Secondary Segmentation

This notebook will explore the next set of segmenation options

Author: Alexander Goudemond, Student Number: 219030365

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
```

Directories for the Processed of datasets

This section of the notebook will find a way to create directories for the images!

The file order of the dataset is important as we have manually segmented and manually tracked pictures, which we do not plan on processing. We need to find a way to generate the 2 processed datasets without altering this information

An initial option to consider, is generating a list of all the file paths to our images...

This is quite simple, thankfully:

In [2]:
```
def get_directories(startPath):
    location_array = []
    acceptable_folders = ["\\01", "\\02", "SEG", "TRA"]

    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_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', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Pr
ocessed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG', 'c:\\Users\\G5\\Documents\\Gi
tHub\\COMP700\\..\\..\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\TR
A', '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', 'c:\\Users\\G5\\D
ocuments\\GitHub\\COMP700\\..\\..\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HS
C\\02_GT\\SEG', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Processed_Da
taSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\TRA', 'c:\\Users\\G5\\Documents\\GitHub\\COMP
700\\..\\..\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_ST\\SEG', 'c:\\U
sers\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Processed_DataSets_1\\BF-C2DL-HSC
(1)\\BF-C2DL-HSC (1)\\01', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_P
rocessed_DataSets_1\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\02']
Number of folders: 96
```

Great! We can use that variable to generate the locations for our processed images! We just need to replace the keyword "Comp700_DataSets" with our desired folder name, and everything else will follow nicely!

We can further improve the folder readability though, by only keeping the Comp700_DataSets etc. :

In [6]:
```
def cut_string_array(position, array):
    new_array = []

    for item in array:
        new_array.append( item[position : ])

    return new_array
###
```

In [7]:
```
position = len(current_directory + "\\..\\..\\")
# print(position)

reduced_location_array = cut_string_array(position, location_array)
```

In [8]:
```
# first 10
print(reduced_location_array[ 0 : 10])
print()
print("Number of folders:", len( reduced_location_array ) )
```

```
['Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01', 'Comp700_Processed_DataSe
ts_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG', 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC
\\BF-C2DL-HSC\\01_GT\\TRA', 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_
ST\\SEG', 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02', 'Comp700_Process
ed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG', 'Comp700_Processed_DataSets_1\\BF-
C2DL-HSC\\BF-C2DL-HSC\\02_GT\\TRA', 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-
HSC\\02_ST\\SEG', 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\01',
'Comp700_Processed_DataSets_1\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\02']

Number of folders: 96
```

Let's modify the keyword now to our destination folder:

In [9]:
```
def replace_part_of_array(key_word, new_word, array):
```

```
    new_array = []
    temp = ""

    for item in array:
        temp = item.replace(key_word, new_word)
        new_array.append(temp)

    return new_array
###
```

In [10]:
```
desired_locations = replace_part_of_array("Comp700_Processed_DataSets_1", "Comp700_Segme
```

In [11]:
```
print( desired_locations[0:10] )
print("Number of folders:", len( desired_locations ) )
```

```
['Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\01', 'Comp700_Segmented\\BF-C2DL-HSC\\BF-
C2DL-HSC\\01_GT\\SEG', 'Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\TRA', 'Comp7
00_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_ST\\SEG', 'Comp700_Segmented\\BF-C2DL-HSC\\BF
-C2DL-HSC\\02', 'Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG', 'Comp700_Segm
ented\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\TRA', 'Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-H
SC\\02_ST\\SEG', 'Comp700_Segmented\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\01', 'Comp700_Seg
mented\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\02']
Number of folders: 96
```

Okay! We now have a variable containing the folder locations! We can now define some functions to validate all directories exist

In [12]:
```
# 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...")
###

def createBulkDirectories(current_directory, array):
    sub_folders = []
    path = "..\\..\\"

    for item in array:
        sub_folders = item.split("\\")
        # print(sub_folders)

        for folder in sub_folders:
            path += folder
            tryMakeDirectory(current_directory, path)
            path += "\\"

        # reset
        path = "..\\..\\"

    print("Done!")
###
```

In [13]:
```
createBulkDirectories(current_directory, desired_locations)
```

```
Done!
```

dataset Segmentation

This section of the notebook focusses on processing the entire dataset, following the methods found in 005 for 1103_10 and 1103_11

We are going to take advantage of the Thresholding found with OpenCV - specifically the mask value of 17. Let's create a function to do that processing for us:

In [20]:
```
def opencvThresh(img, value=17):
    img = np.array(img).astype(np.uint8)
    ret, thresh = cv2.threshold(img, 0, 255, value)

    return thresh
###

# used to make the segmented values visible, by saving via matplotlib
def getImage(filePath):
    img = plt.imread(filePath)
    plt.imsave("temp.jpg", img, cmap="gray") # desired colourmap for us
    img = cv2.imread( "temp.jpg", cv2.IMREAD_GRAYSCALE)

    return img
###

# process choice influences processOne or processTwo
def bulkProcess(current_directory, original_dataset, location_array):
    kernel = np.ones((3,3), np.uint8)
    counter = 0
    valid_folders = ["01", "02", "SEG", "TRA"]

    name = "segmented_"

    # go to the original_dataset
    path = walk(current_directory + "\\" + original_dataset)

    print("Starting...")

    for root, dirs, files in path:
        # skip zipped files
        if ("OriginalZipped" in root):
            continue
        # end loop because locations exhausted
        elif (counter >= len(location_array)):
            break

        # print(root)

        for item in files:
            # manual info, simply copy as is
            if ("man_" in item):
                # print("Counter:", counter)
                img_path = current_directory + "\\..\\..\\" + location_array[counter] +
                # print(img_path)

                # handle text files
                if (".txt" in item):
                    copyfile(root + "\\" + item, img_path)
                else:
                    # print("EISH")
                    # img = getImage(root + "\\" + item)
                    # should be able to read and save, as 006 did a good job with format
                    img = cv2.imread(root + "\\" + item)
```

```
                    cv2.imwrite(img_path, img)



            # stop working, zipped files found
            elif (".zip" in item):
                break
            else:
                # print("Nope")

                img = getImage(root + "\\" + item)
                # pic_path = root + "\\" + item
                # img = cv2.imread(pic_path)
                # cv2.imshow("Pic", img)
                # cv2.waitKey(0)

                processed_pic = opencvThresh(img)

                # print("Counter:", counter)
                img_path = current_directory + "\\..\\..\\" + location_array[counter] +
                # print(img_path)

                cv2.imwrite(img_path, processed_pic)

            # remove later
            # break

        # update counter
        if (basename(root) in valid_folders):
            counter += 1

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

    print("Finished...")
###

def getFileQuantities(path):
    count = 0
    size_array = []
    valid_folders = ["01", "02", "SEG", "TRA"]

    for root, dirs, files in walk(path):
        count = 0

        for file in files:
            count += 1

        if (basename(root) in valid_folders):
            size_array.append(count)

    return size_array
###
```

In [15]:
```
original_sizes = getFileQuantities( current_directory + "\\" + "..\\..\\Comp700_DataSets
```

In [16]:
```
original_sizes
```

Out[16]:
```
[1764,
 49,
 1765,
 1764,
 1764,
 8,
```

1765,
1764,
1763,
1763,
1375,
50,
1377,
1376,
1376,
50,
1377,
1376,
1376,
1375,
84,
9,
85,
84,
84,
9,
85,
84,
115,
115,
30,
8,
31,
30,
5,
31,
30,
30,
48,
18,
49,
48,
48,
33,
49,
48,
48,
48,
92,
30,
93,
92,
92,
20,
93,
92,
92,
92,
65,
65,
66,
150,
150,
151,
110,
138,
92,
28,
93,
92,
92,
8,

```
 93,
 92,
 92,
 92,
 115,
 15,
 116,
 115,
 115,
 19,
 116,
 115,
 115,
 115,
 300,
 2,
 301,
 300,
 300,
 2,
 301,
 300,
 300,
 300]
```

In [17]:
```python
segmented_sizes = getFileQuantities( current_directory + "\\" + "..\\..\\Comp700_Segment

segmented_sizes
```

Out[17]:
```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
```

```
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0]
```

In [21]:

```
if (original_sizes == segmented_sizes):
    print("True")
else:
    print("False")
    print("\nGenerating now")
    bulkProcess(current_directory, "..\\..\\Comp700_Processed_DataSets_1", desired_locat
```

```
False

Generating now
Starting...
Finished...
```

Segmentation Comparisons

We have some folders provided by the datasets that contain manually segmented images. Let us go through the segmented folder now and identify the quantities of images. We can then generate some videos and stitch them side by side, to identify the success of the segmentation!

In [19]:
```
'''
We can take advantage of reduced_location_array to map segmented images onto the corresp
'''

reduced_location_array[0:10]
```

Out[19]:
```
['Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\TRA',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_ST\\SEG',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\TRA',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_ST\\SEG',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\01',
 'Comp700_Processed_DataSets_1\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\02']
```

In [20]:
```
len(reduced_location_array)
```

Out[20]:
```
96
```

In [21]:
```
manSegPicArray = [0 for i in range(len(reduced_location_array))]
```

In [22]:
```
len(manSegPicArray)
```

Out[22]:
```
96
```

In [23]:
```
'''
We can take advantage of reduced_location_array to map segmented images onto the corresp
inside manSegPicArray
'''

desired_pics = "COMP700_Segmented"

path = walk(current_directory + "\\..\\..\\" + desired_pics)

keyword = "man_seg"
count = 0

for root, dirs, files in path:
    # print(files)
    # print(count, end="; ")
    for item in files:
        if (keyword in item):
            # print(len(files))
            manSegPicArray[count] = len(files)
            break
        break
```

```
    # only update if non empty set!
    if ( len(files) != 0):
        count += 1
```

In [24]:
```
manSegPicArray
```

Out[24]:
```
[0,
 49,
 0,
 1764,
 0,
 8,
 0,
 1764,
 0,
 0,
 0,
 50,
 0,
 1376,
 0,
 50,
 0,
 1376,
 0,
 0,
 0,
 9,
 0,
 84,
 0,
 9,
 0,
 84,
 0,
 0,
 0,
 8,
 0,
 0,
 5,
 0,
 0,
 0,
 0,
 18,
 0,
 48,
 0,
 33,
 0,
 48,
 0,
 0,
 0,
 30,
 0,
 92,
 0,
 20,
 0,
 92,
 0,
 0,
```

```
0,
65,
0,
0,
150,
0,
0,
0,
0,
28,
0,
92,
0,
8,
0,
92,
0,
0,
0,
15,
0,
115,
0,
19,
0,
115,
0,
0,
0,
2,
0,
300,
0,
2,
0,
300,
0,
0]
```

Let's marry this information together:

In [25]:
```
count = 0

for i in range(len(manSegPicArray)):
    if (manSegPicArray[i] != 0):
        print(reduced_location_array[i], manSegPicArray[i], sep=" ... ")
        count += 1

print("\nThere are", count, "folders of segmented images")
```

```
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\01_GT\SEG ... 49
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\01_ST\SEG ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\02_GT\SEG ... 8
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\02_ST\SEG ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\01_GT\SEG ... 50
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\01_ST\SEG ... 1376
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\02_GT\SEG ... 50
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\02_ST\SEG ... 1376
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\01_GT\SEG ... 9
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\01_ST\SEG ... 84
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\02_GT\SEG ... 9
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\02_ST\SEG ... 84
Comp700_Processed_DataSets_1\Fluo-C2DL-Huh7\Fluo-C2DL-Huh7\01_GT\SEG ... 8
Comp700_Processed_DataSets_1\Fluo-C2DL-Huh7\Fluo-C2DL-Huh7\02_GT\SEG ... 5
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\01_GT\SEG ... 18
```

```
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\01_ST\SEG ... 48
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\02_GT\SEG ... 33
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\02_ST\SEG ... 48
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\01_GT\SEG ... 30
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\01_ST\SEG ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\02_GT\SEG ... 20
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\02_ST\SEG ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\01_GT\SEG ... 65
Comp700_Processed_DataSets_1\Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\02_GT\SEG ... 150
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\01_GT\SEG ... 28
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\01_ST\SEG ... 92
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\02_GT\SEG ... 8
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\02_ST\SEG ... 92
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\01_GT\SEG ... 15
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\01_ST\SEG ... 115
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\02_GT\SEG ... 19
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\02_ST\SEG ... 115
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\01_GT\SEG ... 2
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\01_ST\SEG ... 300
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\02_GT\SEG ... 2
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\02_ST\SEG ... 300

There are 36 folders of segmented images
```

Let's extend that summary to see how many images are in the corresponding folder:

In [26]:
```
correspondingPicArray = [0 for i in range(len(reduced_location_array))]
```

In [27]:
```
desired_pics = "COMP700_Segmented"

path = walk(current_directory + "\\..\\..\\" + desired_pics)

keyword1 = "_GT"; keyword2 = "_ST"
count = 0
index = 0

for root, dirs, files in path:
    # print(files)
    # print(count, end="; ")
    for item in files:
        if (keyword1 in reduced_location_array[index]) or (keyword2 in reduced_location_
            # print(len(files))
            # print(count)
            correspondingPicArray[index] = count
            break
        else:
            count = len(files)
            break

    if (len(files) != 0):
        index += 1
```

In [28]:
```
correspondingPicArray
```

Out[28]:
```
[0,
 1764,
 1764,
 1764,
 0,
 1764,
 1764,
 1764,
 0,
 0,
 0,
```

1375,
1375,
1375,
0,
1376,
1376,
1376,
0,
0,
0,
84,
84,
84,
0,
84,
84,
84,
0,
0,
0,
30,
30,
0,
30,
30,
0,
0,
0,
48,
48,
48,
0,
48,
48,
48,
0,
0,
0,
92,
92,
92,
0,
92,
92,
92,
0,
0,
0,
65,
65,
0,
150,
150,
0,
0,
0,
92,
92,
92,
0,
92,
92,
92,
0,
0,
0,

```
115,
115,
115,
0,
115,
115,
115,
0,
0,
0,
300,
300,
300,
0,
300,
300,
300,
0,
0]
```

In [29]:

```
count = 0

for i in range(len(manSegPicArray)):
    if (manSegPicArray[i] != 0):
        print(reduced_location_array[i], "Segmented Pictures:", manSegPicArray[i], "Corr
        count += 1

print("\nThere are", count, "folders of segmented images")
```

```
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\01_GT\SEG ... Segmented Pictures:
... 49 ... Corresponding Pictures: ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\01_ST\SEG ... Segmented Pictures:
... 1764 ... Corresponding Pictures: ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\02_GT\SEG ... Segmented Pictures:
... 8 ... Corresponding Pictures: ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-HSC\BF-C2DL-HSC\02_ST\SEG ... Segmented Pictures:
... 1764 ... Corresponding Pictures: ... 1764
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\01_GT\SEG ... Segmented Pictures:
... 50 ... Corresponding Pictures: ... 1375
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\01_ST\SEG ... Segmented Pictures:
... 1376 ... Corresponding Pictures: ... 1375
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\02_GT\SEG ... Segmented Pictures:
... 50 ... Corresponding Pictures: ... 1376
Comp700_Processed_DataSets_1\BF-C2DL-MuSC\BF-C2DL-MuSC\02_ST\SEG ... Segmented Pictures:
... 1376 ... Corresponding Pictures: ... 1376
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\01_GT\SEG ... Segmented Picture
s: ... 9 ... Corresponding Pictures: ... 84
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\01_ST\SEG ... Segmented Picture
s: ... 84 ... Corresponding Pictures: ... 84
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\02_GT\SEG ... Segmented Picture
s: ... 9 ... Corresponding Pictures: ... 84
Comp700_Processed_DataSets_1\DIC-C2DH-HeLa\DIC-C2DH-HeLa\02_ST\SEG ... Segmented Picture
s: ... 84 ... Corresponding Pictures: ... 84
Comp700_Processed_DataSets_1\Fluo-C2DL-Huh7\Fluo-C2DL-Huh7\01_GT\SEG ... Segmented Pictu
res: ... 8 ... Corresponding Pictures: ... 30
Comp700_Processed_DataSets_1\Fluo-C2DL-Huh7\Fluo-C2DL-Huh7\02_GT\SEG ... Segmented Pictu
res: ... 5 ... Corresponding Pictures: ... 30
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\01_GT\SEG ... Segmented Picture
s: ... 18 ... Corresponding Pictures: ... 48
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\01_ST\SEG ... Segmented Picture
s: ... 48 ... Corresponding Pictures: ... 48
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\02_GT\SEG ... Segmented Picture
s: ... 33 ... Corresponding Pictures: ... 48
Comp700_Processed_DataSets_1\Fluo-C2DL-MSC\Fluo-C2DL-MSC\02_ST\SEG ... Segmented Picture
s: ... 48 ... Corresponding Pictures: ... 48
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\01_GT\SEG ... Segmented Pic
```

```
tures: ... 30 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\01_ST\SEG ... Segmented Pic
tures: ... 92 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\02_GT\SEG ... Segmented Pic
tures: ... 20 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-GOWT1\Fluo-N2DH-GOWT1\02_ST\SEG ... Segmented Pic
tures: ... 92 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\01_GT\SEG ... Segmented Pictu
res: ... 65 ... Corresponding Pictures: ... 65
Comp700_Processed_DataSets_1\Fluo-N2DH-SIM+\Fluo-N2DH-SIM+\02_GT\SEG ... Segmented Pictu
res: ... 150 ... Corresponding Pictures: ... 150
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\01_GT\SEG ... Segmented Pictu
res: ... 28 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\01_ST\SEG ... Segmented Pictu
res: ... 92 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\02_GT\SEG ... Segmented Pictu
res: ... 8 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\Fluo-N2DL-HeLa\Fluo-N2DL-HeLa\02_ST\SEG ... Segmented Pictu
res: ... 92 ... Corresponding Pictures: ... 92
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\01_GT\SEG ... Segmented Picture
s: ... 15 ... Corresponding Pictures: ... 115
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\01_ST\SEG ... Segmented Picture
s: ... 115 ... Corresponding Pictures: ... 115
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\02_GT\SEG ... Segmented Picture
s: ... 19 ... Corresponding Pictures: ... 115
Comp700_Processed_DataSets_1\PhC-C2DH-U373\PhC-C2DH-U373\02_ST\SEG ... Segmented Picture
s: ... 115 ... Corresponding Pictures: ... 115
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\01_GT\SEG ... Segmented Pictures:
... 2 ... Corresponding Pictures: ... 300
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\01_ST\SEG ... Segmented Pictures:
... 300 ... Corresponding Pictures: ... 300
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\02_GT\SEG ... Segmented Pictures:
... 2 ... Corresponding Pictures: ... 300
Comp700_Processed_DataSets_1\PhC-C2DL-PSC\PhC-C2DL-PSC\02_ST\SEG ... Segmented Pictures:
... 300 ... Corresponding Pictures: ... 300

There are 36 folders of segmented images
```

A detailed look at those results indicates that the GT Segmentation always <= quantity of images in the corresponding folder. Perhaps GT stands for Generated Trace? ST, on the other hand, is always = quantity of images in the corresponding folder.

FLUO-N2DL-SIM+ as well as Fluo-C2DL-Huh7 also contain 2 folders of segmentated images, all the rest have 4. Meaning 8 *4* + *2* 2 = 32 + 4 = 36

We can use this information to generate 10 videos, each stitched together, and watch them to see the the segmentation differences. We can then decide if we need to further segment, or can move to post processing immediately!

In [30]:
```
from shutil import move # moves and replaces files

from moviepy.editor import clips_array, VideoFileClip
from IPython.display import Video
```

First, we need an array of the folder locations:

In [31]:
```
'''
We only need to show every _OTHER_ folder, as each dataset has a
 training and challenge set. So out of 20 files, we need to show 10

First things first, let us create an array of the directory locations
'''
```

```
data_sets = "..\\..\\Comp700_DataSets"
current_directory = getcwd()

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

directory_array = [] # contains the main folders

i = 1
for root, dirs, files in path:
    if (i == 2):
        directory_array = dirs
        break

    i += 1

print("Directory Array")
print(directory_array)
```

```
Directory Array
['BF-C2DL-HSC', 'BF-C2DL-HSC (1)', 'BF-C2DL-MuSC', 'BF-C2DL-MuSC (1)', 'DIC-C2DH-HeLa',
'DIC-C2DH-HeLa (1)', 'Fluo-C2DL-Huh7', 'Fluo-C2DL-Huh7 (1)', 'Fluo-C2DL-MSC', 'Fluo-C2DL
-MSC (1)', 'Fluo-N2DH-GOWT1', 'Fluo-N2DH-GOWT1 (1)', 'Fluo-N2DH-SIM+', 'Fluo-N2DH-SIM+
(1)', 'Fluo-N2DL-HeLa', 'Fluo-N2DL-HeLa (1)', 'PhC-C2DH-U373', 'PhC-C2DH-U373 (1)', 'PhC
-C2DL-PSC', 'PhC-C2DL-PSC (1)']
```

In [32]:

```
def replace_part_of_array(key_word, new_word, array):
    new_array = []
    temp = ""

    for item in array:
        temp = item.replace(key_word, new_word)
        new_array.append(temp)

    return new_array
###

# First, generate a list of the locations for each folder of Petri Dish images
data_sets = "..\\..\\Comp700_Segmented"
path = walk(current_directory + "\\" + data_sets) # reset path

location_array = reduced_location_array

location_array = replace_part_of_array("Comp700_Processed_DataSets_1", "Comp700_Segmente

for i in range(len(location_array)):
    location_array[i] = current_directory + "\\..\\..\\" + location_array[i] + "\\"


print(location_array[0:10])

print("\n", len(location_array))
```

```
['c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC\\BF-
C2DL-HSC\\01\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Segmented
\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\SEG\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_GT\\TRA\\', 'c:\\Users\\G5\\Do
cuments\\GitHub\\COMP700\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\01_ST\\SE
G\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC
\\BF-C2DL-HSC\\02\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Segmen
ted\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\SEG\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700
\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_GT\\TRA\\', 'c:\\Users\\G5\\Do
cuments\\GitHub\\COMP700\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC\\BF-C2DL-HSC\\02_ST\\SE
G\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700_Segmented\\BF-C2DL-HSC
(1)\\BF-C2DL-HSC (1)\\01\\', 'c:\\Users\\G5\\Documents\\GitHub\\COMP700\\..\\..\\Comp700
_Segmented\\BF-C2DL-HSC (1)\\BF-C2DL-HSC (1)\\02\\']
```

In [33]:
```
# generate this for the videos!
folderNameArray = []

for i in range(len(location_array)):
    temp = location_array[i].replace(current_directory + "\\..\\..\\" + "Comp700_Segment


    #remove leading folder name and trailing braces
    position = temp.index("\\")
    temp = temp[ position + 1 : -1 ]

    temp = temp.replace("\\", "_")

    folderNameArray.append(temp)



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

```
['BF-C2DL-HSC_01', 'BF-C2DL-HSC_01_GT_SEG', 'BF-C2DL-HSC_01_GT_TRA', 'BF-C2DL-HSC_01_ST_
SEG', 'BF-C2DL-HSC_02', 'BF-C2DL-HSC_02_GT_SEG', 'BF-C2DL-HSC_02_GT_TRA', 'BF-C2DL-HSC_0
2_ST_SEG', 'BF-C2DL-HSC (1)_01', 'BF-C2DL-HSC (1)_02']
```

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]:
```
image_size_array = []

path = walk(current_directory + "\\" + data_sets)  # reset path

i = -1 # will grow from 0 to 39
for root, dirs, files in path:
    for item in files:
        # print(item)
        if (".txt" not in item) and (".zip" not in item):
            i += 1

            img = cv2.imread( (location_array[i] + item), cv2.IMREAD_GRAYSCALE)
            (x, y) = img.shape

            image_size_array.append([x, y])
            break


print(image_size_array)
```

```
[[1010, 1010], [1010, 1010], [1010, 1010], [1010, 1010], [1010, 1010], [1010, 1010], [10
10, 1010], [1010, 1010], [1010, 1010], [1010, 1010], [1036, 1070], [1036, 1070], [1036,
1070], [1036, 1070], [1036, 1070], [1036, 1070], [1036, 1070], [1036, 1070], [1036, 107
0], [1036, 1070], [512, 512], [512, 512], [512, 512], [512, 512], [512, 512], [512, 51
2], [512, 512], [512, 512], [512, 512], [512, 512], [1024, 1024], [1024, 1024], [1024, 1
024], [1024, 1024], [1024, 1024], [1024, 1024], [1024, 1024], [1024, 1024], [832, 992],
[832, 992], [832, 992], [832, 992], [782, 1200], [782, 1200], [782, 1200], [782, 1200],
[832, 992], [782, 1200], [1024, 1024], [1024, 1024], [1024, 1024], [1024, 1024], [1024,
1024], [1024, 1024], [1024, 1024], [1024, 1024], [1024, 1024], [1024, 1024], [690, 628],
[690, 628], [690, 628], [773, 739], [773, 739], [773, 739], [718, 660], [790, 664], [70
0, 1100], [700, 1100], [700, 1100], [700, 1100], [700, 1100], [700, 1100], [700, 1100],
[700, 1100], [700, 1100], [700, 1100], [520, 696], [520, 696], [520, 696], [520, 696],
[520, 696], [520, 696], [520, 696], [520, 696], [520, 696], [520, 696], [576, 720], [57
6, 720], [576, 720], [576, 720], [576, 720], [576, 720], [576, 720], [576, 720], [576, 7
20], [576, 720]]
```

Useful functions:

In [35]:
```
def generateVideos(current_directory, desired_folder, use_colour):
    # 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(location_array)):
            print("All Videos exist already!")
        else:
            print("Not all Videos exist")
            makeVideos = True
    else:
        makeVideos = True

    if (makeVideos):
        path = walk(current_directory + "\\" + data_sets) # reset path

        i = -1
        output_video = cv2.VideoWriter()
        frames_per_second = 10
        picNum = 0
        fileName = ""

        # Generates Colour Videos
        for root, dirs, files in path:
            for item in files:
                if (".txt" not in item) and (".zip" not in item):
                    picNum += 1
                    # update on first element only
                    if  (picNum == 1):
                        i += 1
                        index = i // 2 # used for output video as 2 copies for each dire

                        size = (image_size_array[i][1], image_size_array[i][0] ) # notic
                        fileName = "video_segmented_" + folderNameArray[i] + ".mp4"

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

                    img = plt.imread(location_array[i] + 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")
###
```

In [36]:
```
# from os.path import join
# from shutil import move # moves and replaces files

def moveBulkVideos(current_directory, desired_folder):
    # only progress if files don't exist
    if (exists(current_directory + "\\" + desired_folder)):
        print("Videos already exist!")
    else:
        # 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
                    move(join(current_directory, item), join(new_destination, item)) # s

        # 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(location_array)):
            print("All Videos Moved Successfully!")
        else:
            print("Not all Videos Moves Successfully")
###
```

In [37]:
```
generateVideos(current_directory, "..\\..\\Comp700_VideosOfSegmentation", use_colour=Fal
```

```
All Videos exist already!
```

In [38]:
```
moveBulkVideos(current_directory, "..\\..\\Comp700_VideosOfSegmentation")
```

```
Videos already exist!
```

Video Comaprisons

What we can do now is to compare the collection of segmentation videos together. We can then compare videos against the provided Solutions and see how we are doing

Let's combine the videos from the sama dataset together now

First, we need the locations and names of the videos:

In [39]:
```
desired_directory = getcwd() + "\\..\\..\\" + "Comp700_VideosOfSegmentation"

path = walk(desired_directory)

videoFiles = []
```

```
for root, dirs, files in path:
    videoFiles = files
    break

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

```
['video_segmented_BF-C2DL-HSC (1)_01.mp4', 'video_segmented_BF-C2DL-HSC (1)_02.mp4', 'vi
deo_segmented_BF-C2DL-HSC_01.mp4', 'video_segmented_BF-C2DL-HSC_01_GT_SEG.mp4', 'video_s
egmented_BF-C2DL-HSC_01_GT_TRA.mp4', 'video_segmented_BF-C2DL-HSC_01_ST_SEG.mp4', 'video
_segmented_BF-C2DL-HSC_02.mp4', 'video_segmented_BF-C2DL-HSC_02_GT_SEG.mp4', 'video_segm
ented_BF-C2DL-HSC_02_GT_TRA.mp4', 'video_segmented_BF-C2DL-HSC_02_ST_SEG.mp4']

96
```

Notice from the above that the best way to find out which files belong together (automatically) is to remove the last 3 characters and check for String equality. I can also generate a label array of the 10 datasets to use later

Lets do the label array first:

In [40]:
```
desired_directory = getcwd() + "\\..\\..\\" + "Comp700_DataSets\\Extracted"

path = walk(desired_directory)

labels = []

temp = []

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

print(labels)
print()
print(len(labels))
```

```
['BF-C2DL-HSC', 'BF-C2DL-HSC (1)', 'BF-C2DL-MuSC', 'BF-C2DL-MuSC (1)', 'DIC-C2DH-HeLa',
'DIC-C2DH-HeLa (1)', 'Fluo-C2DL-Huh7', 'Fluo-C2DL-Huh7 (1)', 'Fluo-C2DL-MSC', 'Fluo-C2DL
-MSC (1)', 'Fluo-N2DH-GOWT1', 'Fluo-N2DH-GOWT1 (1)', 'Fluo-N2DH-SIM+', 'Fluo-N2DH-SIM+
(1)', 'Fluo-N2DL-HeLa', 'Fluo-N2DL-HeLa (1)', 'PhC-C2DH-U373', 'PhC-C2DH-U373 (1)', 'PhC
-C2DL-PSC', 'PhC-C2DL-PSC (1)']

20
```

We need to swap every neighbour, so that the labels with parenthesis appear first:

In [41]:
```
temp = labels.copy()
newLabels = temp.copy()

for i in range(len(temp)):
    if (i == len(labels)-1):
        newLabels[i] = temp[i-1]
    elif (i % 2 == 1):
        newLabels[i] = temp[i-1]
    else:
        newLabels[i] = temp[i+1]

print(newLabels)
```

```
['BF-C2DL-HSC (1)', 'BF-C2DL-HSC', 'BF-C2DL-MuSC (1)', 'BF-C2DL-MuSC', 'DIC-C2DH-HeLa
(1)', 'DIC-C2DH-HeLa', 'Fluo-C2DL-Huh7 (1)', 'Fluo-C2DL-Huh7', 'Fluo-C2DL-MSC (1)', 'Flu
o-C2DL-MSC', 'Fluo-N2DH-GOWT1 (1)', 'Fluo-N2DH-GOWT1', 'Fluo-N2DH-SIM+ (1)', 'Fluo-N2DH-
```

```
SIM+', 'Fluo-N2DL-HeLa (1)', 'Fluo-N2DL-HeLa', 'PhC-C2DH-U373 (1)', 'PhC-C2DH-U373', 'Ph
C-C2DL-PSC (1)', 'PhC-C2DL-PSC']
```

Cool! We can now identify which videos should be clustered together

In [42]:
```
# we need to traverse the videoFiles, looking for the
#  keywords found in newLabels

countArray = [0 for i in range(len(videoFiles))]
prevKeyword = ""; temp = ""
keywordIndex = -1

for i in range(len(videoFiles)):
    temp = videoFiles[i]

    for j in range(len(newLabels)):
        if (newLabels[j] in temp):
            keywordIndex = j
            break

    countArray[i] = keywordIndex

# simple counting algorithm - counts quantity of numbers, because sorted list
start = -1; end = -1; count = -1
quantityArray = []

for k in range(len(videoFiles)):
    if (k == 0):
        start = k
        count = countArray[k]
        continue

    if (countArray[k] != count):
        end = k
        count = countArray[k]
        quantityArray.append(end-start)
        start = k
# update at end as well
quantityArray.append( (k+1)-start)

quantityArray
```

Out[42]:
```
[2, 8, 2, 8, 2, 8, 2, 6, 2, 8, 2, 8, 2, 6, 2, 8, 2, 8, 2, 8]
```

The significance of that variable, quantityArray, is that we can use it to identity how many videos need to be stitched together.

We will then stitch the videos together, and watch the videos to identify trends in the dataset, after segmentation!

We can use the 2 variables below to find the locations of the videos

In [43]:
```
videoDestinations = getcwd() + "\\..\\..\\" + "Comp700_VideosOfSegmentation"
print(videoDestinations)

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

```
c:\Users\G5\Documents\GitHub\COMP700\..\..\Comp700_VideosOfSegmentation
['video_segmented_BF-C2DL-HSC (1)_01.mp4', 'video_segmented_BF-C2DL-HSC (1)_02.mp4', 'vi
deo_segmented_BF-C2DL-HSC_01.mp4', 'video_segmented_BF-C2DL-HSC_01_GT_SEG.mp4', 'video_s
egmented_BF-C2DL-HSC_01_GT_TRA.mp4', 'video_segmented_BF-C2DL-HSC_01_ST_SEG.mp4', 'video
_segmented_BF-C2DL-HSC_02.mp4', 'video_segmented_BF-C2DL-HSC_02_GT_SEG.mp4', 'video_segm
ented_BF-C2DL-HSC_02_GT_TRA.mp4', 'video_segmented_BF-C2DL-HSC_02_ST_SEG.mp4']
```

Now, we either need to stitch 2 videos together, 6 videos together or 8 videos together.

We can use 'quantityArray' to automate this

In [47]:
```
def bulkStitchVideos(test_directory):
    name_array =    ["vid1.mp4", "vid2.mp4", "vid3.mp4", "vid4.mp4",
                "vid5.mp4", "vid6.mp4", "vid7.mp4", "vid8.mp4"]
    videoIndex = -1; count = 0

    # use quantityArray to stitch the videos together
    for a in quantityArray:

        # print(a)
        for b in range(a):
            videoIndex += 1
            c = VideoFileClip(videoDestinations + "\\" + videoFiles[videoIndex])

            # 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[b])

            # new_clip.ipython_display(width=100)

        if (a == 2):
            a = VideoFileClip(test_directory + "\\" + name_array[0])
            b = VideoFileClip(test_directory + "\\" + name_array[1])

            # now, stitch together!
            stitched_video = clips_array([[a, b]])
        elif (a == 6):
            a = VideoFileClip(test_directory + "\\" + name_array[0])
            b = VideoFileClip(test_directory + "\\" + name_array[1])
            c = VideoFileClip(test_directory + "\\" + name_array[2])
            d = VideoFileClip(test_directory + "\\" + name_array[3])
            e = VideoFileClip(test_directory + "\\" + name_array[4])
            f = VideoFileClip(test_directory + "\\" + name_array[5])

            # now, stitch together!
            stitched_video = clips_array([[a, b, c], [d, e, f]])
        elif (a == 8):
            a = VideoFileClip(test_directory + "\\" + name_array[0])
            b = VideoFileClip(test_directory + "\\" + name_array[1])
            c = VideoFileClip(test_directory + "\\" + name_array[2])
            d = VideoFileClip(test_directory + "\\" + name_array[3])
            e = VideoFileClip(test_directory + "\\" + name_array[4])
            f = VideoFileClip(test_directory + "\\" + name_array[5])
            g = VideoFileClip(test_directory + "\\" + name_array[6])
            h = VideoFileClip(test_directory + "\\" + name_array[7])

            # now, stitch together!
            stitched_video = clips_array([[a, b, c, d], [e, f, g, h]])
        else:
            print("Invalid number of videos specified")
            break

        stitched_video.write_videofile(test_directory + "\\" + newLabels[count] + ".mp4"
```

```
        resized_clip.close()
        try:
            a.close(); b.close(); c.close(); d.close()
            e.close(); f.close(); g.close(); h.close()
        except:
            pass

        stitched_video.close()

        count += 1

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

In [46]:

```
test_directory = "009_Segmentation_Videos"

tryMakeDirectory(getcwd(), test_directory)

path = walk(test_directory)

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

if ( not ( len(fileCollection) == 20 ) ):
    bulkStitchVideos(test_directory)
else:
    print("Videos Already created!")
```

```
Videos Already created!
```

From the 20 videos generated, we can see a trend:

The videos with a (1) attached are challenge sets - which contain training sets and no test sets

The other videos contain 6 or 8 folders - 2 training sets, 2 or 4 test sets for segmentation and 2 test sets for tracking

	Challenge Sets		6/8 Sets
A(1)	Left video has some some pulsating frames, Right video mainly white	A	Top-Left video has some pulsating frames
B(1)	Left video has some some pulsating frames	B	Top-Left video has some pulsating frames, Far left videos are a bit 'soft'
C(1)	Both Left and Right video has some pulsating frames	C	Far left videos are a bit 'soft'
D(1)	Right video has some Dark Spots	D	
E(1)		E	
F(1)		F	
G(1)		G	
H(1)		H	
I(1)		I	
J(1)	Both videos have some harsh light	J	Far left videos have some harsh light

The other videos are reasonably well done!

What we can try next is to implement an idea explored in the following video:
https://www.youtube.com/watch?v=jvZm8REF2KY

The author uses masks to train a Neural Network, using U-net Architecture.

We can attempt to generate our own masks, by scanning the Test Images and looking for patterns! This may not work, if the colours are assigned randomly, but it is worth a try. We will explore this in another notebook