In [1]:
```
import cv2
import numpy as np
import os

cap_video = cv2.VideoCapture('example.mp4')

try:
    if not os.path.exists('frames'):
        os.makedirs('frames')
except OSError:
    print ('Error while creating directory.')

currentFrame = 0
frame_array = []
while(True):

    ret, frame = cap_video.read()
    if not ret: break

    name = './frames/frame' + str(currentFrame) + '.jpg'
    print ('Creating            =' + name)
    cv2.imwrite(name, frame)
    img = cv2.imread(name)
    frame_array.append(img)
    currentFrame += 1

height, width, layers = frame_array[0].shape
size = (width,height)
fps = cap_video.get(cv2.CAP_PROP_FPS)
out = cv2.VideoWriter('recheck.mp4',cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
for i in range(len(frame_array)):
    out.write(frame_array[i])
out.release()


cap_video.release()
cv2.destroyAllWindows()
```

In [ ]:
```
#the code run perfectlyt to create a directory for the broken down frames from the video and then comparing the
#frames based on a certain formula/logic through which the code eliminates frames which are the same. Now, the uneli
minated
#frames are sequenced/combined in order to create the required video.
import cv2
import numpy as np
import os
from skimage import measure
import matplotlib.pyplot as plt

cap_video = cv2.VideoCapture('example1.mov')
frame_array = []

def convert_frame():
    try:
        if not os.path.exists('frames1'):
            os.makedirs('frames1')
    except OSError:
        print ('Error while creating directory.')
    currentFrame = 0
    while(True):

        ret, frame = cap_video.read()
        if not ret: break

        name = './frames1/frame' + str(currentFrame) + '.jpg'
        print ('Creating            =' + name)
        cv2.imwrite(name, frame)
        currentFrame += 1

    for i in range(1,currentFrame-1):
        p_name = "./frames1/frame{}.jpg".format(i)
        nf_name = "./frames1/frame{}.jpg".format(i+1)
        compare_images(p_name,nf_name)
    height, width, layers = frame_array[0].shape
    size = (width,height)
    fps = round(cap_video.get(cv2.CAP_PROP_FPS))
    out = cv2.VideoWriter('recheck1.mp4',cv2.VideoWriter_fourcc(*'DIVX'), fps, size)
    for i in range(len(frame_array)):
        out.write(frame_array[i])
    out.release()


    cap_video.release()
    cv2.destroyAllWindows()

def compare_images(imageA, imageB):

    first_frame = cv2.imread(imageA)
    next_frame = cv2.imread(imageB)
    imageA1 = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)
    imageB1 = cv2.cvtColor(next_frame, cv2.COLOR_BGR2GRAY)
    mse = np.sum((imageA1.astype("float") - imageB1.astype("float")) ** 2)
    mse /= float(imageA1.shape[0] * imageA1.shape[1])

    ssim = measure.compare_ssim(imageA1, imageB1)
    if mse <= 155.0 and ssim >= 0.66: #these values can be changed based on trial and error of the actual camera
        img = cv2.imread(imageB)
        frame_array.append(img)
    #to confirm and see the frames which are been compared based on the mse and ssim values.
#         fig = plt.figure("Compare")
#         plt.suptitle("MSE: %.2f, SSIM: %.2f" % (mse, ssim))

#         ax = fig.add_subplot(1, 2, 1)
#         plt.imshow(imageA1,cmap = plt.cm.gray)
#         plt.axis("off")

#         ax = fig.add_subplot(1, 2, 2)
#         plt.imshow(imageB1,cmap = plt.cm.gray)
#         plt.axis("off")

#         plt.show()
    else:
        frame_array.append(first_frame)
        frame_array.append(next_frame)

convert_frame()
```

In [ ]:
```

```

In [ ]:
```

```