

Isolation of Training Data

This notebook will prepare the data for the next set of segmentation options - Neural Networks

Author: Alexander Goudemond, Student Number: 219030365

This notebook will prepare our data to be used for training, and in another notebook we can train NN!

Imports

```
In [1]: from os import getcwd, walk, mkdir, stat, remove
        from os import sep # used later on, in a function, to print directory contents
        from os.path import exists, basename, join

        from shutil import copyfile

        from PIL.Image import fromarray
        import cv2

        import matplotlib.pyplot as plt
        import numpy as np
```

Directory of Images

```
In [2]: def get_manual_segmentation_directories(startPath):
        location_array = []
        acceptable_folders = ["SEG"]

        for root, dirs, files in walk(startPath):
            # skip this folder
            if ("OriginalZipped" in root):
                continue

            elif (root[-3 : ] not in acceptable_folders):
                continue

            location_array.append(root)

        return location_array
        ###
```

```
In [3]: current_directory = getcwd()
        desired_directory = "..\\..\\Comp700_Processed_DataSets_1"
```

```
In [4]: path = (current_directory + "\\\" + desired_directory)
        location_array = get_manual_segmentation_directories(path)
```

```
In [5]: # first 10
        print( location_array[0:10] )
        print("Number of folders:", len( location_array ) )
```

```
['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Processed_DataSets_1\\BF-C2
```

```
DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\01_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\02_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\02_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01_ST\\SEG']
```

Number of folders: 36

We are now in a position where we can investigate the quantity of pictures inside each of those 36 folders. These 36 folders act as the Memo for our dataset, so collecting the names together will allow us to fetch the relevant images as our training images:

In [6]: *# loop through location_array, looking at the number of images:*

```
image_quantities = []

for item in location_array:
    for root, dirs, files in walk(item):
        image_quantities.append(len(files))

image_quantities[0:10]
```

Out[6]: [49, 1764, 8, 1764, 50, 1376, 50, 1376, 9, 84]

In [7]: *# use image_quantities and location_array to generate a collection of paths for each ima*

```
y_training_data_paths = []
temp_array = []

for item in location_array:

    for root, dirs, files in walk(item):
        # temp_array = files
        for name in files:
            temp_array.append(item + "\\ " + name)

    y_training_data_paths.append(temp_array)
    temp_array = [] # reset
```

In [8]: y_training_data_paths

Out[8]:

```
[['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0058.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0108.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0126.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0175.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0183.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0218.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\man_seg0332.png',
  'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Sanity check:

```
index = -1

for row in y_training_data_paths:
    index += 1
```

```
if (len(row) != image_quantities[index]):  
    print("Incompatible sizes!")
```

Fantastic!

We are now in a position where we can go through each of the elements in the `y_training_data_paths` and extract the number and extensions. This will then be used to fetch the image paths for our `x_training_data_paths`!

Now, it is worth noting that `location_array` sometimes has 2 folders of manually segmented images. So we actually have the opportunity for twice as many training sets!

To recognize this:

```
In [10]: location_array[0:10]
```

```
Out[10]: ['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-HSC\\BF-C2DL-HSC\\01_ST\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-HSC\\BF-C2DL-HSC\\02_ST\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-MuSC\\BF-C2DL-MuSC\\01_GT\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-MuSC\\BF-C2DL-MuSC\\01_ST\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-MuSC\\BF-C2DL-MuSC\\02_GT\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2  
DL-MuSC\\BF-C2DL-MuSC\\02_ST\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\DIC-C  
2DH-HeLa\\DIC-C2DH-HeLa\\01_GT\\SEG',  
          'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\DIC-C  
2DH-HeLa\\DIC-C2DH-HeLa\\01_ST\\SEG']
```

So, we can split this into 2 arrays: `st_array` and `gt_array`!

```
In [11]: gt_y_training_data_paths = []  
st_y_training_data_paths = []  
index = -1  
  
for item in location_array:  
    index += 1  
    if ("_GT" in item):  
        gt_y_training_data_paths.append(y_training_data_paths[index])  
    else:  
        st_y_training_data_paths.append(y_training_data_paths[index])  
  
print("GT_ :", len(gt_y_training_data_paths))  
print("ST_ :", len(st_y_training_data_paths))
```

```
GT_ : 20  
ST_ : 16
```

This is expected because not all of the datasets have ST folders. To verify which folders are missing that data:

-- Fluo-C2DL-Huh7 (x2)

-- Fluo-N2DH-SIM+ (x2)

Let us now look at the x data:

```
In [12]: def get_segmentation_directories(startPath):
    location_array = []
    acceptable_folders = ["\\01", "\\02"]

    for root, dirs, files in walk(startPath):
        # skip this folder
        if ("OriginalZipped" in root):
            continue

        elif (root[-3 : ] not in acceptable_folders) or ("(1)" in root):
            continue

        location_array.append(root)

    return location_array
###
```

```
In [13]: path = (current_directory + "\\" + desired_directory)
image_location_array = get_segmentation_directories(path)

print(image_location_array[0:10])
print(len(image_location_array))
```

```
['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2
DL-HSC\\BF-C2DL-HSC\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Pr
ocessed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02', 'c:\\Users\\G5\\Documents\\GitHub\\CO
MP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\01', 'c:\\Users
\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF
-C2DL-MuSC\\02', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_D
ataSets_1\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\...\\Comp700_Processed_DataSets_1\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\02', 'c:\\Users\\G
5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\Fluo-C2DL-Huh7\\Flu
o-C2DL-Huh7\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_
DataSets_1\\Fluo-C2DL-Huh7\\Fluo-C2DL-Huh7\\02', 'c:\\Users\\G5\\Documents\\GitHub\\COMP
700\\...\\Comp700_Processed_DataSets_1\\Fluo-C2DL-MSC\\Fluo-C2DL-MSC\\01', 'c:\\Users
\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\Fluo-C2DL-MSC\\F
luo-C2DL-MSC\\02']
20
```

Now all we have to do is loop through all images inside the respective paths in image_location_array, and extract the corresponding images!

```
In [14]: def extractNumber(path, symbol):
    right_most_index = path.rfind(symbol)
    return (path[right_most_index + len(symbol) : -4]) # also cut the extension
###

print(y_training_data_paths[7][3])
print( extractNumber(y_training_data_paths[7][3], "seg") ) # must use seg as "png" in na

print(y_training_data_paths[7][3])
print( extractNumber(y_training_data_paths[7][3], "_") ) # Notice how this also works

print(y_training_data_paths[7][3])
print( extractNumber(y_training_data_paths[7][3], "\\") ) # Notice how this also works

c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-
C2DL-MuSC\\02_ST\\SEG\\man_seg0003.png
0003
c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Processed_DataSets_1\\BF-C2DL-MuSC\\BF-
C2DL-MuSC\\02_ST\\SEG\\man_seg0003.png
seg0003
```

c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\02_ST\SEG\man_seg0003.png
man_seg0003

```
In [15]: image_number = ""
row = 0
column = 0

gt_x_training_data_paths = []
temp_array = []
count = -1

for item in image_location_array:
    # print("Item:", item)
    count += 1
    # print(item)
    for root, dirs, files in walk(item):
        # print(files)
        for image in files:
            # initialize
            if (column == 0):
                image_number = extractNumber(gt_y_training_data_paths[row][column], "seg")
                # print("Image Number:", image_number)
                column += 1

            if (image_number in image):
                # print(count)
                temp_array.append(item + "\\\" + image)
                # print(temp_array)

                # stop at final index position
                if (column < len(gt_y_training_data_paths[row])):
                    image_number = extractNumber(gt_y_training_data_paths[row][column],
                    column += 1

        row += 1; column = 0
        gt_x_training_data_paths.append(temp_array)
        temp_array = []

print("folder", (count+1), "of", len(image_location_array) , "complete")

# _GT\\SEG
```

```
folder 1 of 20 complete
folder 2 of 20 complete
folder 3 of 20 complete
folder 4 of 20 complete
folder 5 of 20 complete
folder 6 of 20 complete
folder 7 of 20 complete
folder 8 of 20 complete
folder 9 of 20 complete
folder 10 of 20 complete
folder 11 of 20 complete
folder 12 of 20 complete
folder 13 of 20 complete
folder 14 of 20 complete
folder 15 of 20 complete
folder 16 of 20 complete
folder 17 of 20 complete
folder 18 of 20 complete
folder 19 of 20 complete
folder 20 of 20 complete
```

Let's do a sanity check to ensure we have corresponding images:


```

        column = 0
        if (len(temp_array) != 0):
            row += 1;
            st_x_training_data_paths.append(temp_array)
            temp_array = []

print("folder", (count+1), "of", len(image_location_array) , "complete")

```

```

folder 1 of 20 complete
folder 2 of 20 complete
folder 3 of 20 complete
folder 4 of 20 complete
folder 5 of 20 complete
folder 6 of 20 complete
folder 7 of 20 complete
folder 8 of 20 complete
folder 9 of 20 complete
folder 10 of 20 complete
folder 11 of 20 complete
folder 12 of 20 complete
folder 13 of 20 complete
folder 14 of 20 complete
folder 15 of 20 complete
folder 16 of 20 complete
folder 17 of 20 complete
folder 18 of 20 complete
folder 19 of 20 complete
folder 20 of 20 complete

```

```

In [18]: for row in range(len(st_x_training_data_paths)):
          print( len(st_x_training_data_paths[row]), ":", len(st_y_training_data_paths[row])

```

```

1764 :: 1764
1764 :: 1764
1376 :: 1376
1376 :: 1376
84 :: 84
84 :: 84
48 :: 48
48 :: 48
92 :: 92
92 :: 92
92 :: 92
92 :: 92
115 :: 115
115 :: 115
300 :: 300
300 :: 300

```

Okay! We are now in a position where we can save the contents of the images into dedicated locations! Let's have a folder called training data, which will be partitioned into ST and GT folders

What we can use to our advantage is to just cut off the filename for each of the file paths, and use that to try make directories!

We can then write the content into the locations, making sure that each folder has a sub-folder called "x" and a sub-folder called "y"

Let us start with making the directories:

```

In [19]: # create directories for work we create

```

```

def tryMakeDirectories(current_directory, list):
    path = ""
    for item in list:
        if (path == ""):
            path = item
        else:
            path = path + "\\\" + item

    try:
        # join comes from os.path
        mkdir( join(current_directory, path) )
    except FileExistsError:
        # print("Folder already exists!")
        pass
    except:
        print("Unknown Error Encountered...")

###

def extractLeftHandSide(path, symbol):
    right_most_index = path.rfind(symbol)
    return (path[ : right_most_index ])

###

def extractRightHandSide(path, symbol):
    right_most_index = path.rfind(symbol)
    return (path[ right_most_index : ])

###

```

```

In [20]: print(st_x_training_data_paths[0][0])
print(extractLeftHandSide( st_x_training_data_paths[0][0] , "\\\" ))

```

```

c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\01\processed1_t0000.tif
c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\01

```

```

In [21]: # tryMakeDirectories(getcwd(), "..\\..\\COMP700_Processed_Training_ST\\BF-C2DL-HSC\\BF-C

```

We can also replace the 'Comp700_Processed_DataSets_1' keyword with a more valuable keyword, like "Training_ST" or "Training_GT"

```

In [22]: # check the first 4 results from each array:
row = 1
print(extractLeftHandSide( st_x_training_data_paths[row][0] , "\\\" ))
print(extractLeftHandSide( st_y_training_data_paths[row][0] , "\\\" ))
print(extractLeftHandSide( gt_x_training_data_paths[row][0] , "\\\" ))
print(extractLeftHandSide( gt_y_training_data_paths[row][0] , "\\\" ))

```

```

c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\02
c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\02_ST\SEG
c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\02
c:\Users\G5\Documents\GitHub\COMP700\...\Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C
2DL-HSC\02_GT\SEG

```

```

In [23]: # we only need to use the x_training data as all the folders will correspond!

keyword = "COMP700_Processed_Training_ST"
dirsToMake = []

for row in st_x_training_data_paths:
    temp = row[0] # get first path

```

```

temp = extractRightHandSide(temp, "Comp700_Processed_DataSets_1") # remove initial
temp = extractLeftHandSide(temp, "\\") # get path
temp = temp.replace("Comp700_Processed_DataSets_1", keyword) # replace
temp = "..\\..\\\" + temp # add missing stuff
# print(temp)
dirsToMake.append(temp + "\\\" + \"X\")
dirsToMake.append(temp + "\\\" + \"Y\")

# make directories
for item in dirsToMake:
    tryMakeDirectories(getcwd(), item.split(\"\\\"))

# verify all exist
count = 0
for item in dirsToMake:
    if (exists(item)):
        count += 1
print("Total Directories needed: ", len(dirsToMake), "; Total Directories present: ", co

Total Directories needed: 32; Total Directories present: 32

```

In [24]:

```

# we only need to use the x_training data as all the folders will correspond!

keyword = "COMP700_Processed_Training_GT"
dirsToMake = []

for row in gt_x_training_data_paths:
    temp = row[0] # get first path
    temp = extractRightHandSide(temp, "Comp700_Processed_DataSets_1") # remove initial
    temp = extractLeftHandSide(temp, "\\") # get path
    temp = temp.replace("Comp700_Processed_DataSets_1", keyword) # replace
    temp = "..\\..\\\" + temp # add missing stuff
    # print(temp)
    dirsToMake.append(temp + "\\\" + \"X\")
    dirsToMake.append(temp + "\\\" + \"Y\")

# make directories
for item in dirsToMake:
    tryMakeDirectories(getcwd(), item.split(\"\\\"))

# verify all exist
count = 0
for item in dirsToMake:
    if (exists(item)):
        count += 1
print("Total Directories needed: ", len(dirsToMake), "; Total Directories present: ", co

Total Directories needed: 40; Total Directories present: 40

```

WooHoo! We now have the directories we need for the training data! The next thing to do is to save the files into their corresponding positions!

Let's loop through the variables and save them to their locations using shutil:

In [25]:

```

def bulkCopy(list2D, keyword, keywordToReplace, folderName):
    destination = ""
    filename = ""
    for row in list2D:
        for item in row:

            destination = item.replace(keyword, keywordToReplace)
            destination = extractLeftHandSide(destination, "\\")
            destination = destination + "\\\" + folderName

    # special check:

```



```

        if ("_ST\\SEG" in destination):
            destination = destination.replace("_ST\\SEG", "")
        elif ("_GT\\SEG" in destination):
            destination = destination.replace("_GT\\SEG", "")

        filename = extractNumber(item, "\\")
        # Add extension back now, as it was removed in function above
        filename += item[ -4 : ]

        # print(destination + "\t" + filename)

        #shutil.copyfile()
        copyfile(item, destination + "\\\" + filename)

###

```

We should only do a bulk copy if the files dont exist! Let us verify the files exist:

```

In [26]: def getImageLengths(array):
        image_lengths = []
        for row in range(len(array)):
            image_lengths.append( len(array[row]) )

        return image_lengths
###

def filesAlreadyExist(desired_folder, image_lengths):
    path = getcwd() + "\\..\\..\\\" + desired_folder
    index = -1
    answer = False

    for root, dirs, files in walk(path):
        if (len(files) != 0):
            # initialize
            if (index == -1):
                answer = True

            # print(files)
            index += 1
            if (len(files) != image_lengths[index]):
                answer = False
                break

    return answer
###

```

```

In [27]: gt_x_image_lengths = getImageLengths(gt_x_training_data_paths)
        gt_y_image_lengths = getImageLengths(gt_y_training_data_paths)

        gt_stitched_paths = []
        for i in range(len(gt_x_image_lengths)):
            gt_stitched_paths.append( gt_x_image_lengths[i] )
            gt_stitched_paths.append( gt_y_image_lengths[i] )

        #####

        st_x_image_lengths = getImageLengths(st_x_training_data_paths)
        st_y_image_lengths = getImageLengths(st_y_training_data_paths)

        st_stitched_paths = []
        for i in range(len(st_x_image_lengths)):
            st_stitched_paths.append( st_x_image_lengths[i] )
            st_stitched_paths.append( st_y_image_lengths[i] )

```

```

In [28]: st_files_exist = filesAlreadyExist("COMP700_Processed_Training_ST", st_stitched_paths)

```

```
print(st_files_exist)
```

```
gt_files_exist = filesAlreadyExist("COMP700_Processed_Training_GT", gt_stitched_paths)
print(gt_files_exist)
```

```
True
```

```
True
```

```
In [29]: if ( not st_files_exist):
        bulkCopy(st_x_training_data_paths, "Comp700_Processed_DataSets_1", "COMP700_Processe
```

```
In [30]: if ( not st_files_exist):
        bulkCopy(st_y_training_data_paths, "Comp700_Processed_DataSets_1", "COMP700_Processe
```

```
In [31]: if ( not gt_files_exist):
        bulkCopy(gt_x_training_data_paths, "Comp700_Processed_DataSets_1", "COMP700_Processe
```

```
In [32]: if ( not gt_files_exist):
        bulkCopy(gt_y_training_data_paths, "Comp700_Processed_DataSets_1", "COMP700_Processe
```

Great! We now have a dedicated training set!!

Let us generate some videos to verify that the data corresponds:

Last Sanity check before moving on:

```
In [33]: # "COMP700_Processed_Training_ST"
x_length = []
y_length = []

for root, dirs, files in walk(getcwd() + "\\..\\..\\.." + "COMP700_Processed_Training_ST"):
    # print(dirs)
    if ("X" in dirs or "Y" in dirs):
        # print("yes", root)
        for root2, dirs2, files2 in walk(root + "\\X"):
            x_length.append(len(files2))

        for root2, dirs2, files2 in walk(root + "\\Y"):
            y_length.append(len(files2))

print(x_length)
print(y_length)
```

```
[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]
```

```
[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]
```

```
In [34]: # "COMP700_Processed_Training_GT"
x_length = []
y_length = []

for root, dirs, files in walk(getcwd() + "\\..\\..\\.." + "COMP700_Processed_Training_GT"):
    # print(dirs)
    if ("X" in dirs or "Y" in dirs):
        # print("yes", root)
        for root2, dirs2, files2 in walk(root + "\\X"):
            x_length.append(len(files2))

        for root2, dirs2, files2 in walk(root + "\\Y"):
            y_length.append(len(files2))

print(x_length)
print(y_length)
```

```
[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]
```

[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]

Video Generation

```
In [30]: from moviepy.editor import clips_array, VideoFileClip
         from IPython.display import Video
```

We need to generate a video for each row in st training as well as gt training

We need a list of folder names, as well as a list of file dimensions:

```
In [31]: # generate this for the videos!
def generateFoldersFrom2DList(list):
    folderNameArray = []

    for row in list:
        for item in row:
            # temp = item.replace

            temp = extractLeftHandSide(item, "\\0")
            temp = extractRightHandSide(temp, "\\")
            # print(temp)
            folderNameArray.append(temp[ 1 : ])
            break

    return (folderNameArray)
###

def getEvenElements(list):
    temp = []
    for i in range(len(list)):
        if (i % 2 == 0):
            temp.append(list[i])

    return temp
###
```

```
In [32]: gt_folders = getEvenElements(generateFoldersFrom2DList(gt_x_training_data_paths))

gt_folders
```

```
Out[32]: ['BF-C2DL-HSC',
          'BF-C2DL-MuSC',
          'DIC-C2DH-HeLa',
          'Fluo-C2DL-Huh7',
          'Fluo-C2DL-MS',
          'Fluo-N2DH-GOWT1',
          'Fluo-N2DH-SIM+',
          'Fluo-N2DL-HeLa',
          'PhC-C2DH-U373',
          'PhC-C2DL-PSC']
```

```
In [33]: st_folders = getEvenElements(generateFoldersFrom2DList(st_x_training_data_paths))

st_folders
```

```
Out[33]: ['BF-C2DL-HSC',
          'BF-C2DL-MuSC',
          'DIC-C2DH-HeLa',
          'Fluo-C2DL-MS',
          'Fluo-N2DH-GOWT1',
```

```
'Fluo-N2DL-HeLa',  
'PhC-C2DH-U373',  
'PhC-C2DL-PSC']
```

We now have our folders for GT and ST! Next, we need image dimensions:

We already know from 003 that the images have consistent dimensions. So, we fetch the dimensions using the first image from each folder:

```
In [34]: def getDimensionsFrom2DList(list):  
         image_size_array = []  
  
         # only need dimensions of first image as all dimensions same!  
         for row in list:  
             for item in row:  
                 img = cv2.imread( (item), cv2.IMREAD_GRAYSCALE)  
                 (x, y) = img.shape  
  
                 image_size_array.append([x, y])  
                 break  
  
         return (image_size_array)  
####
```

```
In [35]: gt_image_dimensions = getDimensionsFrom2DList(gt_x_training_data_paths)  
  
gt_image_dimensions
```

```
Out[35]: [[1010, 1010],  
          [1010, 1010],  
          [1036, 1070],  
          [1036, 1070],  
          [512, 512],  
          [512, 512],  
          [1024, 1024],  
          [1024, 1024],  
          [832, 992],  
          [782, 1200],  
          [1024, 1024],  
          [1024, 1024],  
          [690, 628],  
          [773, 739],  
          [700, 1100],  
          [700, 1100],  
          [520, 696],  
          [520, 696],  
          [576, 720],  
          [576, 720]]
```

```
In [36]: st_image_dimensions = getDimensionsFrom2DList(st_x_training_data_paths)  
  
st_image_dimensions
```

```
Out[36]: [[1010, 1010],  
          [1010, 1010],  
          [1036, 1070],  
          [1036, 1070],  
          [512, 512],  
          [512, 512],  
          [832, 992],  
          [782, 1200],  
          [1024, 1024],  
          [1024, 1024],  
          [700, 1100],
```

```
[700, 1100],  
[520, 696],  
[520, 696],  
[576, 720],  
[576, 720]]
```

We can now generate videos!

Before we can progress, we need to extend st_folders and gt_folders to have 16 and 20 elements, instead of 8 and 10:

```
In [37]: st_folders_complete = []  
  
for item in st_folders:  
    st_folders_complete.append(item + "_01")  
    st_folders_complete.append(item + "_02")  
  
gt_folders_complete = []  
  
for item in gt_folders:  
    gt_folders_complete.append(item + "_01")  
    gt_folders_complete.append(item + "_02")  
  
print(len(st_folders_complete))  
print(len(gt_folders_complete))
```

```
16  
20
```

```
In [38]: def generateVideosFromArray(current_directory, desired_folder, list, folderName, dimensions):  
    # only progress if files don't exist  
    makeVideos = False  
  
    if (exists(current_directory + "\\\" + desired_folder)):  
        # Now, go to directory and verify all is there  
        path = walk(current_directory + "\\\" + desired_folder)  
  
        count = 0  
        for root, dirs, files in path:  
            for item in files:  
                count += 1  
  
        if (count == len(list)):  
            print("All Videos exist already!")  
        else:  
            print("Not all Videos exist")  
            makeVideos = True  
  
    if (makeVideos):  
        i = -1  
        output_video = cv2.VideoWriter()  
        frames_per_second = 10  
        picNum = 0  
        fileName = ""  
  
        # Generates Colour Videos  
        for row in list:  
            for item in row:  
                picNum += 1  
                # update on first element only  
                if (picNum == 1):  
                    i += 1  
  
                size = (dimensions[i][1], dimensions[i][0]) # notice order  
                fileName = tag + folderName[i] + ".mp4"
```

```

        output_video = cv2.VideoWriter(
            fileName,
            cv2.VideoWriter_fourcc(*'DIVX'),
            frames_per_second,
            size,
            isColor=useColour # either True or False
        )

        img = plt.imread(item)
        plt.imsave("temp.jpg", img, cmap='gray')
        img = cv2.imread("temp.jpg", cv2.IMREAD_GRAYSCALE)

        output_video.write(img)

    picNum = 0 # reset

    if (len(fileName) != 0):
        cv2.destroyAllWindows()
        output_video.release()
        print("Video finished for ", fileName, sep="")

    # remove at end
    if (exists("temp.jpg")):
        remove("temp.jpg")
###

# from os.path import join
from shutil import move # moves and replaces files

def moveBulkVideos(current_directory, desired_folder, names_list):
    moveVids = False

    try:
        # only progress if files don't exist
        if (exists(current_directory + "\\\" + desired_folder)):
            print("Directory already exists!")

            path = walk(current_directory + "\\\" + desired_folder)

            count = 0
            for root, dirs, files in path:
                for item in files:
                    count += 1

            if (count == len(names_list)):
                print("All Videos Already Exist!")
            else:
                print("Not all Videos in desired directory")
                moveVids = True

    if (moveVids):
        # local function
        tryMakeDirectory(current_directory, desired_folder)

        path = walk(current_directory)

        for root, dirs, files in path:
            for item in files:
                if (".mp4" in item):
                    new_destination = current_directory + "\\\" + desired_folder
                    path1 = join(current_directory, item)
                    path2 = join(new_destination, item)
                    move(path1, path2) # should overwrite existing data

    # Now, go to directory and verify all is there

```

```

        path = walk(current_directory + "\\\" + desired_folder)

        count = 0
        for root, dirs, files in path:
            for item in files:
                count += 1

        if (count == len(names_list)):
            print("All Videos Moved Successfully!")
        else:
            print("Not all Videos Moves Successfully")
    except FileNotFoundError:
        pass
    except:
        print("Unknown Error Encountered...")
###

# create directory for work we create
def tryMakeDirectory(current_directory, destination_directory):
    try:
        # join comes from os.path
        mkdir( join(current_directory, destination_directory) )
    except FileExistsError:
        # print("Folder already exists!")
        pass
    except:
        print("Unknown Error Encountered...")
###

```

```
In [39]: # tryMakeDirectory(getcwd(), "..\\..\\COMP700_Training_Videos")
```

Generate directories for videos:

```
In [40]: tryMakeDirectories(getcwd(), ["..\\..\\COMP700_Training_Videos", "GT", "X"])
tryMakeDirectories(getcwd(), ["..\\..\\COMP700_Training_Videos", "GT", "Y"])
tryMakeDirectories(getcwd(), ["..\\..\\COMP700_Training_Videos", "ST", "X"])
tryMakeDirectories(getcwd(), ["..\\..\\COMP700_Training_Videos", "ST", "Y"])

```

```
In [41]: generateVideosFromArray(getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\X", st_x_trainin
All Videos exist already!
```

```
In [42]: moveBulkVideos(getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\X", st_folders_complete)
Directory already exists!
All Videos Already Exist!
```

```
In [43]: generateVideosFromArray(getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\Y", st_y_trainin
All Videos exist already!
```

```
In [44]: moveBulkVideos(getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\Y", st_folders_complete)
Directory already exists!
All Videos Already Exist!
```

```
In [45]: generateVideosFromArray(getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\X", gt_x_trainin
All Videos exist already!
```

```
In [46]: moveBulkVideos(getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\X", gt_folders_complete)
Directory already exists!
All Videos Already Exist!
```

```
In [47]: generateVideosFromArray(getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\Y", gt_y_trainin

All Videos exist already!

In [48]: moveBulkVideos(getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\Y", gt_folders_complete)

Directory already exists!
All Videos Already Exist!
```

Video Comaprison

What we need to do now is to stitch the X and Y datasets together, to verify that all of the data has been grouped together successfully

We need to travel to our desired directory locations to access the videos:

- (getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\X")
- (getcwd(), "..\\..\\COMP700_Training_Videos\\GT\\Y")
- (getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\X")
- (getcwd(), "..\\..\\COMP700_Training_Videos\\ST\\Y")

```
In [49]: def getVideoPaths(path):
    path = walk(path)

    video_paths = []

    for root, dirs, files in path:
        video_paths = files
        break

    return video_paths
###

def leftAppendStringToList(str, list):
    temp = []

    for item in list:
        temp.append(str + "\\\" + item)

    return temp
###
```

```
In [50]: gt_x_path = getcwd() + "\\..\\..\\\" + "COMP700_Training_Videos\\GT\\X"
gt_y_path = getcwd() + "\\..\\..\\\" + "COMP700_Training_Videos\\GT\\Y"
st_x_path = getcwd() + "\\..\\..\\\" + "COMP700_Training_Videos\\ST\\X"
st_y_path = getcwd() + "\\..\\..\\\" + "COMP700_Training_Videos\\ST\\Y"

gt_x_video_paths = getVideoPaths(gt_x_path)
gt_y_video_paths = getVideoPaths(gt_y_path)
st_x_video_paths = getVideoPaths(st_x_path)
st_y_video_paths = getVideoPaths(st_y_path)

full_gt_x_path = leftAppendStringToList(gt_x_path, gt_x_video_paths)
full_gt_y_path = leftAppendStringToList(gt_y_path, gt_y_video_paths)
full_st_x_path = leftAppendStringToList(st_x_path, st_x_video_paths)
full_st_y_path = leftAppendStringToList(st_y_path, st_y_video_paths)
```

We can use the x and y paths together! We also have some variables we can use for the file names:
gt_folders_complete and st_folders_complete


```

In [51]: def bulkStitch2Videos(test_directory, videoSourcePath1, videoSourcePath2, videoFiles1, v
name_array = ["vid1.mp4", "vid2.mp4"]
videoIndex = -1; count = 0

for i in range(len(videoFiles1)):
    for j in range(2):
        if (j == 0):
            c = VideoFileClip(videoSourcePath1 + "\\" + videoFiles1[i])
        else:
            c = VideoFileClip(videoSourcePath2 + "\\" + videoFiles2[i])

        # getting only first 5 seconds
        clip = c.subclip(0, 5)

        # new clip with new duration
        new_clip = clip.set_duration(10)

        # reduce by 75%
        resized_clip = new_clip.resize(0.25)

        resized_clip.write_videofile(test_directory + "\\" + name_array[j])

        # new_clip.ipython_display(width=100)

    a = VideoFileClip(test_directory + "\\" + name_array[0])
    b = VideoFileClip(test_directory + "\\" + name_array[1])

    # now, stitch together!
    stitched_video = clips_array([a, b])

    stitched_video.write_videofile(test_directory + "\\" + newLabels[i] + ".mp4")

    resized_clip.close()

    try:
        a.close(); b.close()
    except:
        pass

    stitched_video.close()

    # at end, remove videos:
    for name in name_array:
        if (exists(test_directory + "\\" + name)):
            remove(test_directory + "\\" + name)

###

```

```

In [52]: tryMakeDirectories(getcwd(), ["010_Training_Data_Videos", "GT"])
tryMakeDirectories(getcwd(), ["010_Training_Data_Videos", "ST"])

```

```

In [55]: videoSource1 = getcwd() + "\\..\\..\\..\\ + "COMP700_Training_Videos\\GT\\X"
videoSource2 = getcwd() + "\\..\\..\\..\\ + "COMP700_Training_Videos\\GT\\Y"
test_directory = "010_Training_Data_Videos\\GT"

path = walk(test_directory)

for root, dirs, files in path:
    fileCollection = files
    break

if ( not ( len(fileCollection) == len(gt_folders_complete) ) ):
    bulkStitch2Videos(test_directory, videoSource1, videoSource2, gt_x_video_paths, gt_y
else:
    print("Videos Already created!")

```

Videos Already created!

```
In [57]: videoSource1 = getcwd() + "\\..\\..\\.." + "COMP700_Training_Videos\\ST\\X"
videoSource2 = getcwd() + "\\..\\..\\.." + "COMP700_Training_Videos\\ST\\Y"
test_directory = "010_Training_Data_Videos\\ST"

path = walk(test_directory)

for root, dirs, files in path:
    fileCollection = files
    break

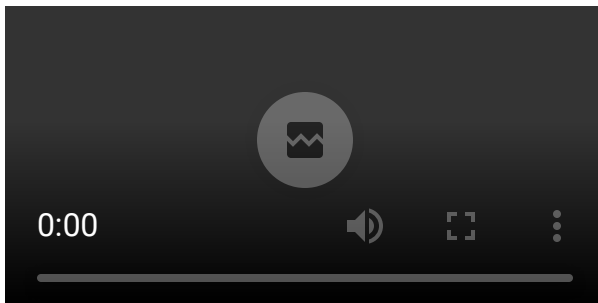
if ( not ( len(fileCollection) == len(st_folders_complete) ) ):
    bulkStitch2Videos(test_directory, videoSource1, videoSource2, st_x_video_paths, st_y
else:
    print("Videos Already created!")
```

Videos Already created!

Fantastic! We are now certain that our training data has been generated successfully! We can verify it by playing a few videos here:

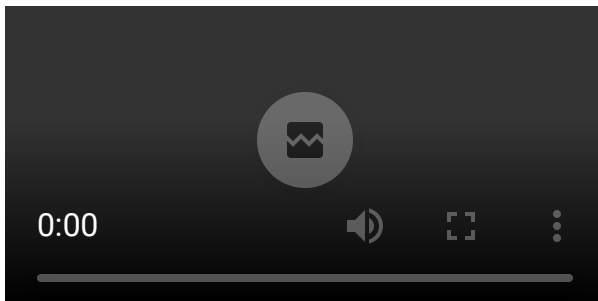
```
In [62]: Video("010_Training_Data_Videos\\ST" + "\\ " + st_folders_complete[0] + ".mp4")
```

Out[62]:



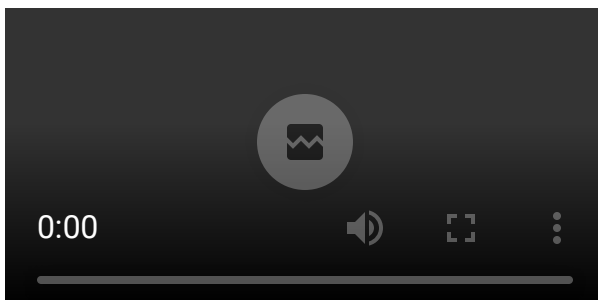
```
In [63]: Video("010_Training_Data_Videos\\ST" + "\\ " + st_folders_complete[5] + ".mp4")
```

Out[63]:



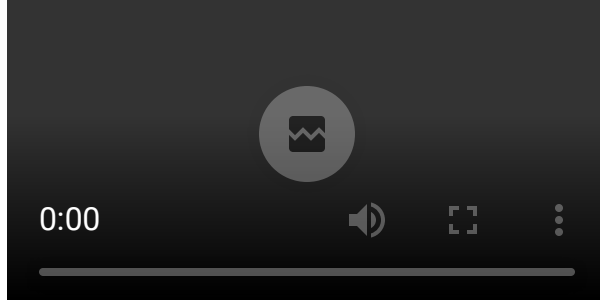
```
In [64]: Video("010_Training_Data_Videos\\GT" + "\\ " + st_folders_complete[2] + ".mp4")
```

Out[64]:



```
In [65]: Video("010_Training_Data_Videos\\GT" + "\\ " + st_folders_complete[7] + ".mp4")
```

Out[65]:



Okay! We are now in a position where we can start training some Neural Networks! We have successfully grouped our training data together, and next we can have a look at some Neural Networks for segmentation

Additional Training Data Sets

In this section of the notebook, we have the option to generate additional training data sets from the RAW data we as well

```
In [34]: desired_directory = "..\\..\\Comp700_DataSets\\Extracted"
```

```
path = (current_directory + "\\\" + desired_directory)
location_array = get_manual_segmentation_directories(path)
```

```
# first 10
```

```
print( location_array[0:10] )
```

```
print("Number of folders:", len( location_array ) )
```

```
['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2D
L-HSC\\BF-C2DL-HSC\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Com
p700_DataSets\\Extracted\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_ST\\SEG', 'c:\\Users\\G5\\Documen
ts\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_G
T\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted
\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\01_GT\\SEG', 'c:\\Use
rs\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2DL-MuSC\\B
F-C2DL-MuSC\\01_ST\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Da
taSets\\Extracted\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\02_GT\\SEG', 'c:\\Users\\G5\\Documents\\G
itHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\02_ST
\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted
\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_DataSets\\Extracted\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01_ST\\SEG']
Number of folders: 36
```

The next step is to get the quantity of images, this should correspond to the work done earlier as well:

```
In [35]: # loop through location_array, looking at the number of images:
```

```
image_quantities = []
```

```
for item in location_array:
    for root, dirs, files in walk(item):
        image_quantities.append(len(files))
```

```
image_quantities[0:10]
```

```
Out[35]: [49, 1764, 8, 1764, 50, 1376, 50, 1376, 9, 84]
```

```
In [36]: # use image_quantities and location_array to generate a collection of paths for each ima

y_training_data_paths = []
temp_array = []

for item in location_array:

    for root, dirs, files in walk(item):
        # temp_array = files
        for name in files:
            temp_array.append(item + "\\\" + name)

    y_training_data_paths.append(temp_array)
    temp_array = [] # reset
```

```
In [37]: index = -1

for row in y_training_data_paths:
    index += 1
    if (len(row) != image_quantities[index]):
        print("Incompatible sizes!")
```

So, we can split this into 2 arrays: st_array and gt_array!

```
In [38]: gt_y_training_data_paths = []
st_y_training_data_paths = []
index = -1

for item in location_array:
    index += 1
    if ("_GT" in item):
        gt_y_training_data_paths.append(y_training_data_paths[index])
    else:
        st_y_training_data_paths.append(y_training_data_paths[index])

print("GT_ :", len(gt_y_training_data_paths))
print("ST_ :", len(st_y_training_data_paths))

GT_ : 20
ST_ : 16
```

This is expected because not all of the datasets have ST folders. To verify which folders are missing that data:

-- Fluo-C2DL-Huh7 (x2)

-- Fluo-N2DH-SIM+ (x2)

Let us now look at the x data:

```
In [39]: path = (current_directory + "\\\" + desired_directory)
image_location_array = get_segmentation_directories(path)

print(image_location_array[0:10])
print(len(image_location_array))

['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_DataSets\\Extracted\\BF-C2D
L-HSC\\BF-C2DL-HSC\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_Dat
aSets\\Extracted\\BF-C2DL-HSC\\BF-C2DL-HSC\\02', 'c:\\Users\\G5\\Documents\\GitHub\\COMP
700\\...\\Comp700_DataSets\\Extracted\\BF-C2DL-MuSC\\BF-C2DL-MuSC\\01', 'c:\\Users\\G
5\\Documents\\GitHub\\COMP700\\...\\Comp700_DataSets\\Extracted\\BF-C2DL-MuSC\\BF-C2D
L-MuSC\\02', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\...\\Comp700_DataSets\\Extra
cted\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\...\\Comp700_DataSets\\Extracted\\DIC-C2DH-HeLa\\DIC-C2DH-HeLa\\02', 'c:\\Users\\G5
```

```

\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\Fluo-C2DL-Huh7\\Fluo-
C2DL-Huh7\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Ex
tracted\\Fluo-C2DL-Huh7\\Fluo-C2DL-Huh7\\02', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_DataSets\\Extracted\\Fluo-C2DL-MSC\\Fluo-C2DL-MSC\\01', 'c:\\Users\\G5
\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_DataSets\\Extracted\\Fluo-C2DL-MSC\\Fluo-C
2DL-MSC\\02']
20

```

Now all we have to do is loop through all images inside the respective paths in image_location_array, and extract the corresponding images!

```

In [40]: image_number = ""
row = 0
column = 0

gt_x_training_data_paths = []
temp_array = []
count = -1

for item in image_location_array:
    # print("Item:", item)
    count += 1
    # print(item)
    for root, dirs, files in walk(item):
        # print(files)
        for image in files:
            # initialize
            if (column == 0):
                image_number = extractNumber(gt_y_training_data_paths[row][column], "seg
                # print("Image Number:", image_number)
                column += 1

            if (image_number in image):
                # print(count)
                temp_array.append(item + "\\\" + image)
                # print(temp_array)

                # stop at final index position
                if (column < len(gt_y_training_data_paths[count])):
                    image_number = extractNumber(gt_y_training_data_paths[row][column],
                    column += 1

            row += 1; column = 0
            gt_x_training_data_paths.append(temp_array)
            temp_array = []

print("folder", (count+1), "of", len(image_location_array) , "complete")

# _GT\\SEG

```

```

folder 1 of 20 complete
folder 2 of 20 complete
folder 3 of 20 complete
folder 4 of 20 complete
folder 5 of 20 complete
folder 6 of 20 complete
folder 7 of 20 complete
folder 8 of 20 complete
folder 9 of 20 complete
folder 10 of 20 complete
folder 11 of 20 complete
folder 12 of 20 complete
folder 13 of 20 complete
folder 14 of 20 complete
folder 15 of 20 complete

```

```
folder 16 of 20 complete
folder 17 of 20 complete
folder 18 of 20 complete
folder 19 of 20 complete
folder 20 of 20 complete
```

```
In [41]: gt_x_image_lengths = getImageLengths(gt_x_training_data_paths)
gt_y_image_lengths = getImageLengths(gt_y_training_data_paths)

print(gt_x_image_lengths)
print(gt_y_image_lengths)

[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]
[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]
```

```
In [42]: image_number = ""
row = 0
column = 0

st_x_training_data_paths = []
temp_array = []
count = -1

for item in image_location_array:
    count += 1
    # print(item)
    for root, dirs, files in walk(item):
        # print(files)
        for image in files:

            # stop because these folders don't have ST folders
            test = image_location_array[count]
            if ("Fluo-C2DL-Huh7" in test) or ("Fluo-N2DH-SIM+" in test):
                break

            # initialize
            if (column == 0):
                image_number = extractNumber(st_y_training_data_paths[row][column], "seg")
                column += 1

            if (image_number in image):
                # print(count)
                temp_array.append(item + "\\\" + image)

            # stop at final index position
            if (column < len(st_y_training_data_paths[row])):
                image_number = extractNumber(st_y_training_data_paths[row][column], "seg")
                column += 1

        column = 0
        if (len(temp_array) != 0):
            row += 1;
            st_x_training_data_paths.append(temp_array)
            temp_array = []

    print("folder", (count+1), "of", len(image_location_array) , "complete")
```

```
folder 1 of 20 complete
folder 2 of 20 complete
folder 3 of 20 complete
folder 4 of 20 complete
folder 5 of 20 complete
folder 6 of 20 complete
folder 7 of 20 complete
folder 8 of 20 complete
```

```

folder 9 of 20 complete
folder 10 of 20 complete
folder 11 of 20 complete
folder 12 of 20 complete
folder 13 of 20 complete
folder 14 of 20 complete
folder 15 of 20 complete
folder 16 of 20 complete
folder 17 of 20 complete
folder 18 of 20 complete
folder 19 of 20 complete
folder 20 of 20 complete

```

```

In [43]: st_x_image_lengths = getImageLengths(st_x_training_data_paths)
st_y_image_lengths = getImageLengths(st_y_training_data_paths)

print(st_x_image_lengths)
print(st_y_image_lengths)

[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]
[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]

```

Next, we replace our desired path names with an appropriate keyword:

```

In [44]: # we only need to use the x_training data as all the folders will correspond!

keyword = "COMP700_Raw_Training_ST"
dirsToMake = []

for row in st_x_training_data_paths:
    temp = row[0] # get first path
    temp = extractRightHandSide(temp, "Comp700_DataSets\\Extracted") # remove initial d
    temp = extractLeftHandSide(temp, "\\") # get path
    temp = temp.replace("Comp700_DataSets\\Extracted", keyword) # replace
    temp = "..\\..\\.." + temp # add missing stuff
    # print(temp)
    dirsToMake.append(temp + "\\\" + "X")
    dirsToMake.append(temp + "\\\" + "Y")

# make directories
for item in dirsToMake:
    tryMakeDirectories(getcwd(), item.split("\\\"))

# verify all exist
count = 0
for item in dirsToMake:
    if (exists(item)):
        count += 1
print("Total Directories needed: ", len(dirsToMake), "; Total Directories present: ", co

Total Directories needed: 32; Total Directories present: 32

```

```

In [45]: # we only need to use the x_training data as all the folders will correspond!

keyword = "COMP700_Raw_Training_GT"
dirsToMake = []

for row in gt_x_training_data_paths:
    temp = row[0] # get first path
    temp = extractRightHandSide(temp, "Comp700_DataSets\\Extracted") # remove initial d
    temp = extractLeftHandSide(temp, "\\") # get path
    temp = temp.replace("Comp700_DataSets\\Extracted", keyword) # replace
    temp = "..\\..\\.." + temp # add missing stuff
    # print(temp)
    dirsToMake.append(temp + "\\\" + "X")
    dirsToMake.append(temp + "\\\" + "Y")

```

```

# make directories
for item in dirsToMake:
    tryMakeDirectories(getcwd(), item.split("\\"))

# verify all exist
count = 0
for item in dirsToMake:
    if (exists(item)):
        count += 1
print("Total Directories needed: ", len(dirsToMake), "; Total Directories present: ", co

```

Total Directories needed: 40; Total Directories present: 40

Next, let us copy them across:

```

In [46]: gt_x_image_lengths = getImageLengths(gt_x_training_data_paths)
gt_y_image_lengths = getImageLengths(gt_y_training_data_paths)

gt_stitched_paths = []
for i in range(len(gt_x_image_lengths)):
    gt_stitched_paths.append( gt_x_image_lengths[i] )
    gt_stitched_paths.append( gt_y_image_lengths[i] )

#####

st_x_image_lengths = getImageLengths(st_x_training_data_paths)
st_y_image_lengths = getImageLengths(st_y_training_data_paths)

st_stitched_paths = []
for i in range(len(st_x_image_lengths)):
    st_stitched_paths.append( st_x_image_lengths[i] )
    st_stitched_paths.append( st_y_image_lengths[i] )

```

```

In [47]: st_files_exist = filesAlreadyExist("COMP700_Raw_Training_ST", st_stitched_paths)
print(st_files_exist)

gt_files_exist = filesAlreadyExist("COMP700_Raw_Training_GT", gt_stitched_paths)
print(gt_files_exist)

True
True

```

```

In [48]: if ( not st_files_exist):
    bulkCopy(st_x_training_data_paths, "Comp700_DataSets\\Extracted", "COMP700_Raw_Train

```

```

In [49]: if ( not st_files_exist):
    bulkCopy(st_y_training_data_paths, "Comp700_DataSets\\Extracted", "COMP700_Raw_Train

```

```

In [50]: if ( not gt_files_exist):
    bulkCopy(gt_x_training_data_paths, "Comp700_DataSets\\Extracted", "COMP700_Raw_Train

```

```

In [51]: if ( not gt_files_exist):
    bulkCopy(gt_y_training_data_paths, "Comp700_DataSets\\Extracted", "COMP700_Raw_Train

```

And that should result in us having dedicated Raw Training data

Last Sanity check before moving on:

```

In [52]: # "COMP700_Raw_Training_GT"
x_length = []
y_length = []

```



```

for root, dirs, files in walk(getcwd() + "\\..\\..\\\\" + "COMP700_Raw_Training_GT"):
    # print(dirs)
    if ("X" in dirs or "Y" in dirs):
        # print("yes", root)
        for root2, dirs2, files2 in walk(root + "\\X"):
            x_length.append(len(files2))

        for root2, dirs2, files2 in walk(root + "\\Y"):
            y_length.append(len(files2))

print(x_length)
print(y_length)

```

```

[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]
[49, 8, 50, 50, 9, 9, 8, 5, 18, 33, 30, 20, 65, 150, 28, 8, 15, 19, 2, 2]

```

In [53]:

```

# "COMP700_Raw_Training_ST"
x_length = []
y_length = []

for root, dirs, files in walk(getcwd() + "\\..\\..\\\\" + "COMP700_Raw_Training_ST"):
    # print(dirs)
    if ("X" in dirs or "Y" in dirs):
        # print("yes", root)
        for root2, dirs2, files2 in walk(root + "\\X"):
            x_length.append(len(files2))

        for root2, dirs2, files2 in walk(root + "\\Y"):
            y_length.append(len(files2))

print(x_length)
print(y_length)

```

```

[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]
[1764, 1764, 1376, 1376, 84, 84, 48, 48, 92, 92, 92, 92, 115, 115, 300, 300]

```