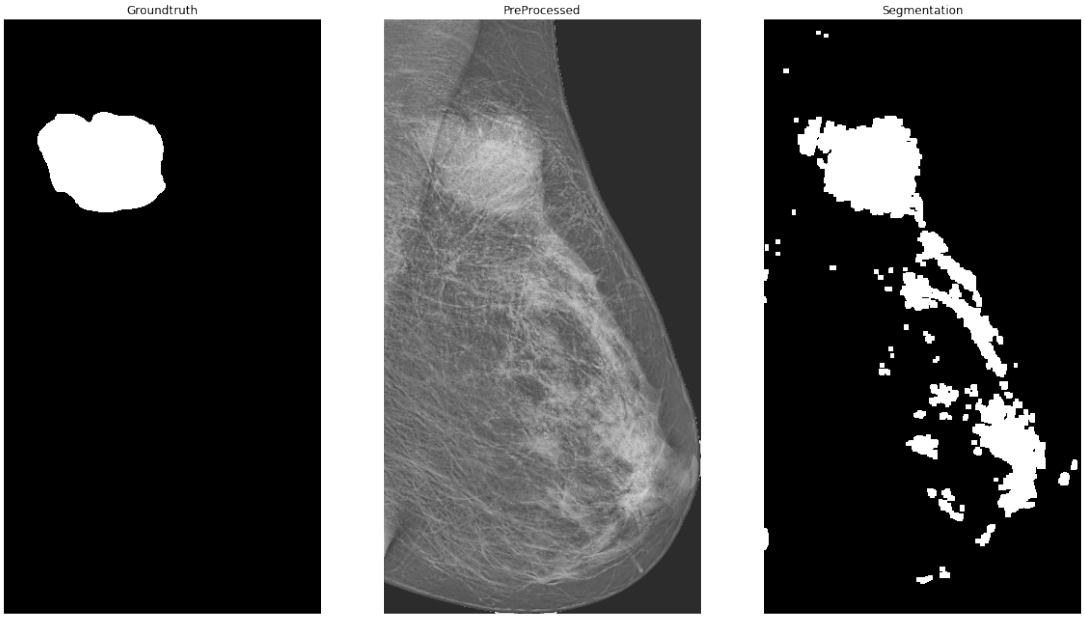


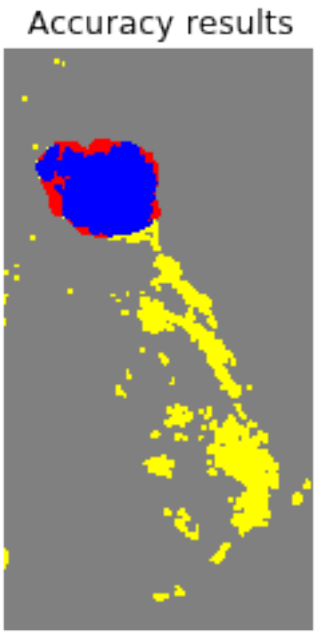
0.5442577451053935 (1, 100, 1)



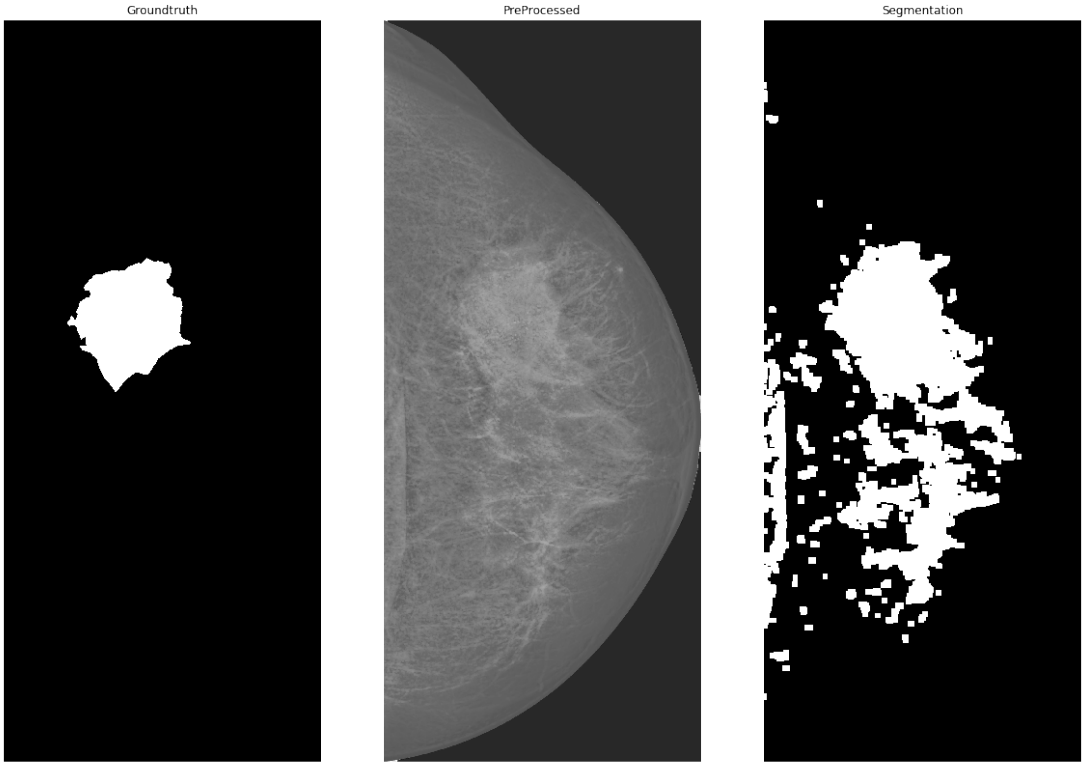


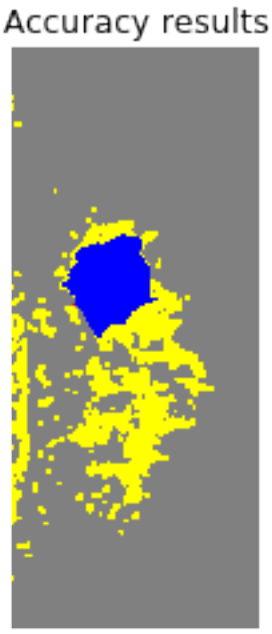

```
0.5909245218965712 (1, 81, 2)
```



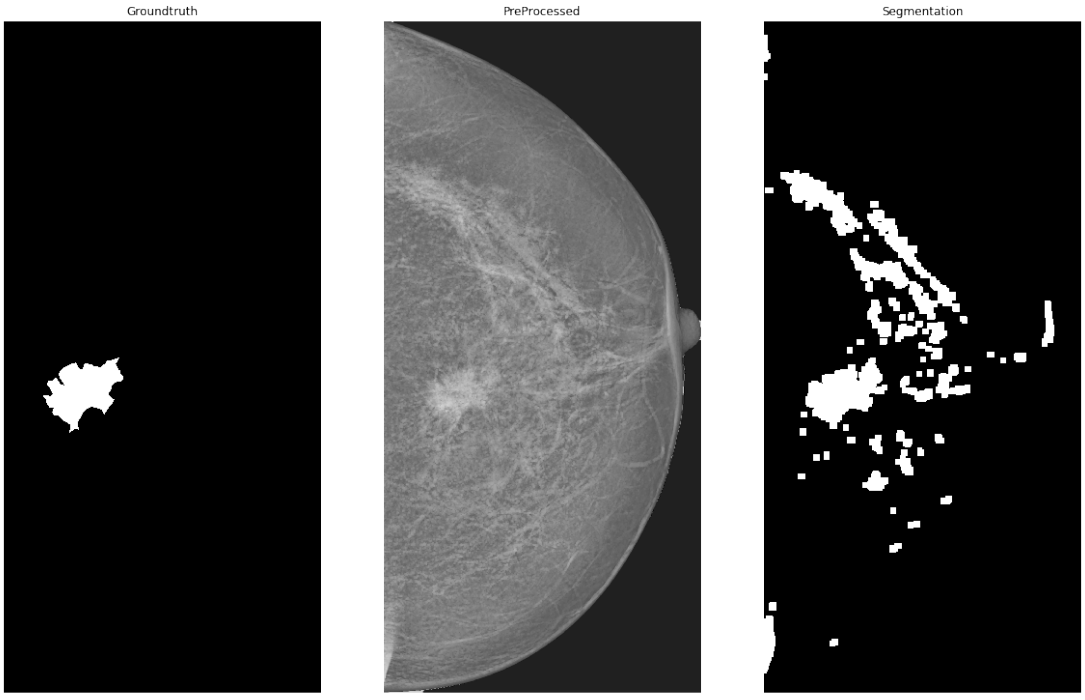


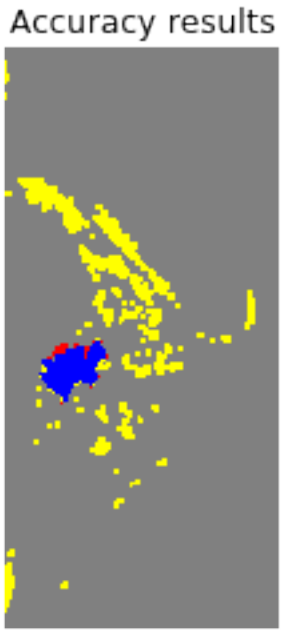
0.771311326218251 (1, 46, 1)



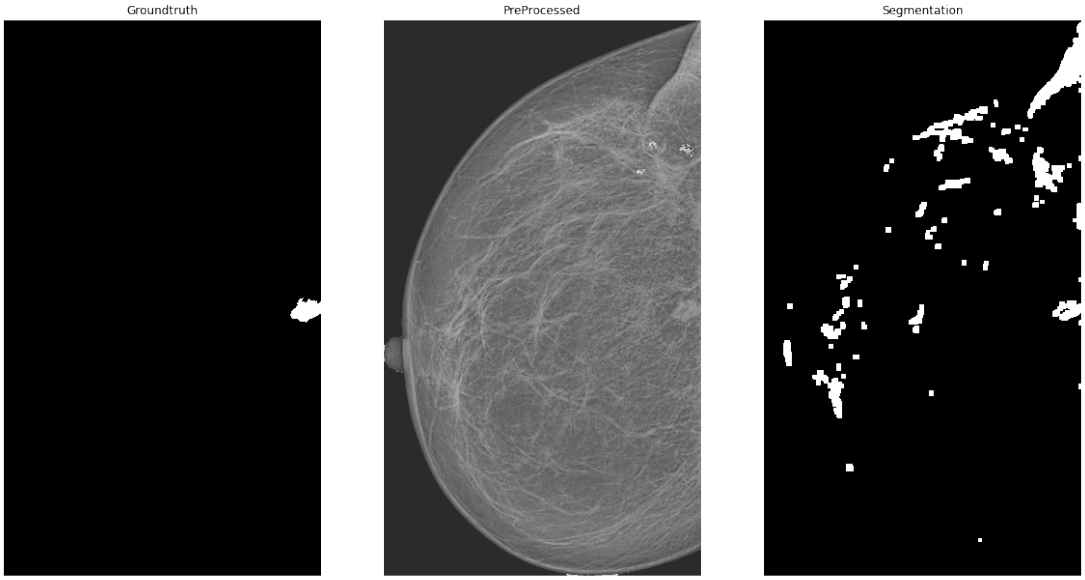


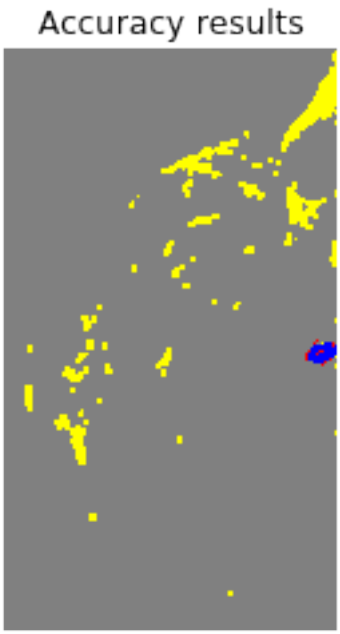

```
0.49216946303369474 (1, 60, 1)
```



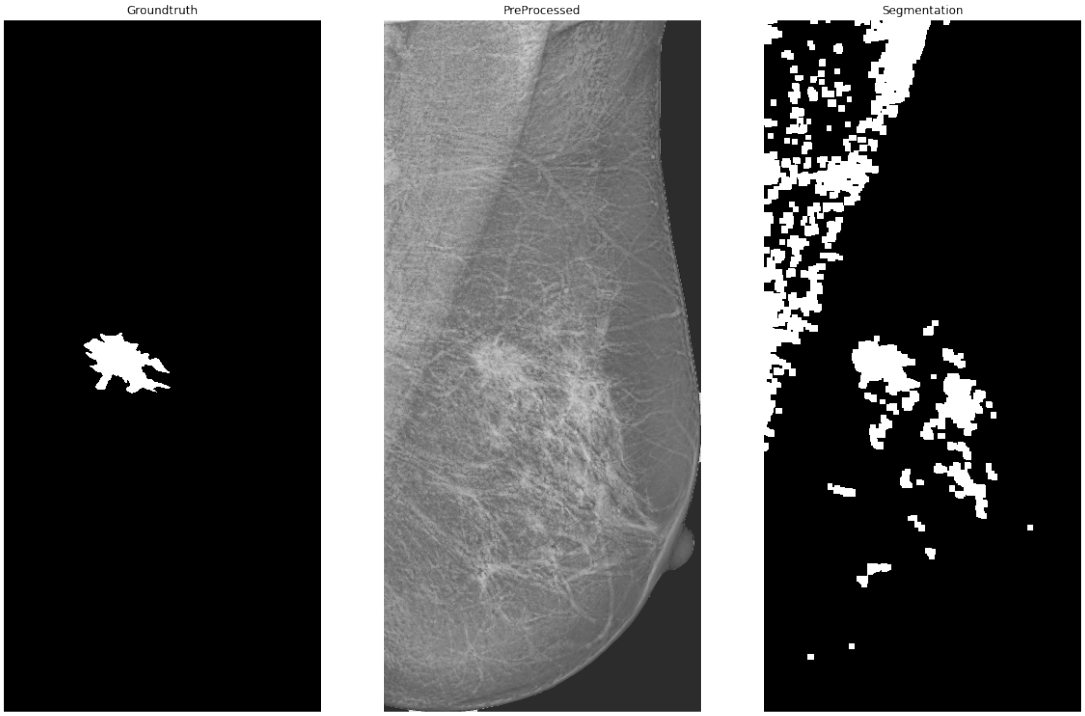


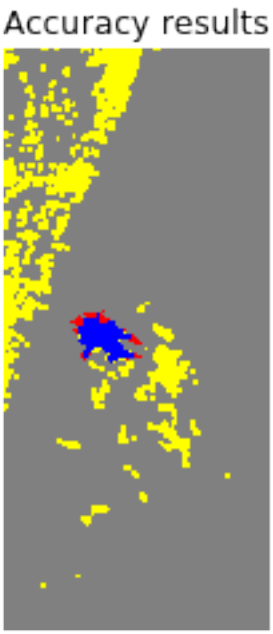

```
0.8291492201641952 (1, 48, 1)
```



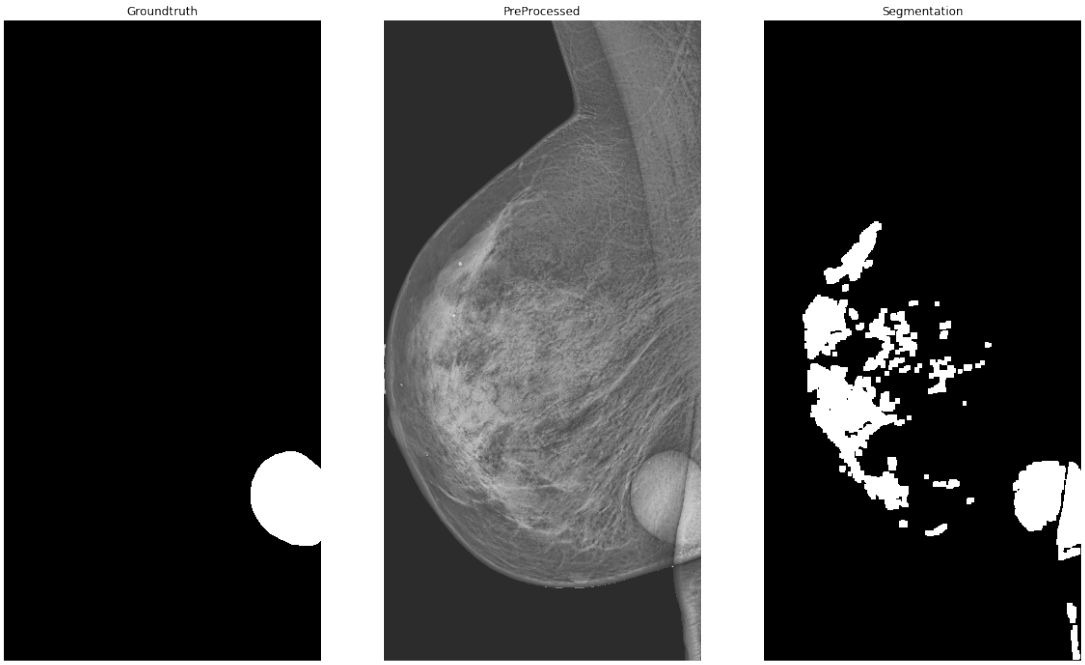


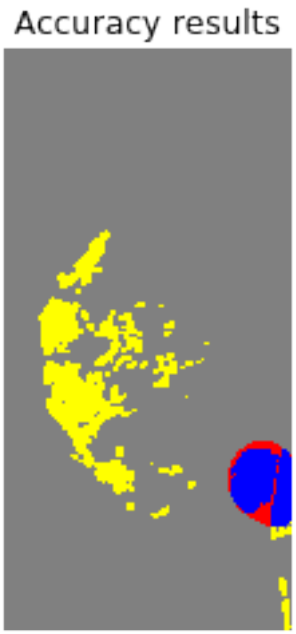

```
0.8461990883693574 (1, 50, 1)
```



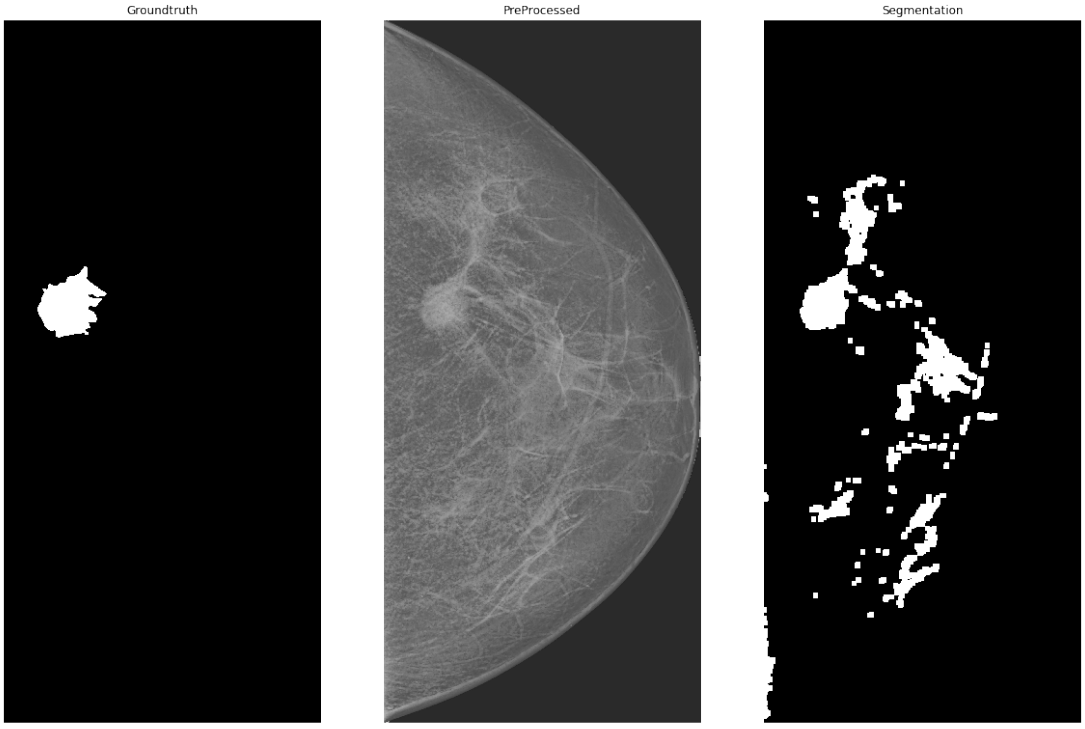


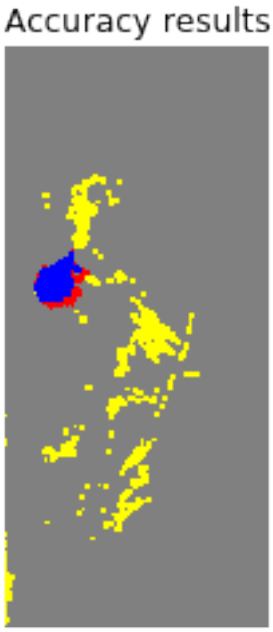

```
0.6987379982542915 (1, 69, 1)
```



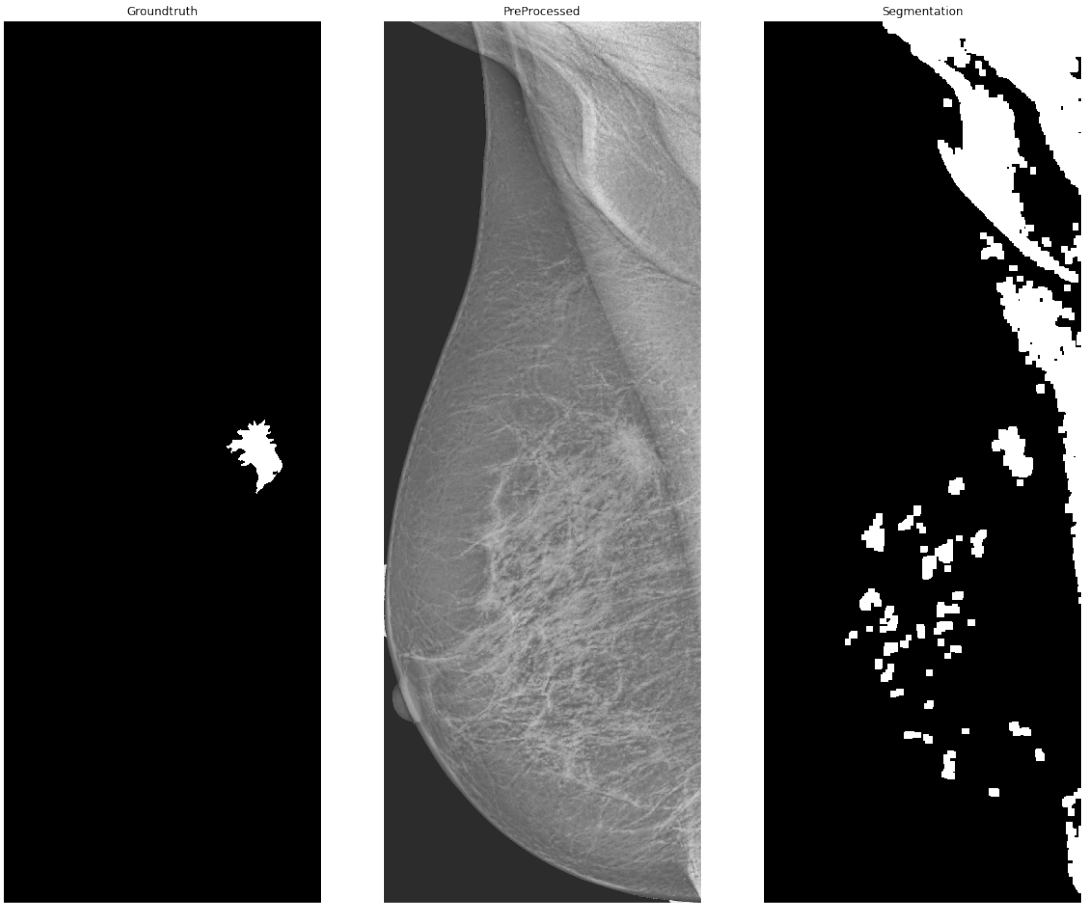


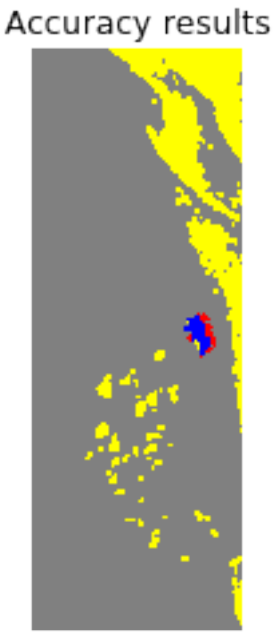

```
0.759915320420558 (1, 33, 1)
```



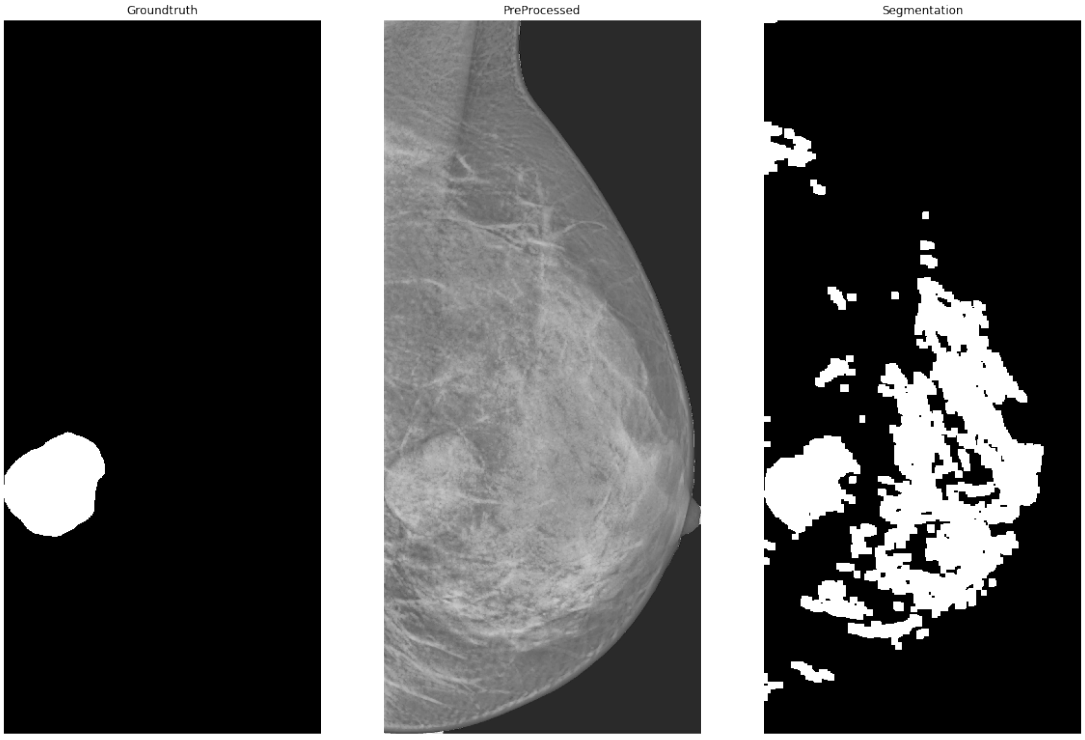


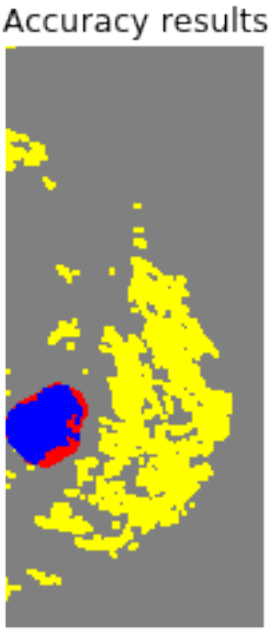

```
0.7431604357514237 (1, 52, 1)
```


Groundtruth
PreProcessed
Segmentation

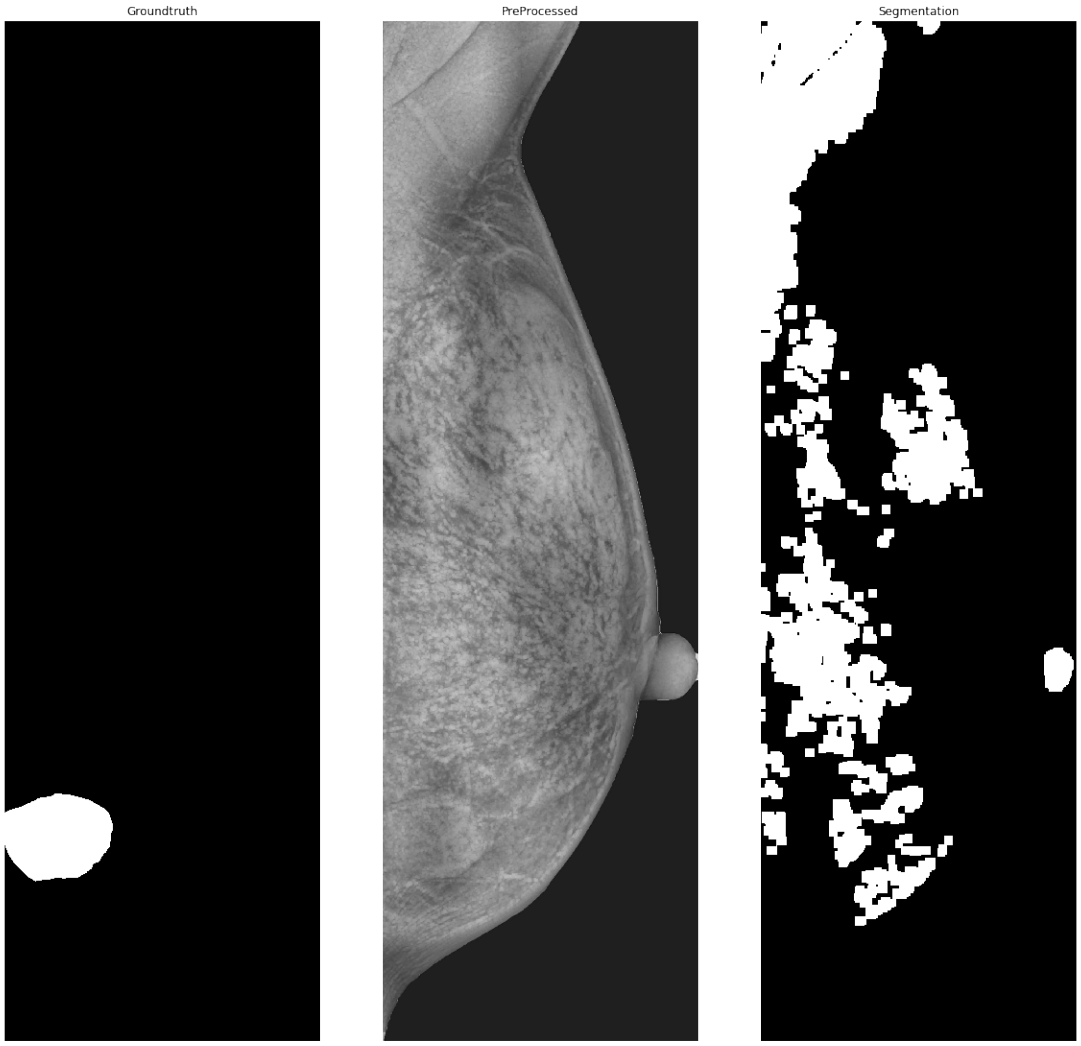



```
0.5678457308463997 (1, 52, 1)
```

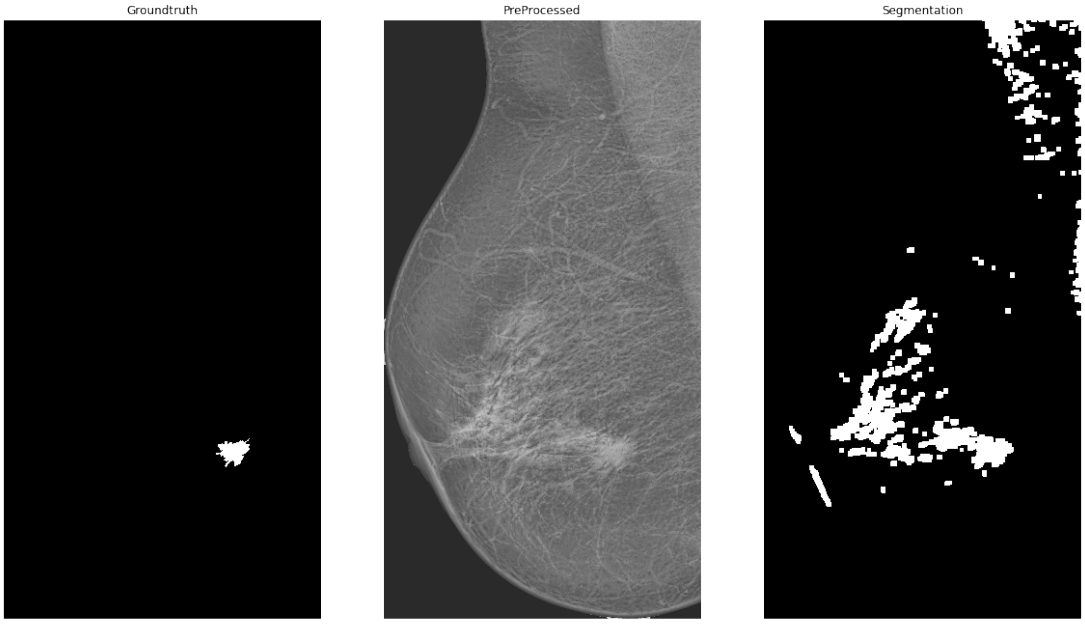


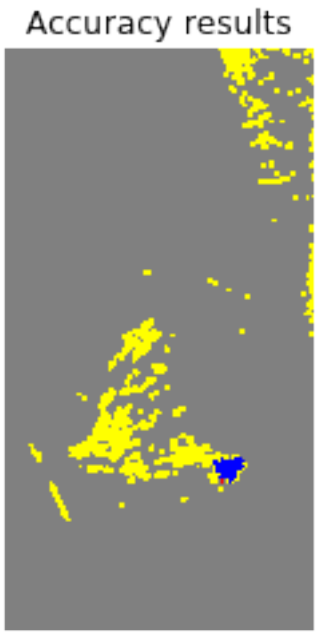



```
0.7662494241268784 (1, 27, 1)
```

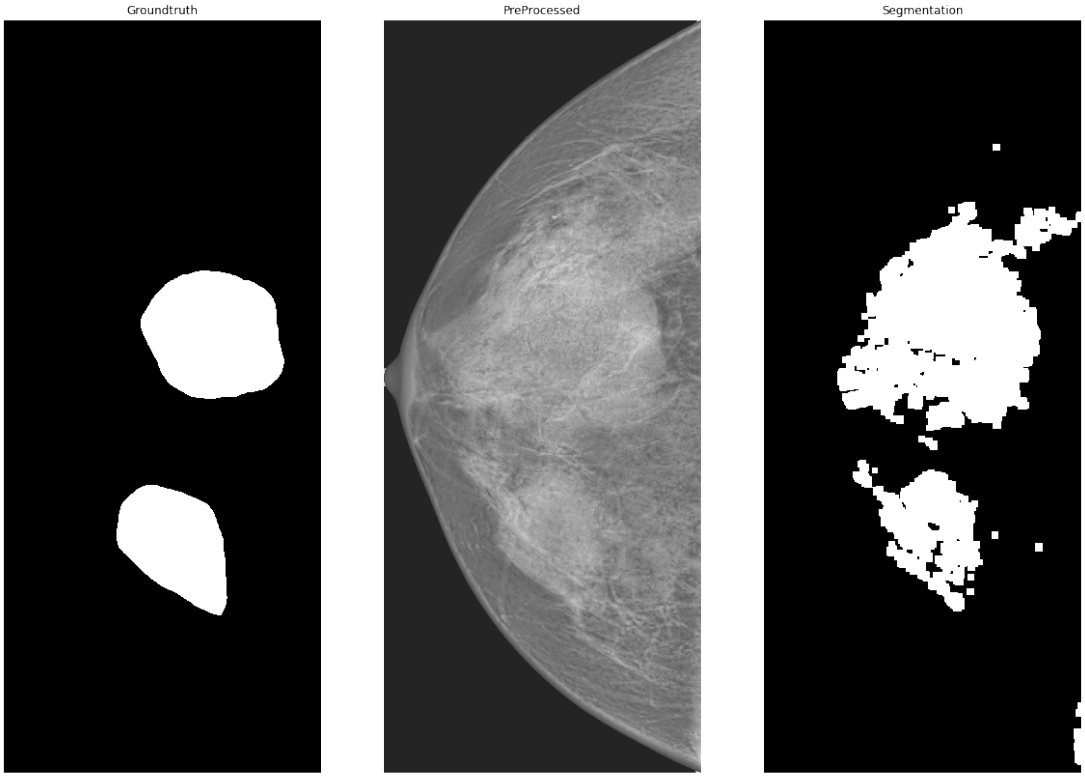

Groundtruth
PreProcessed
Segmentation

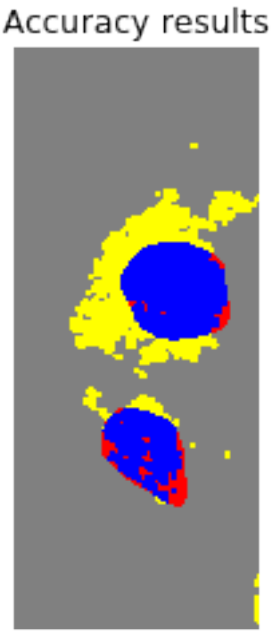

```
0.20375704269877623 (1, 31, 1)
```



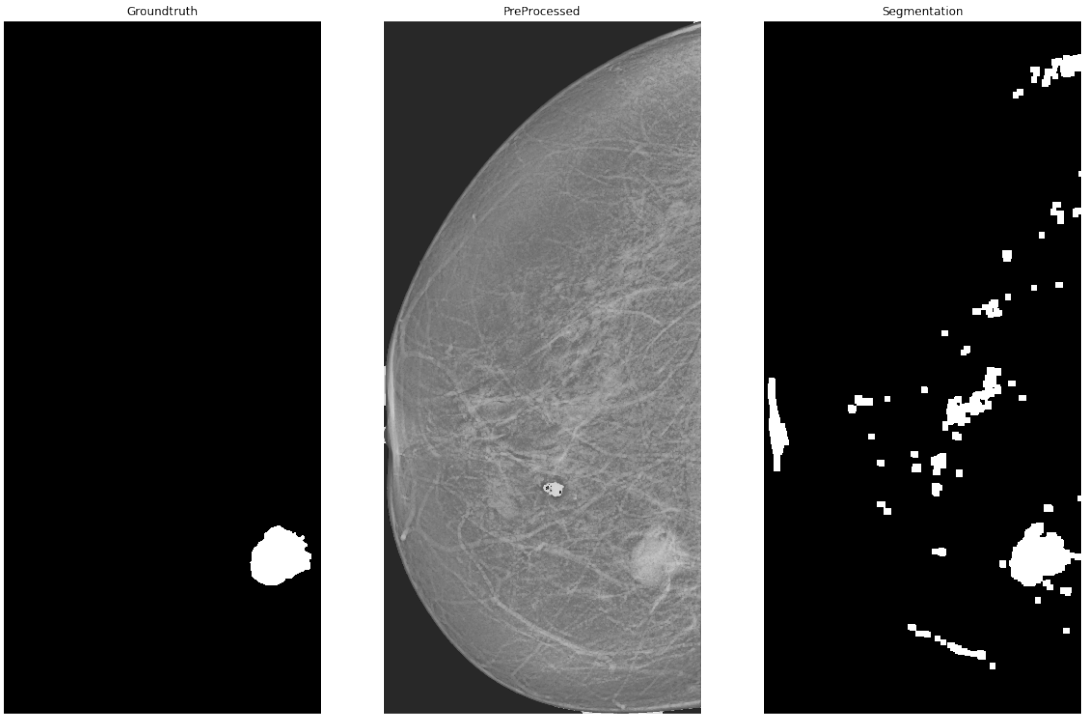


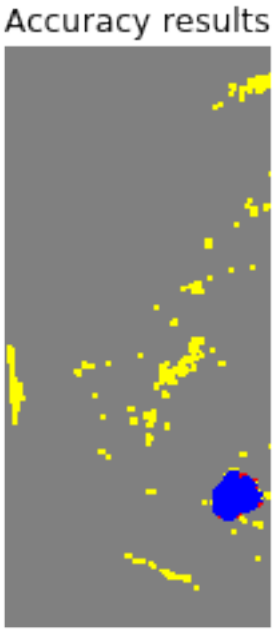

```
0.25044989366149817 (1, 74, 1)
```



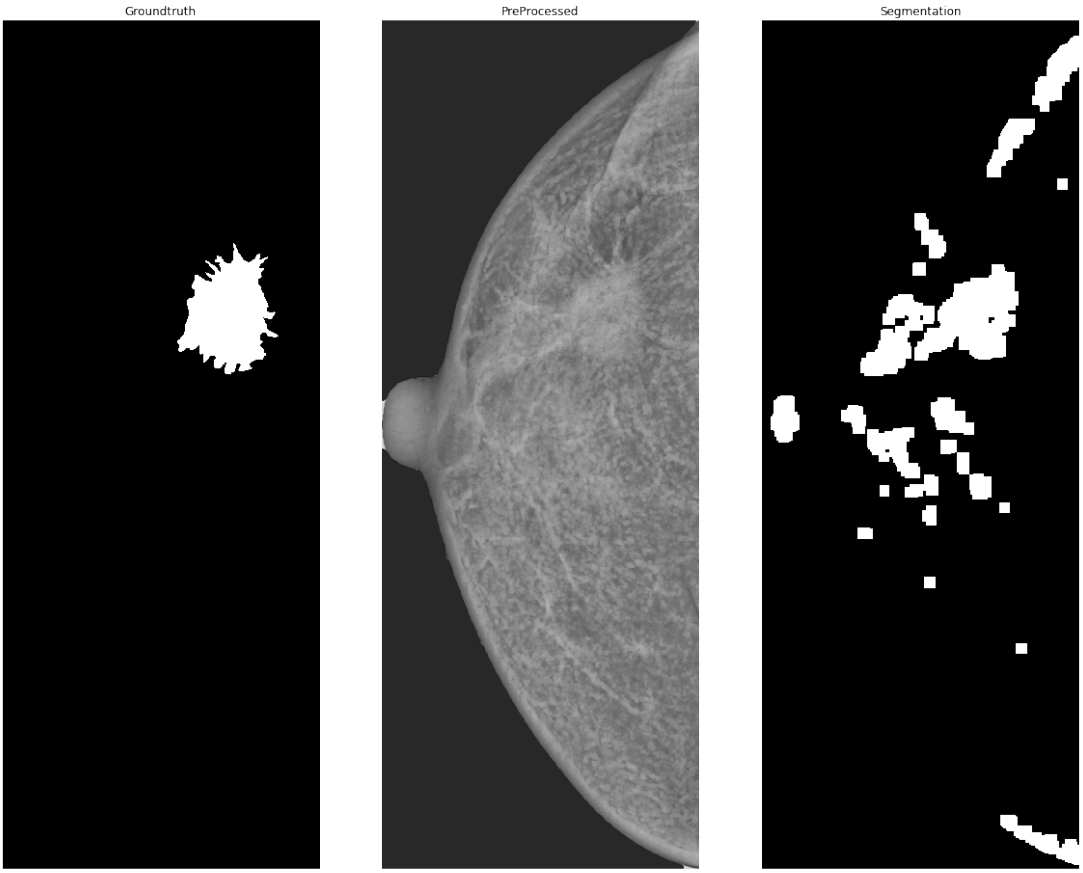


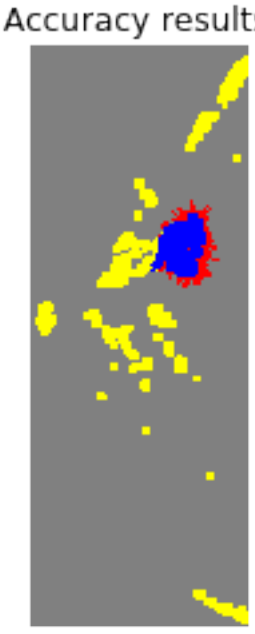

```
0.6900135093006339 (2, 10, 2)
```



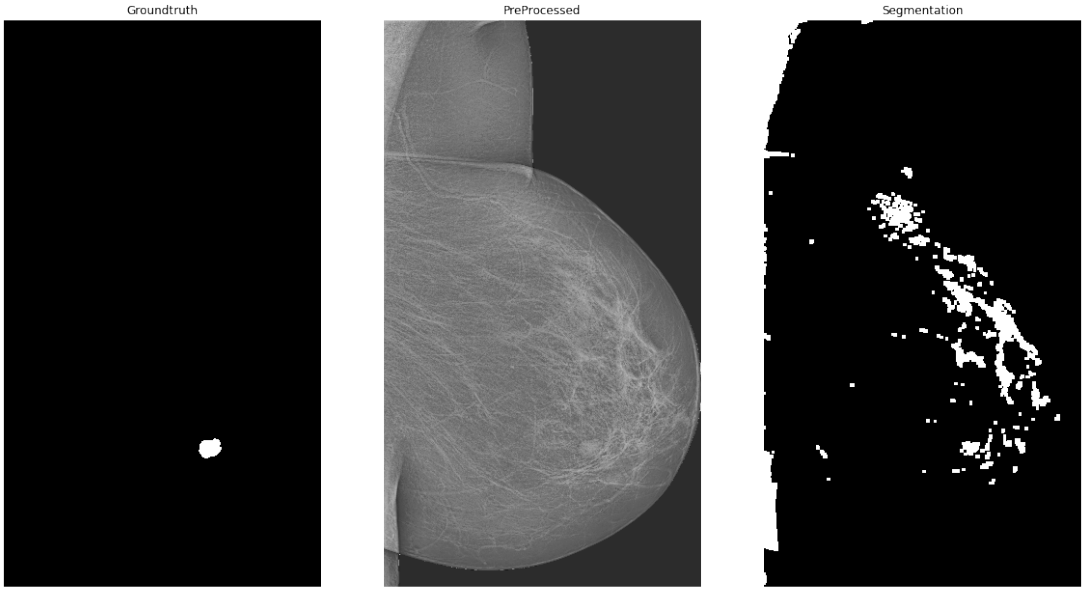


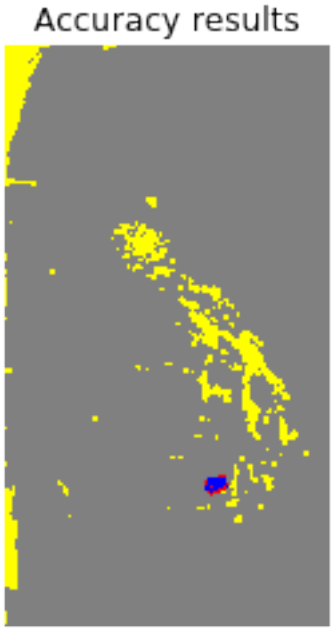

```
0.8547017849368742 (1, 42, 1)
```


Groundtruth
PreProcessed
Segmentation

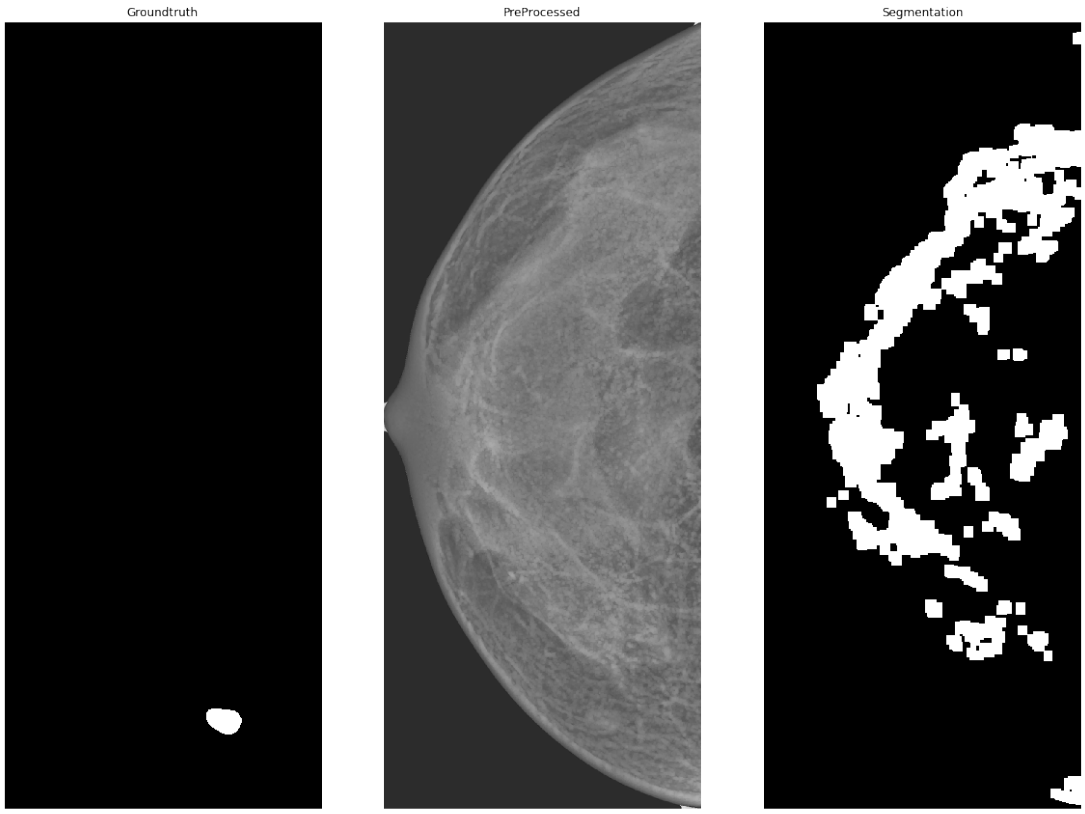

Accuracy results

```
0.6118479439897446 (1, 20, 1)
```

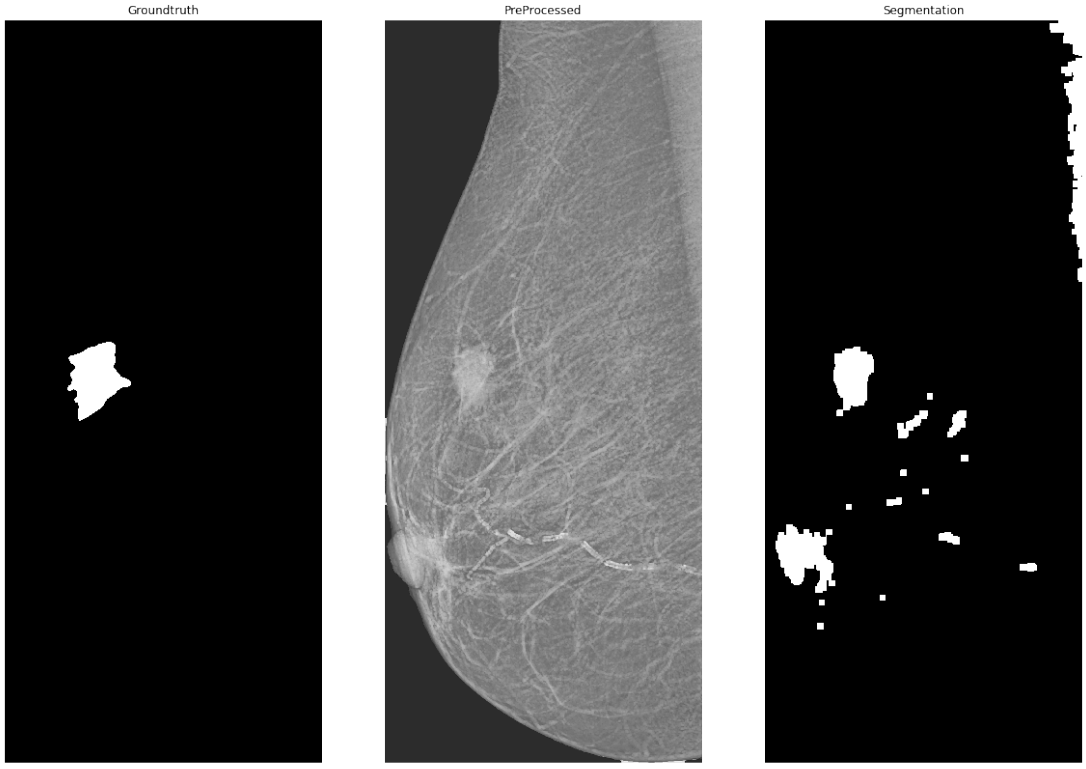


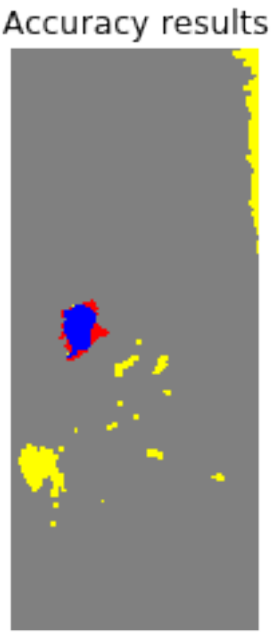


```
0.6271399942742628 (1, 95, 1)
```

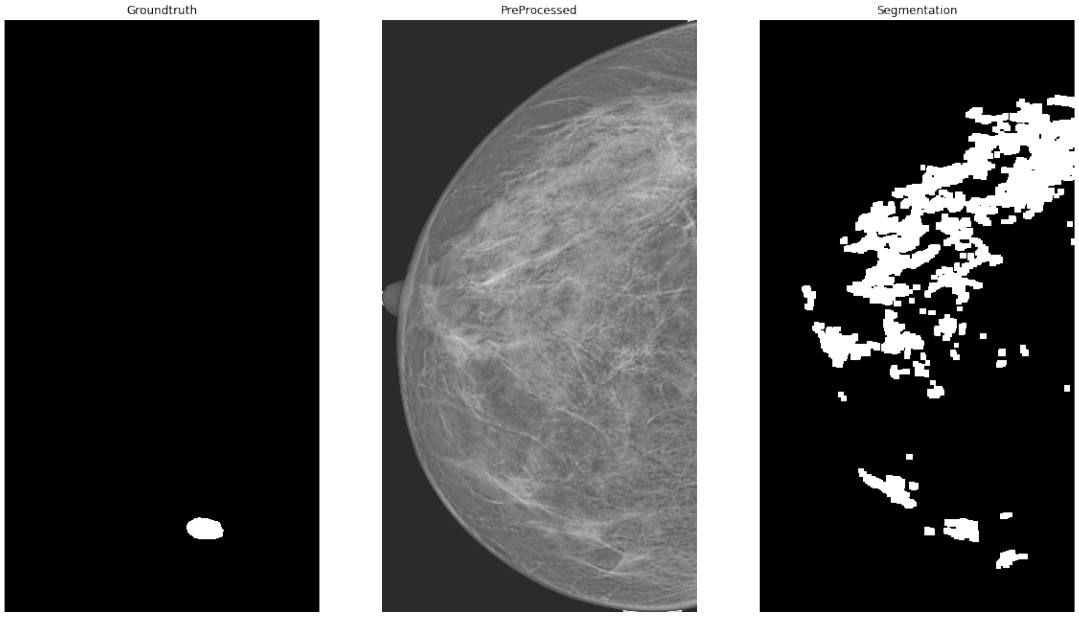


```
0.0 (0, 21, 1)
```

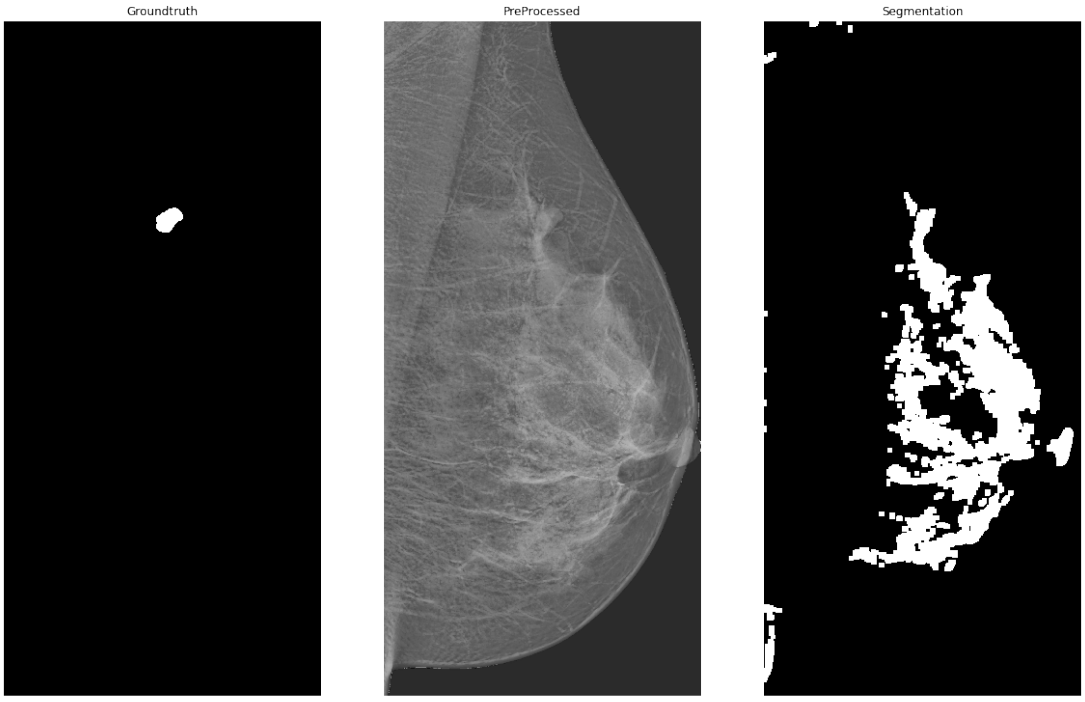


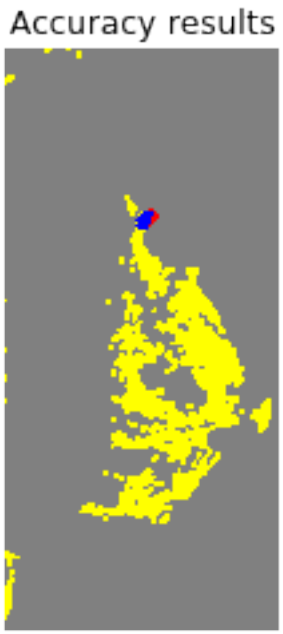



```
0.6729035114290375 (1, 17, 1)
```

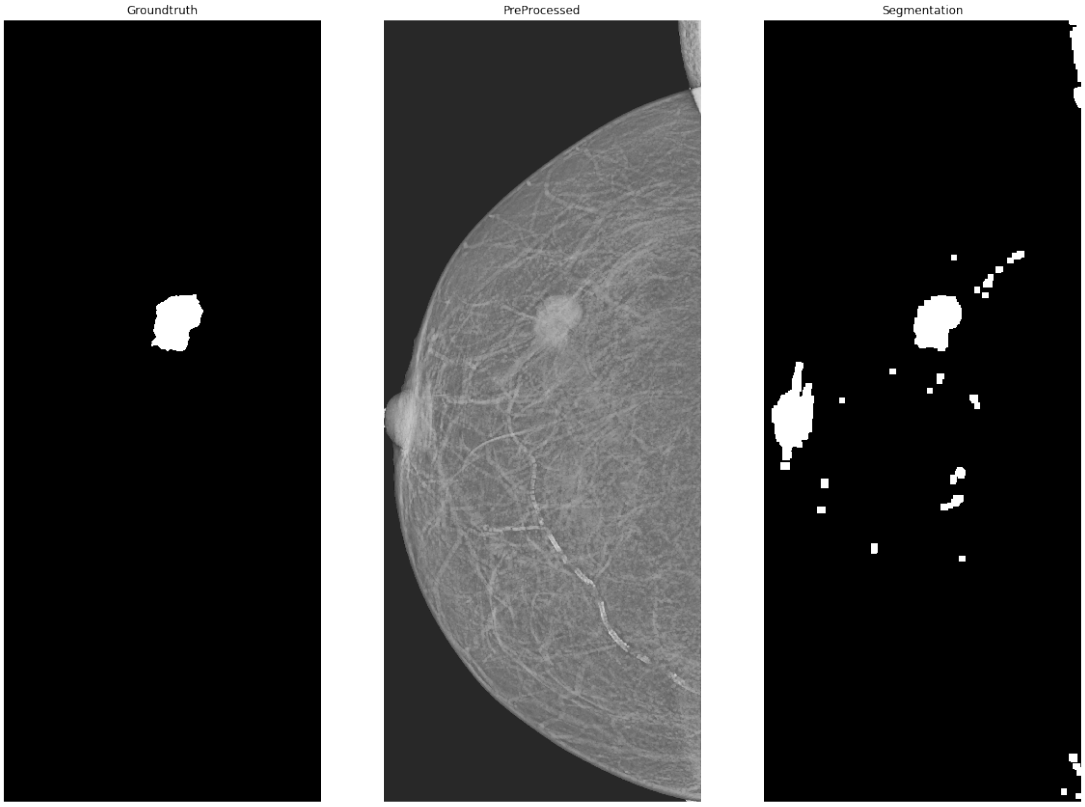


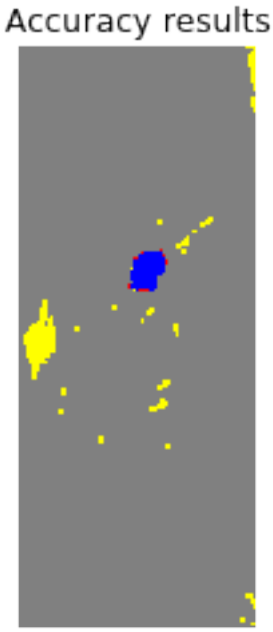
0.8216235129461161 (1, 35, 1)



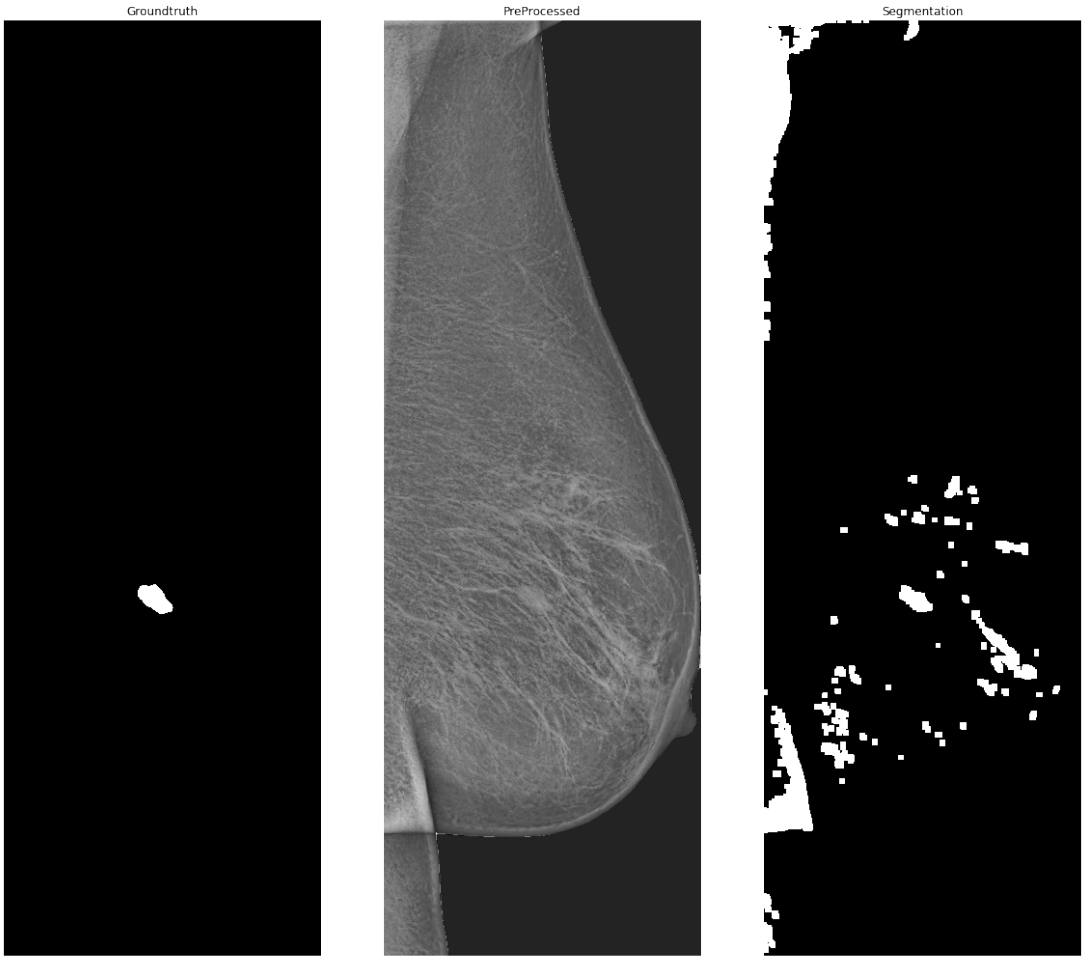


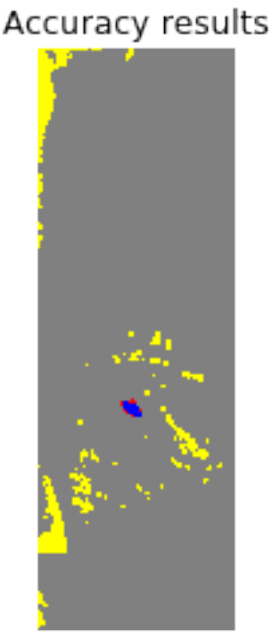

```
0.14219880325087078 (1, 35, 1)
```



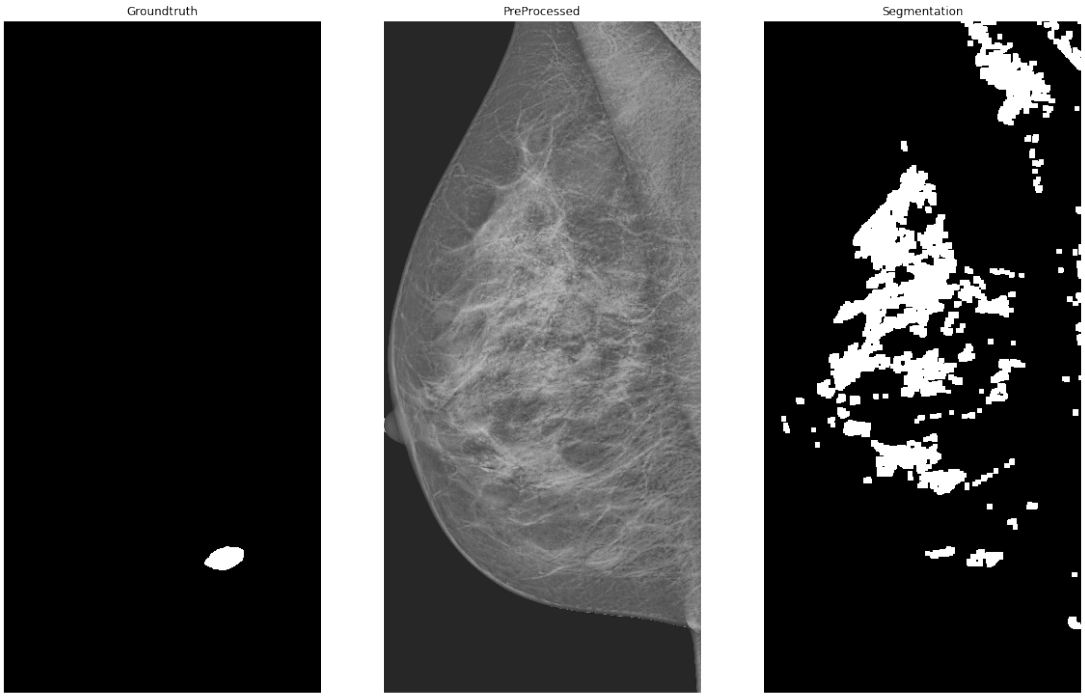



```
0.9205560467030455 (1, 25, 1)
```

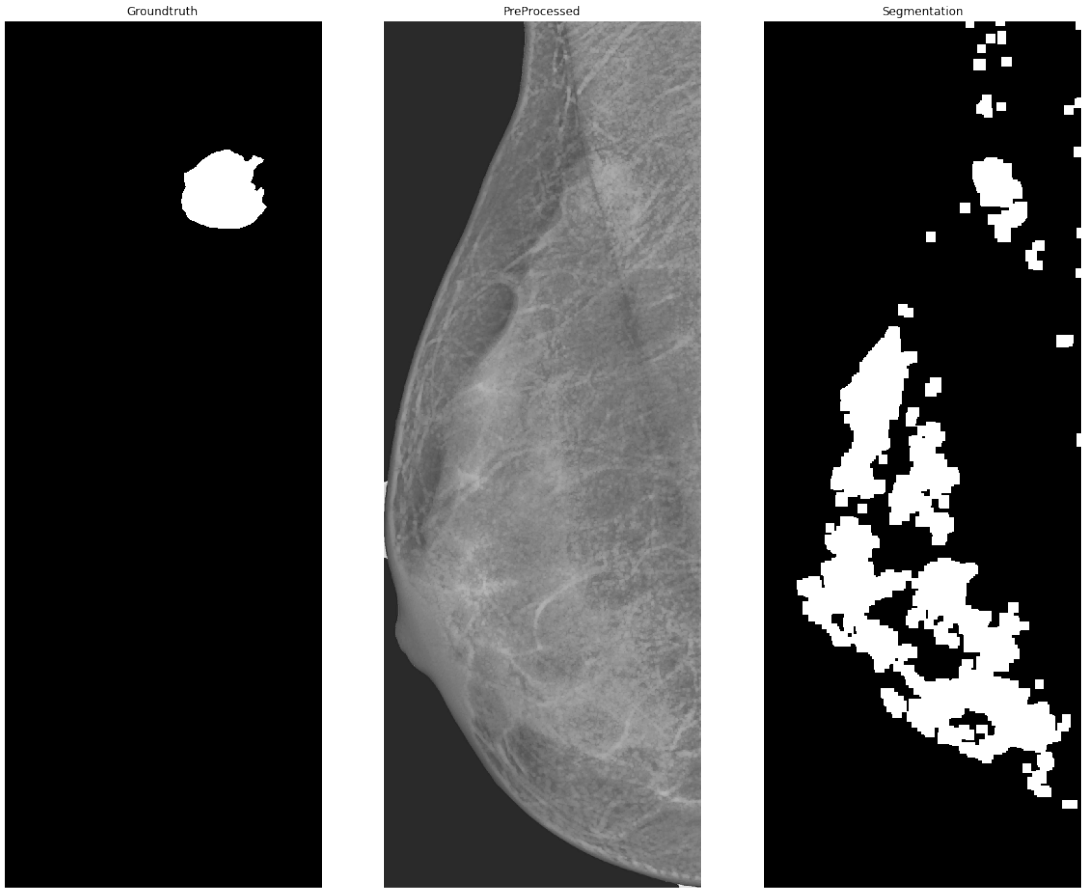

Groundtruth
PreProcessed
Segmentation




```
0.7948717948717948 (1, 58, 1)
```

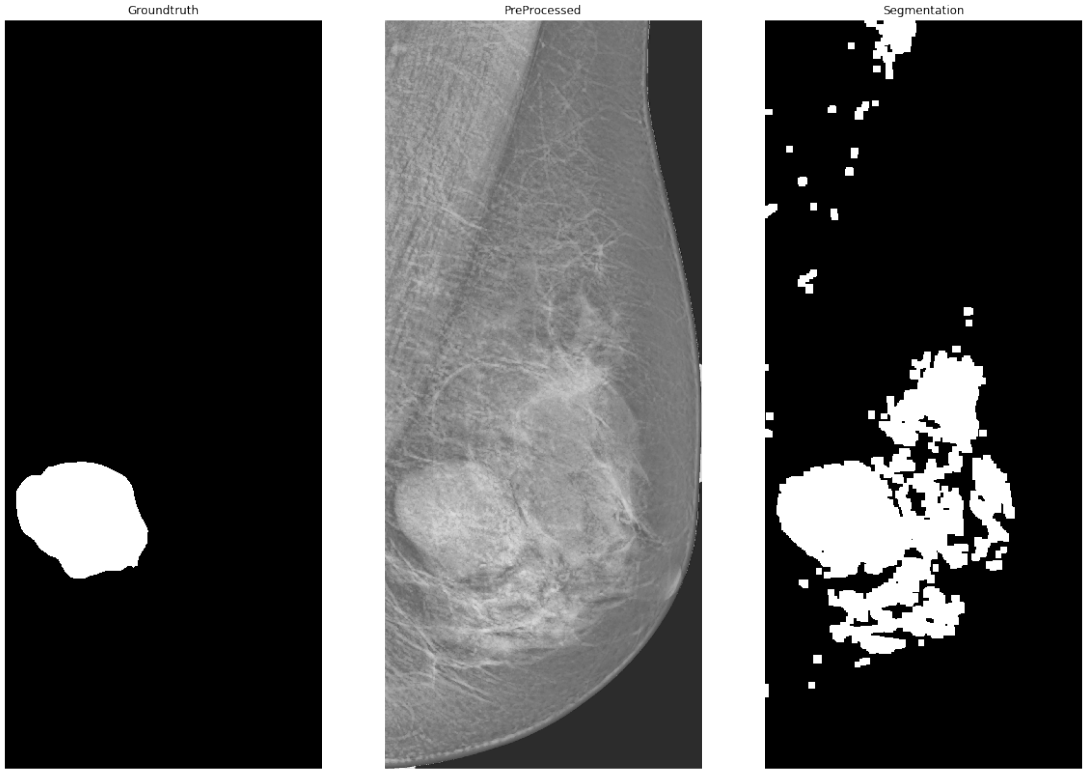


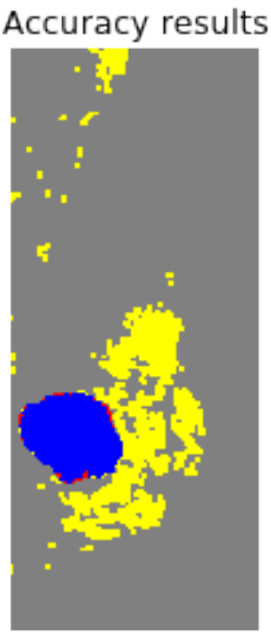

```
0.6630944407233758 (1, 70, 1)
```


Groundtruth
PreProcessed
Segmentation

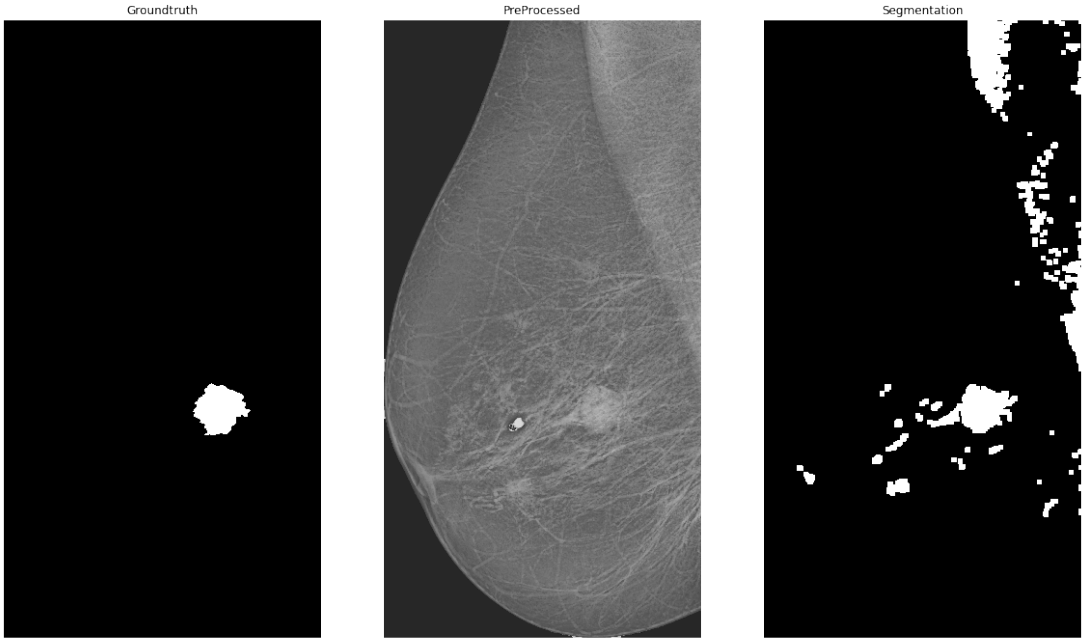

Accuracy results

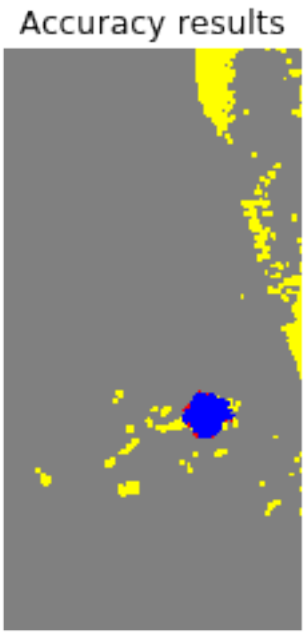
```
0.41785646811152194 (1, 27, 1)
```



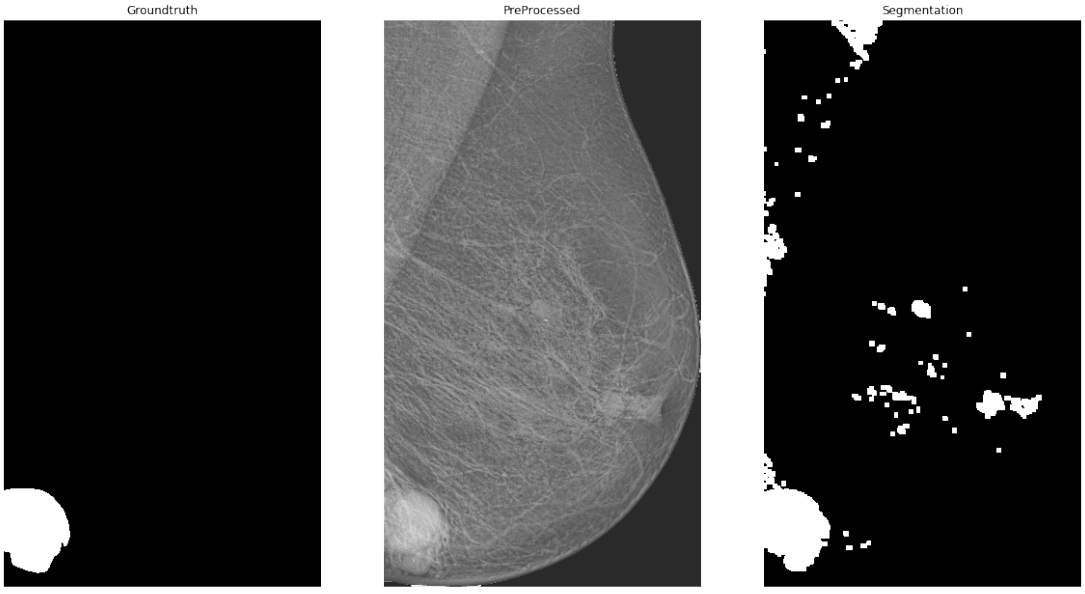



```
0.34997642338060986 (1, 34, 1)
```

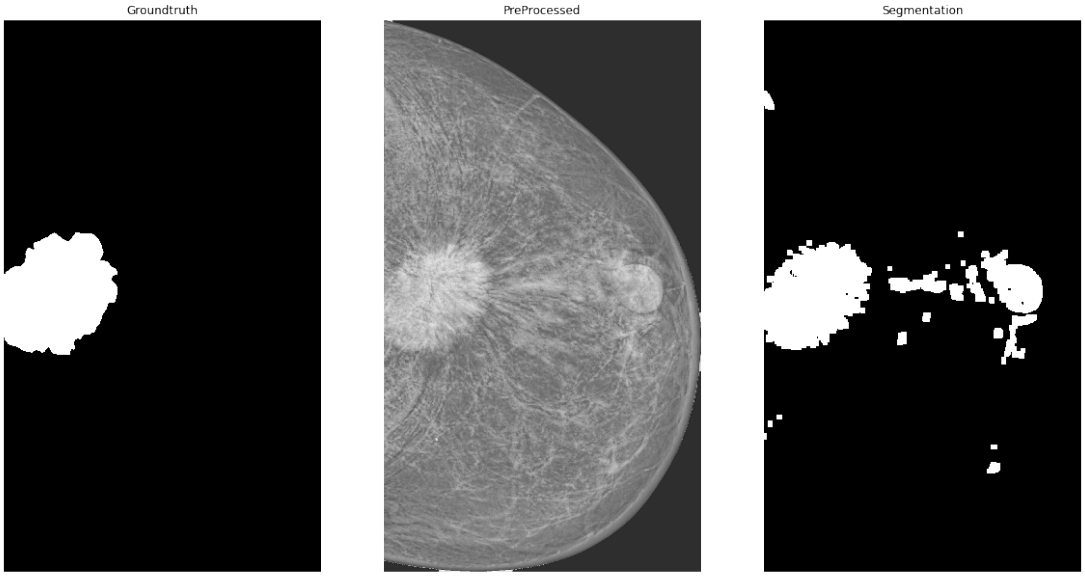


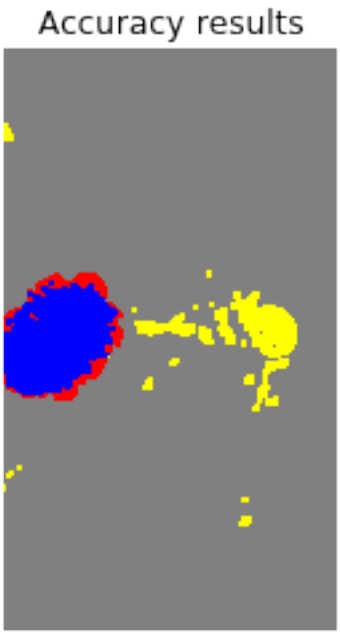


0.7279137429402704 (1, 55, 1)

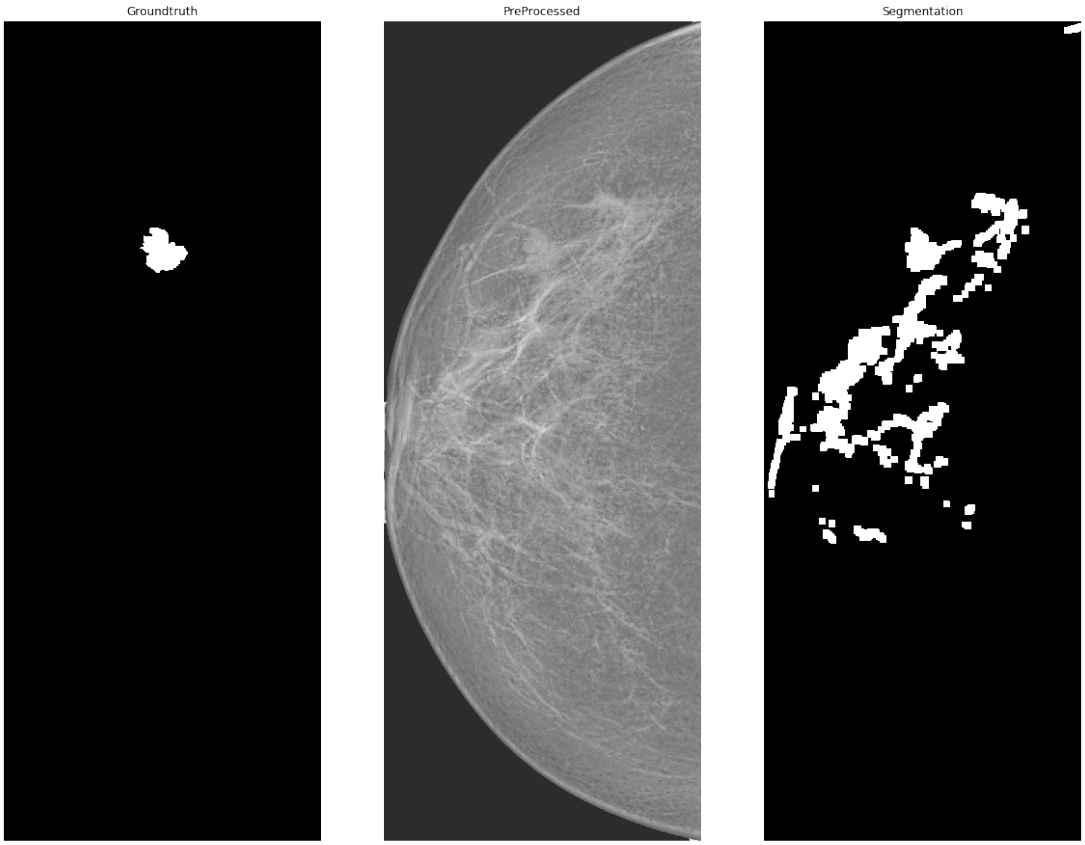


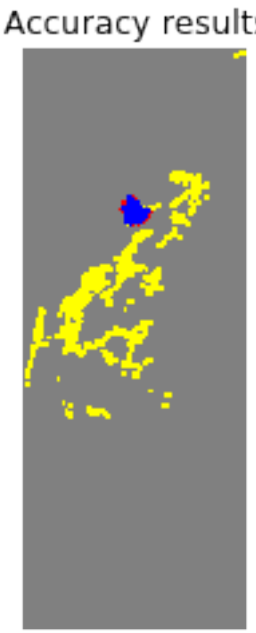

```
0.8639848042873618 (1, 53, 1)
```



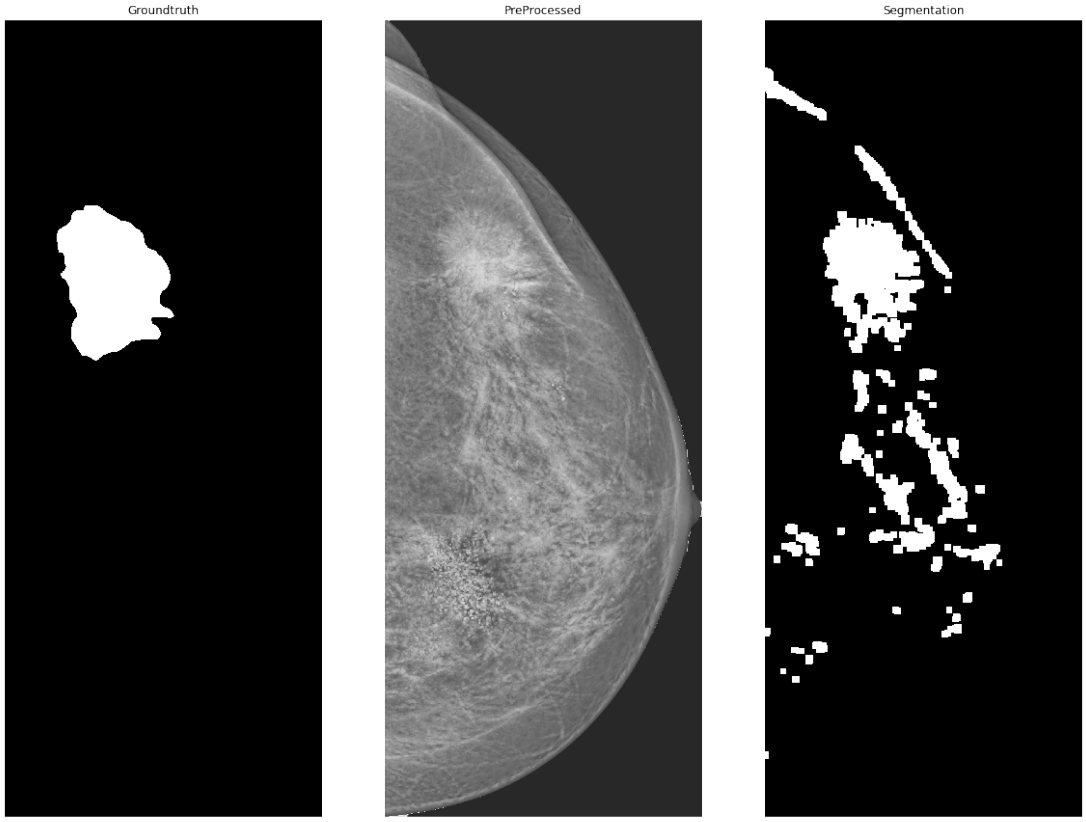


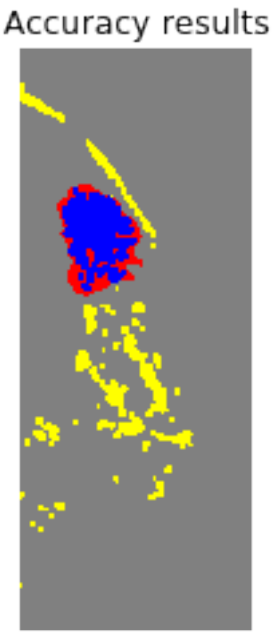

```
0.8124959893906013 (1, 19, 1)
```


Groundtruth
PreProcessed
Segmentation

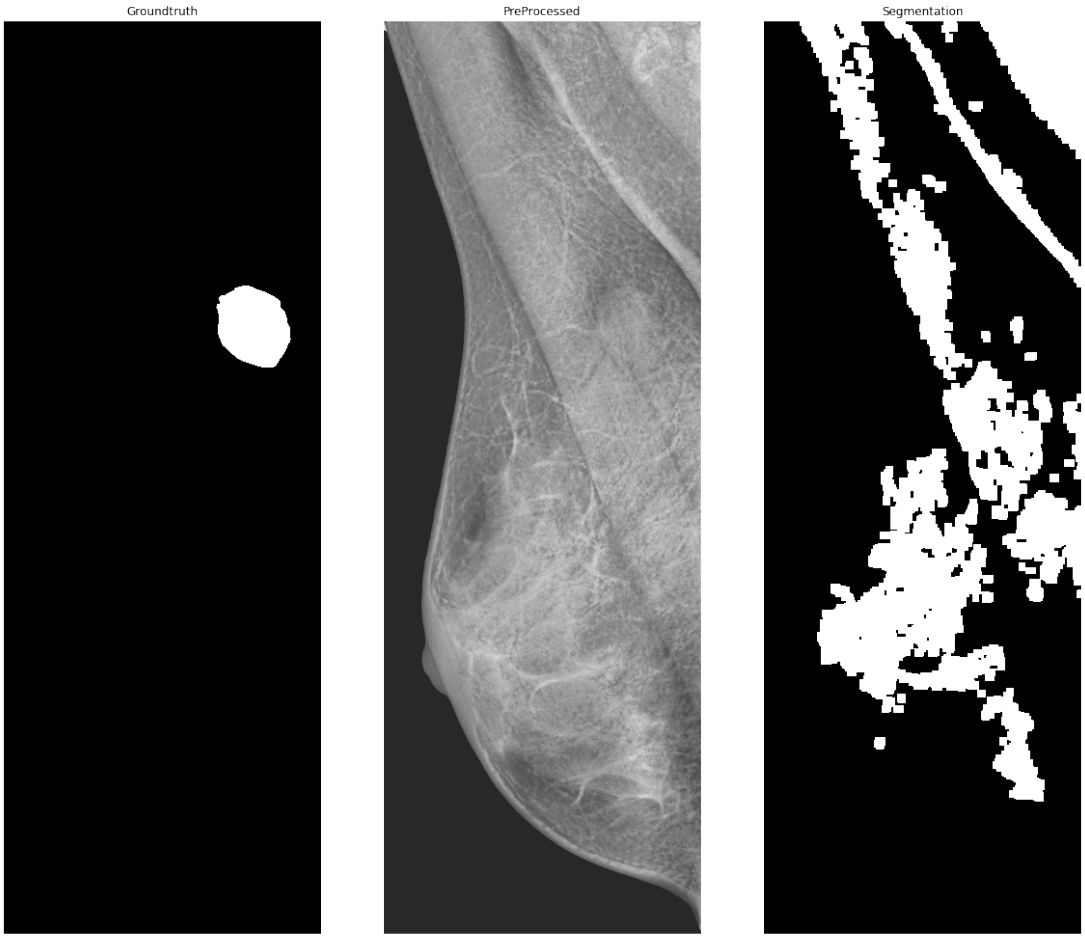

Accuracy results

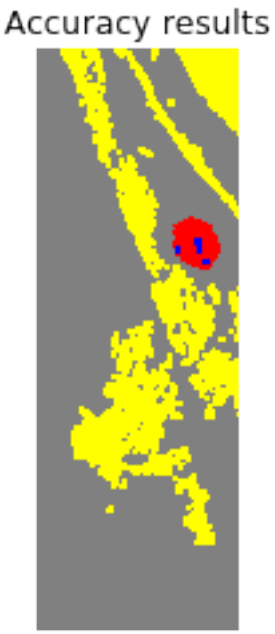
0.7798112518900882 (1, 26, 1)



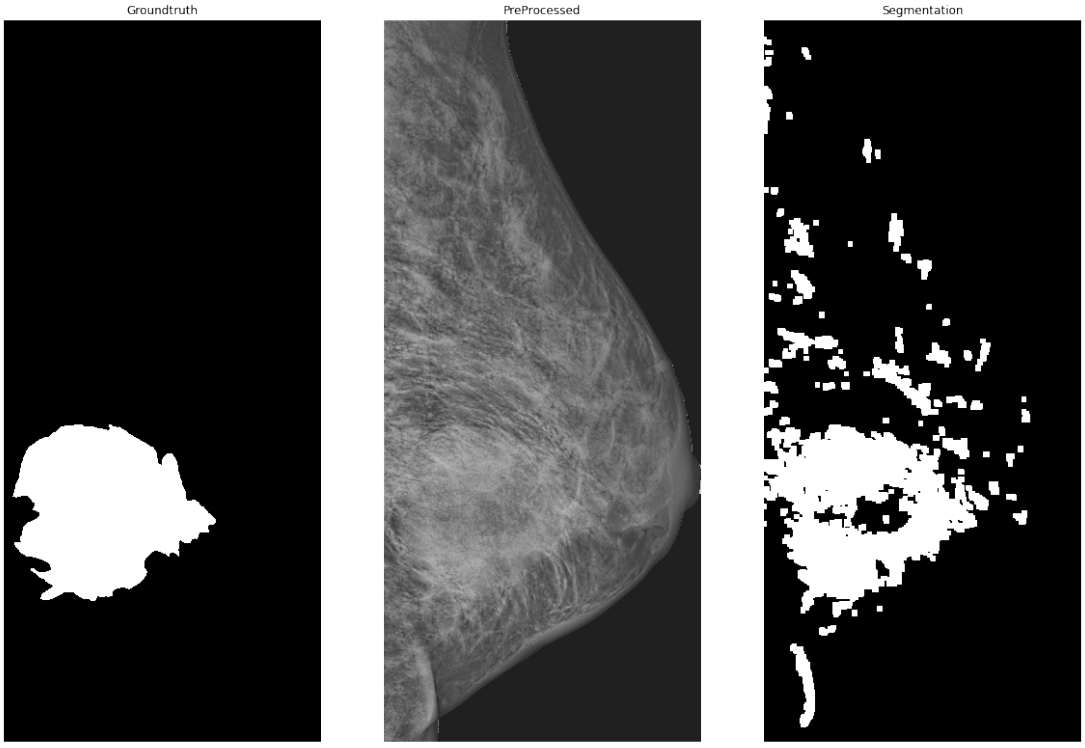


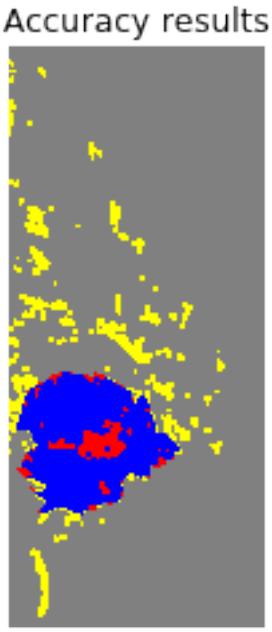

```
0.712507200545478 (1, 42, 1)
```


Groundtruth
PreProcessed
Segmentation

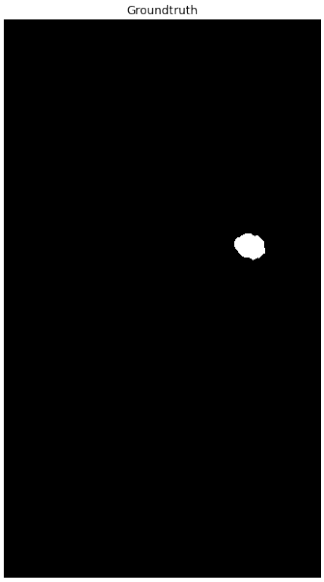


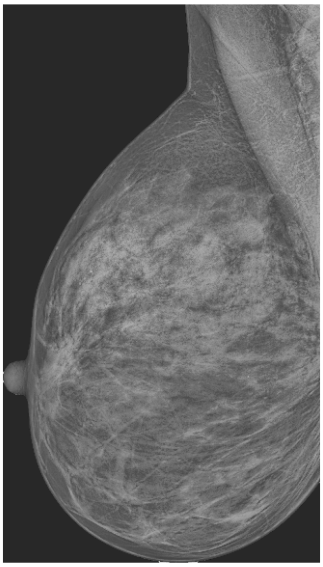

```
0.0627062706270627 (0, 20, 1)
```

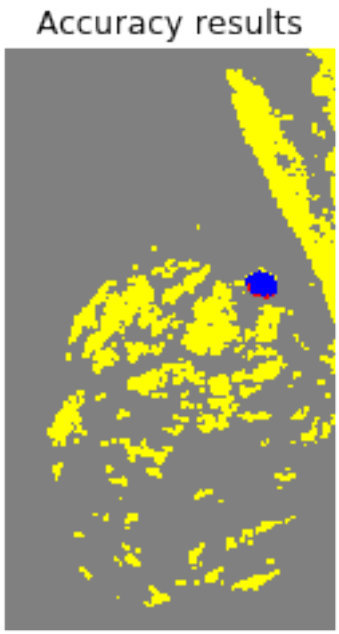




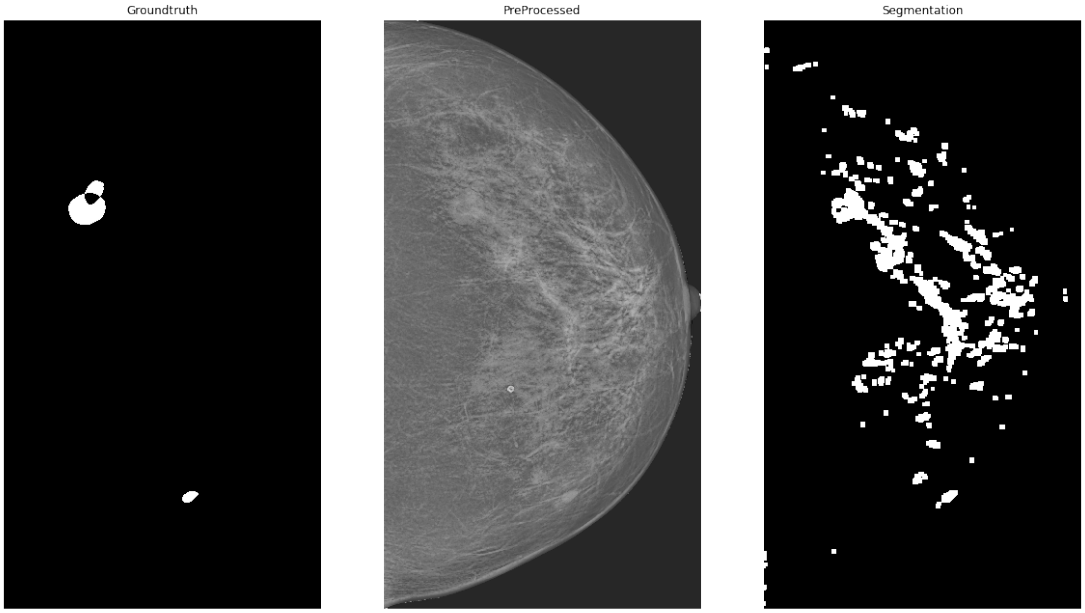

```
0.8598320119036494 (1, 68, 1)
```

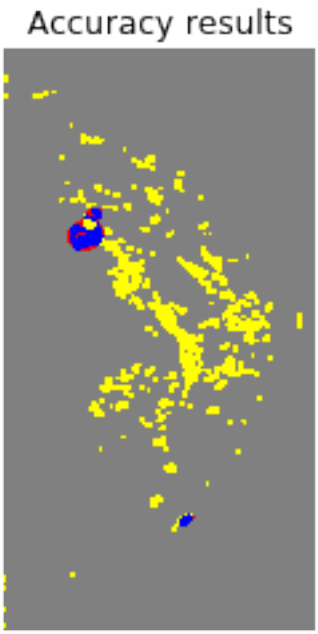




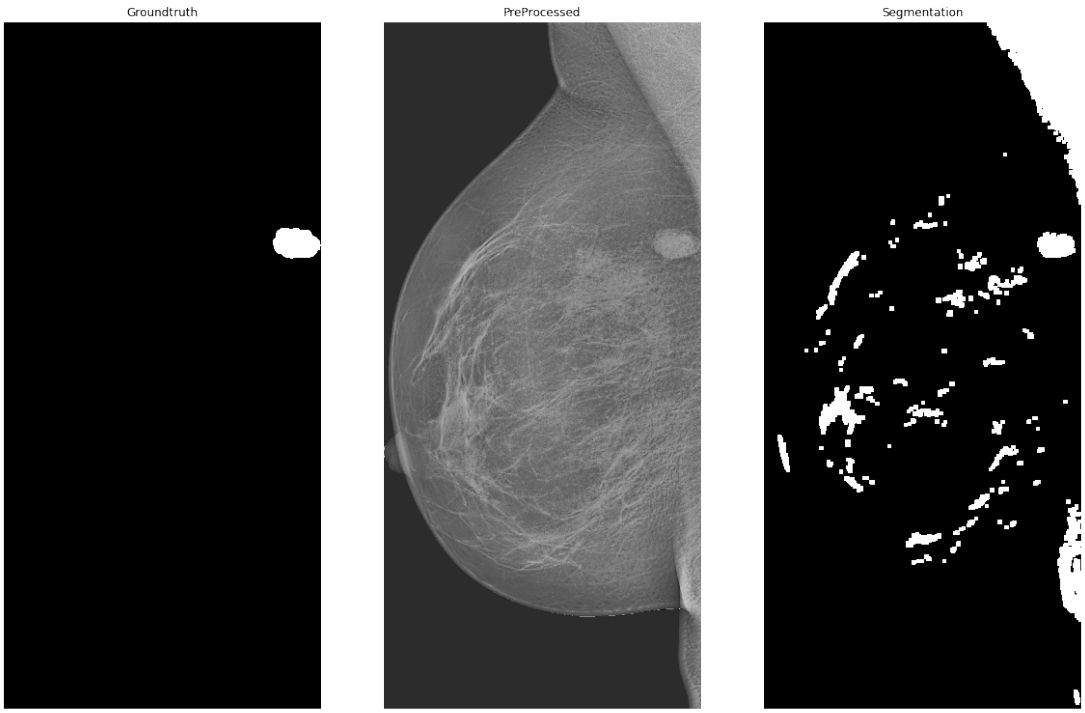


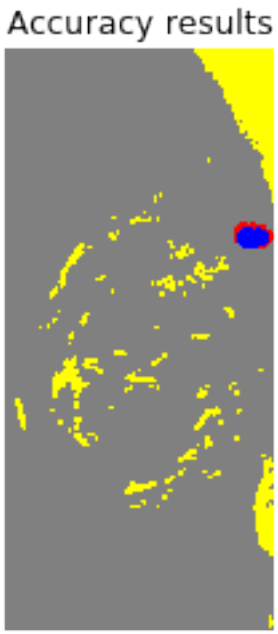
0.7512037037037037 (1, 116, 1)



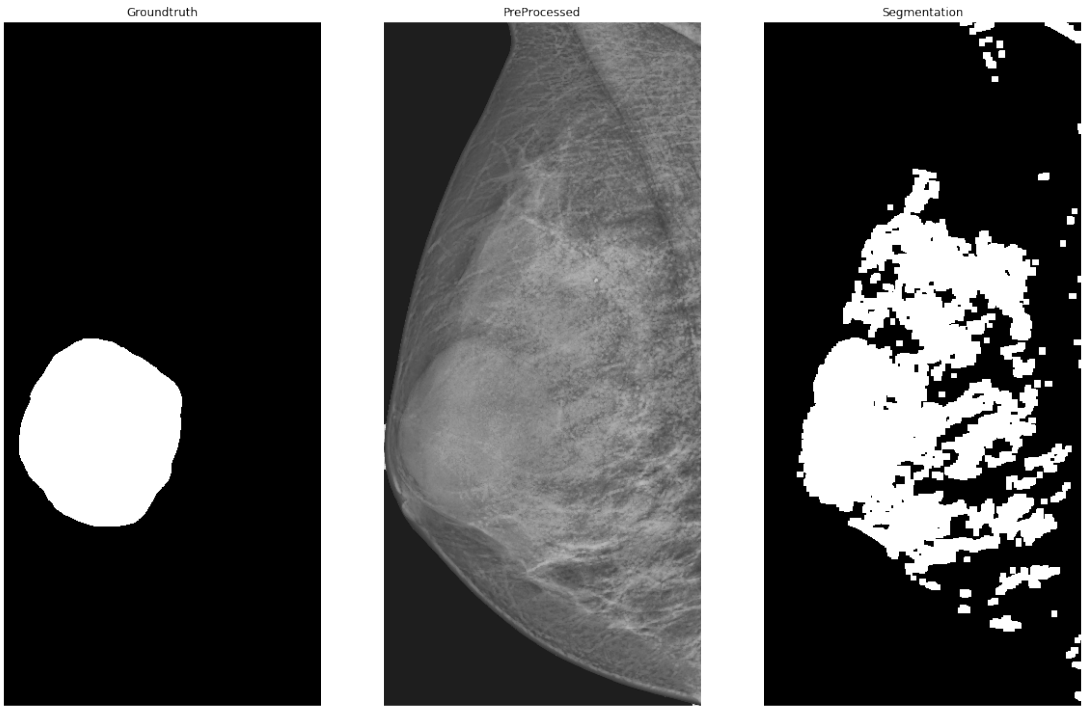


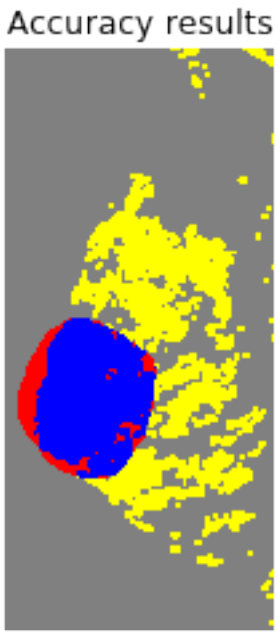

```
0.646603611349957 (2, 101, 2)
```



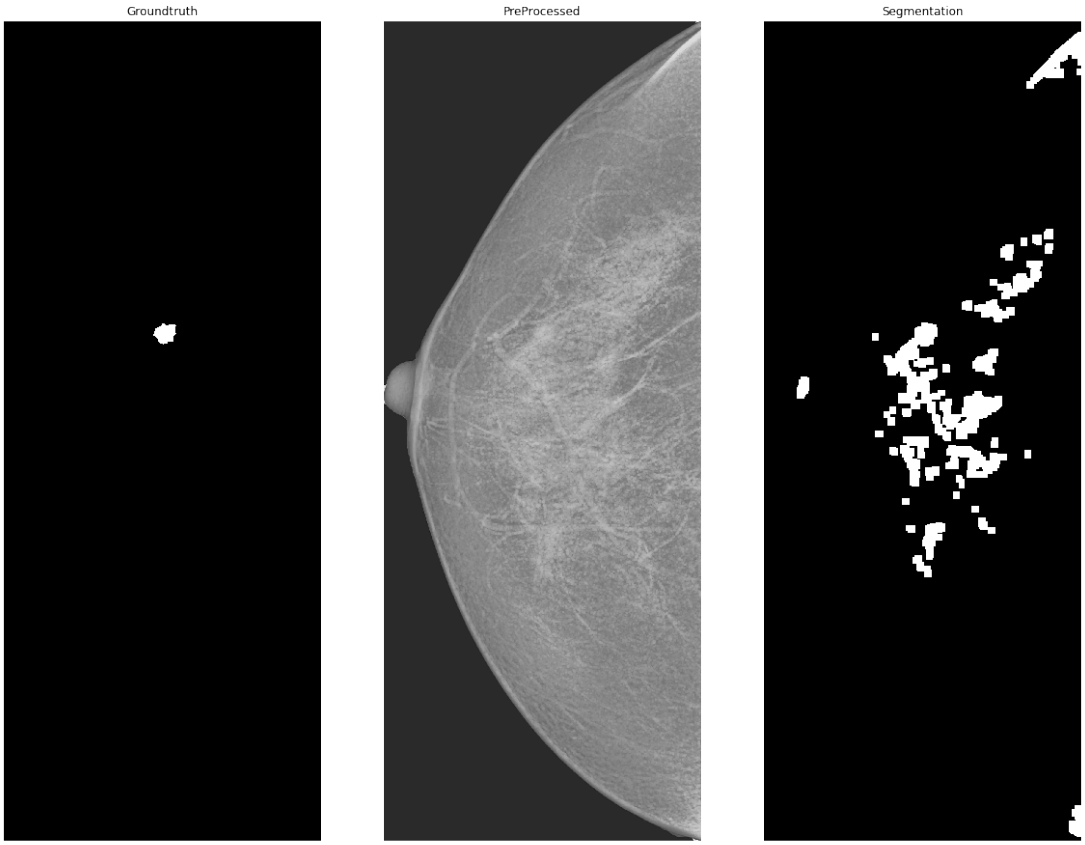


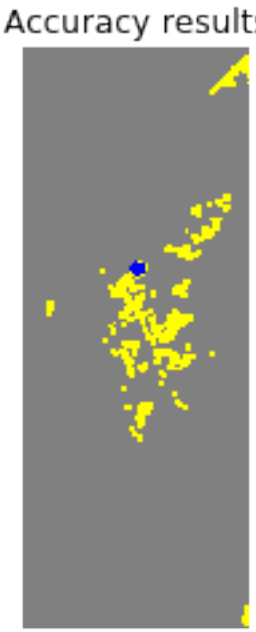

```
0.6788824820864194 (1, 88, 1)
```



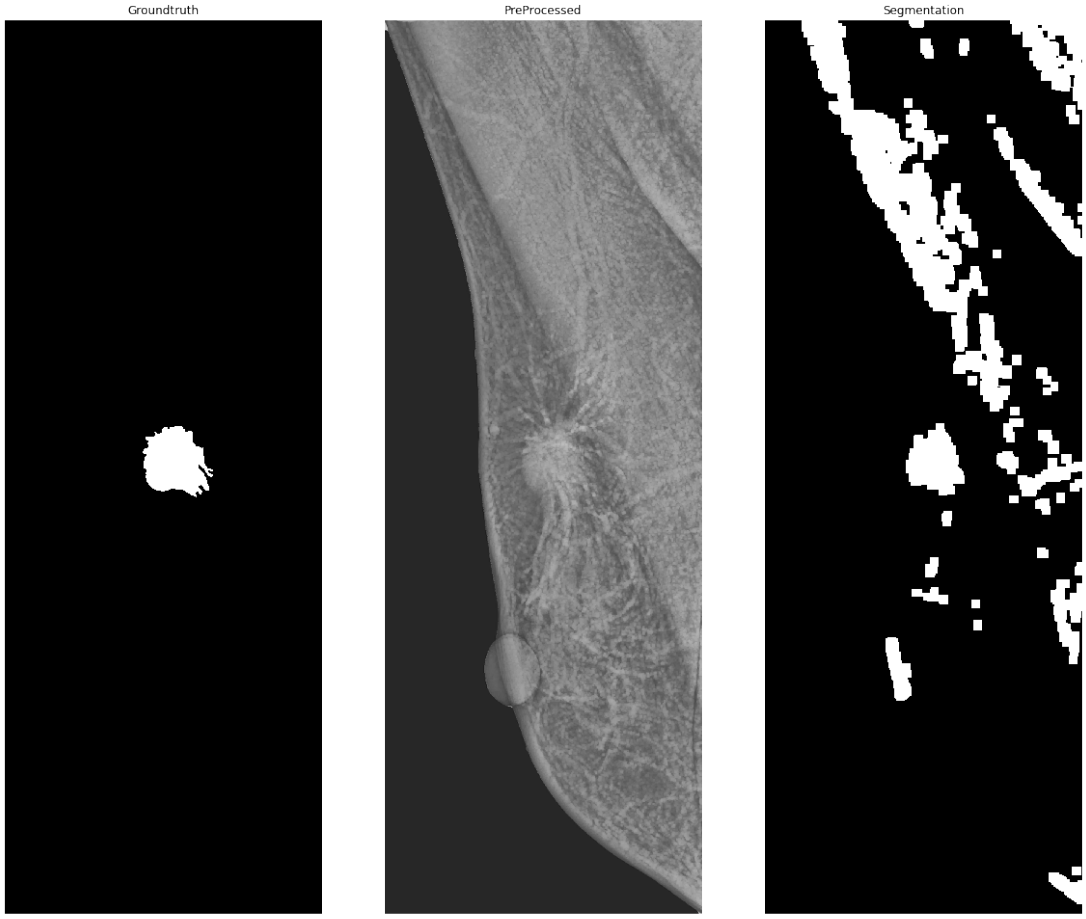


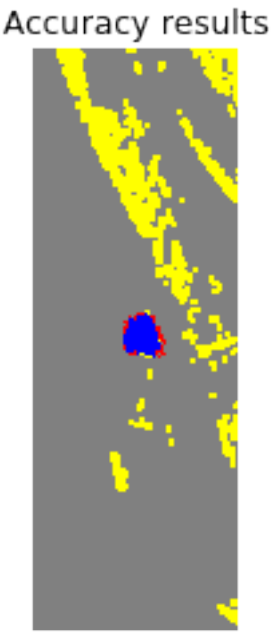

```
0.3422175214808613 (1, 47, 1)
```


Groundtruth
PreProcessed
Segmentation


Accuracy results

0.07515653741104773 (0, 32, 1)

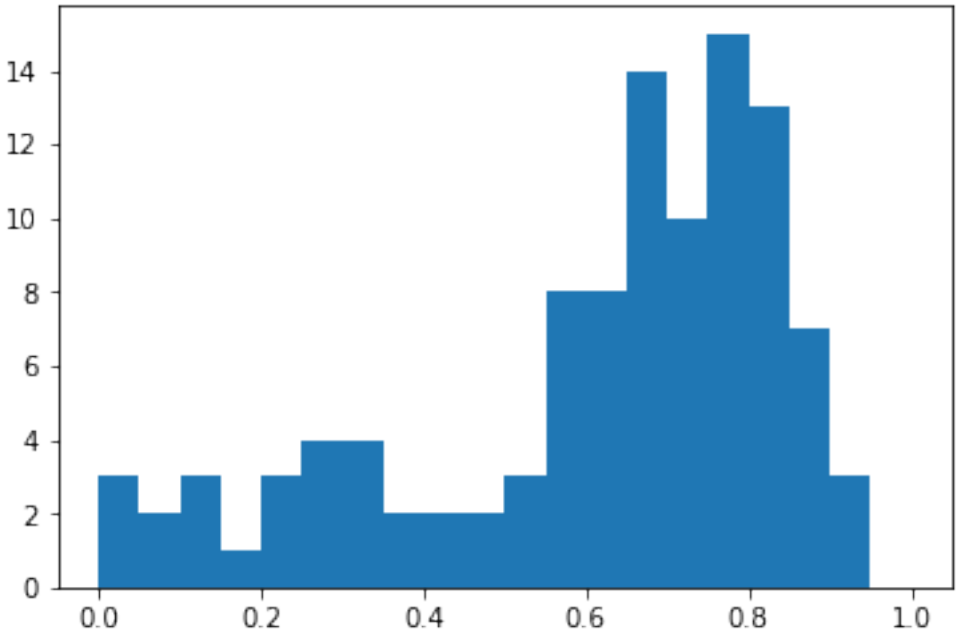





```
0.7753498179030094 (1, 36, 1)
```

```
In [20]: # Histogram of the Jaccard index
         plt.hist(jaccard,bins=20, range=[0,1])
```

```
Out[20]: (array([ 3.,  2.,  3.,  1.,  3.,  4.,  4.,  2.,  2.,  2.,  3.,  8.,  8.,
                14., 10., 15., 13.,  7.,  3.,  0.]),
          array([0.  , 0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ,
                 0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95, 1.  ]),
          <a list of 20 Patch objects>)
```




```
In [25]: # Counting all the positive identifications
         pos = 0
         masses = 0
         for res1, _, res2 in dice:
             pos = pos + res1
             masses = masses + res2
         print(pos, masses, pos/masses)
```

```
110 115 0.9565217391304348
```

```
In [33]: # Evaluating only candidates with Jaccard > 0.5

         pos = 0
         masses = 0
         for dice1, jac in zip(dice, jaccard):
             res1, _, res2 = dice1
             masses = masses + res2
             if jac >= 0.50:
                 pos = pos + res1
         print(pos, masses, pos/masses)
```

```
88 115 0.7652173913043478
```

```
In [30]: np.mean(jaccard)
```

```
Out[30]: 0.6129834039514204
```

Jaccard Index to Verify Results

```
In [ ]: print("Number of images for training is {0}, number of images for testing is {1}".forma
            len(data_set["train"]), len(data_set["test"])))

        av_jaccard = 0
        img = data_set["train"]
        count = 0
        for m in img:
            m.read_data()
            m.image_data = testPreProcessing(m.image_data,20,10.0)
            #Best parameters should be made default ... for segmentations
            Seg, SegA, SegB = Segmentation(m.image_data, Openkernel=(20,20), Openkernel1=(23,23
            av_jaccard = av_jaccard + segmentation.jaccard_index(SegB,  m.cropped_ground_truth)
            count = count + 1
            print(segmentation.jaccard_index(SegB,  m.cropped_ground_truth))

        print("Average jaccard index for training set (Segmentaion1)is {}".format(av_jaccard/co
        #print("Average jaccard index for training set (Segmentaion2)is {}".format(av_jaccard2/

In [ ]: path_image = "../dataset/images/22579730_bbd6a3a35438c11b_MG_R_ML_ANON.tif"
        path_mask = "../dataset/masks/20587080_b6a4f750c6df4f90_MG_R_ML_ANON.png"
        pectoral_muscle = "../dataset/pectoral_muscle_masks/20587080_b6a4f750c6df4f90_MG_R_ML_A
        mm = MammogramImage(path_image, path_mask, pmuscle_mask_path=pectoral_muscle)


        img = mm.image_data


        enhanced = preprocessing.clahe(img)
        morpho = preprocessing.morphoEnhancement(img)
        img_wavelet = preprocessing.waveletTransform(morpho)

        plt.figure(figsize=(20, 10))
        plt.subplot(1,4,1)
        plt.axis("off")
        plt.imshow(img, interpolation="nearest", cmap=plt.cm.gray)
        plt.title('Image')

        plt.subplot(1,4,2)
        plt.axis("off")
        plt.imshow(enhanced, interpolation="nearest", cmap=plt.cm.gray)
        plt.title('CLAHE')

        plt.subplot(1,4,3)
        plt.axis("off")
```

```
plt.imshow(morpho, interpolation="nearest", cmap=plt.cm.gray)
plt.title('Morpho')

plt.subplot(1,4,4)
plt.axis("off")
plt.imshow(img_wavelet, interpolation="nearest", cmap=plt.cm.gray)
plt.title('Wavelet and Morpho')
plt.show()
```