Class Project – Part 1 (240 pts)

The goal of this project is to use data analytics to detect if a train is moving right or left from a streaming video.

Task 0: OpenCV is great ! (10pts)

Let's to back to our favorite character, Mario!
This time we are going to apply thresholding with openCV library function threshold.

To open the *png* file you will use the function from the cvs library:
`imread({FileName},{reading_mode}).`

`reading_mode:`
```
cv2.IMREAD_COLOR : Loads a color image. Any transparency of
image will be neglected. It is the default flag.
cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
cv2.IMREAD_UNCHANGED : Loads image as such including alpha
channel
```

Then you will apply the *`threshold`* function.
`threshold(`*`{Image_Descriptor}`*`,127,255,cv2.THRESH_BINARY)`

You will display the image by using the function *`imshow`*.
`cv2.imshow('Window_name',{Image_to_display})`

PS: to display the image you may need this function:
```
waitKey(0)
destroyAllWindows()
```

Task 1: Preliminary understand about video processing (30pts)

The goal is to remove the details of a video that are useless for your analysis. Usually some backgrounds are moving, are alive, if you don't need to analyze them, we will remove them. Without all these details, we will be able to focus on the big picture.

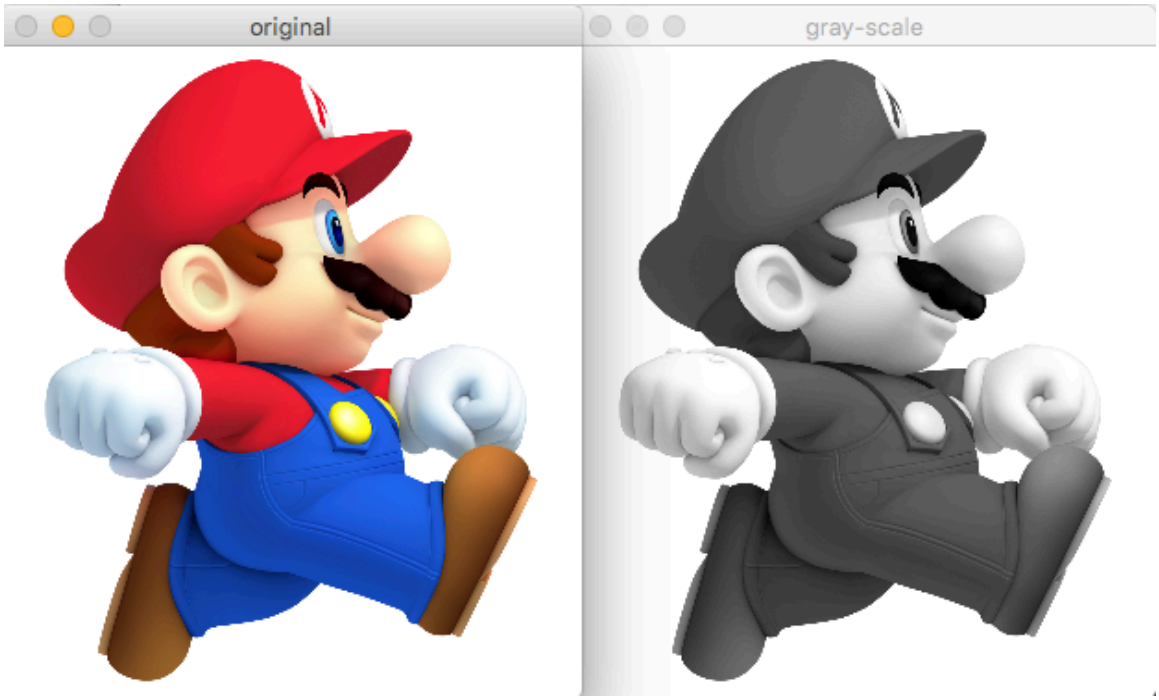
First, we need to define the stationary background in the video.
Then, we will compare each new frame to identify areas of change (we can call ROI, Region of Interest). This method is called the background subtraction.
We will be using the method of frame differencing.

To detect motion in a video, the color doesn't really matter, therefore we can convert the video to gray-scale by using the function

```
cvtColor (frame, cv2.COLOR_BGR2GRAY)
```

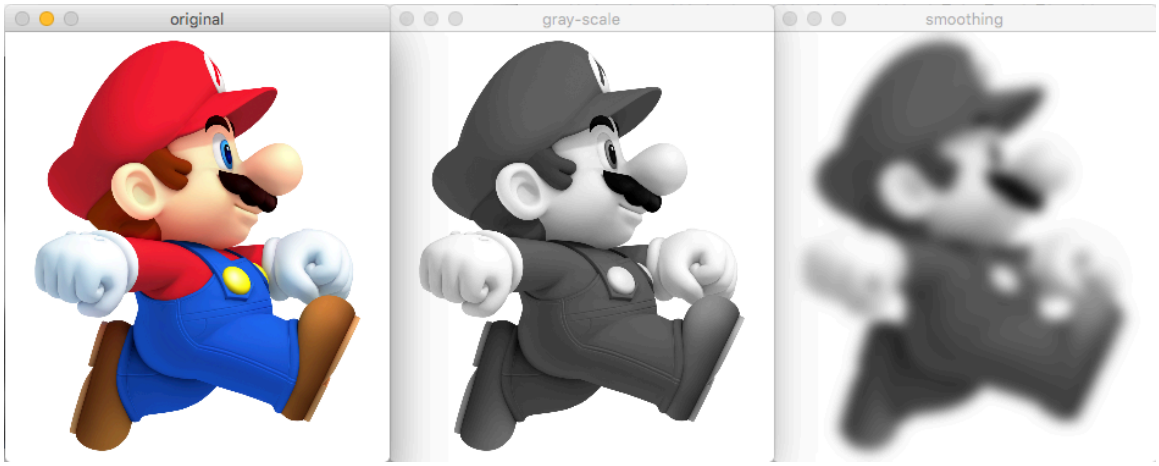
You can apply this function to Mario.

Task 1-a: Display Mario in Gray-Scale (10pts)



Task 1-b: Reducing the background noise of a video or small movements (such as leaves of a tree) (10pts)

To reduce the image of a video, we will apply the Gaussian Blur method (or Gaussian smoothing):
https://en.wikipedia.org/wiki/Gaussian_blur

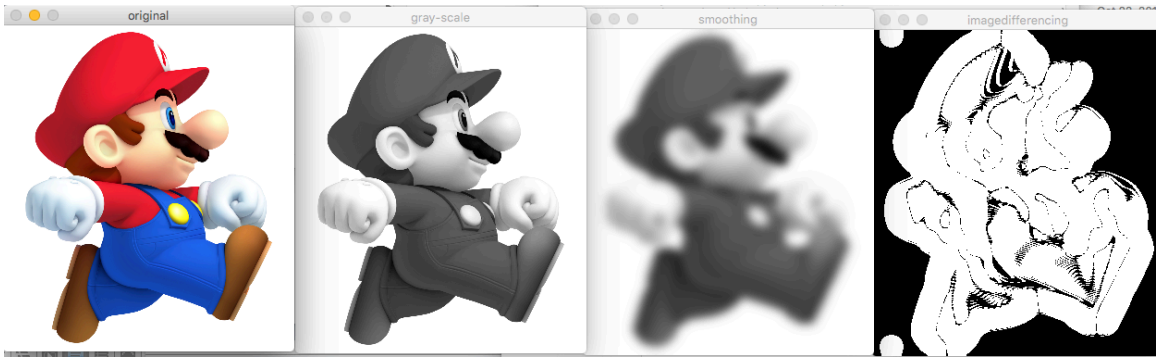


You will use the function
Choose the X value you prefer.

```
GaussianBlur({gray-scale-picture}, (X, X), 0)
```

Task 1-c: Image differencing (10pts)

You will differentiate the previous frames with the current frame.
You can apply the function absdiff to the same image but you will get just a black image since there is no movements. That's why you will use the image 'mario2.png'.
absdiff(np.float32({smooth_mario}), np.float32({smooth_mario2}))



Task 2: Something is moving on the video – Part 1 (50 pts)

This part is based on the excellent tutorial about openCV by SVDS.
http://cmawer.github.io/trainspotting/trainspotting-blog.html
(Please you can get inspired by the structure but DO NOT copy their functions because their method is different from what it is suggested in this Class Project)

The goal of this Task is to detect movement on the video.
You are going to use the previous method and calculate the ratio number of pixels.

I am giving the different steps you will need to have in your code:

#Open a video descriptor using the command
feed_descriptor=cv2.VideoCapture('train_training.mp4')

#Read all the frames from the feed
while (feed_descriptor.isOpened()):
  #Read frame by frame
  current_frame = feed_descriptor.read()[1]

  # Check if current_frame is not different from None
  # If it is None, just break the loop

  # Convert your frame into gray-scale color like we did with Mario

 # Apply Gaussian Blue like we did with Mario

 # Add current frame to running average after
 # This part is to give more important to the most recent frames but keeping in memory the past ones.
You can use the following parameter:
`alpha = 0.02`
when `running_avg` is None, you can assign
`np.float32(smooth_frame)`
`accumulateWeighted(np.float32(smooth_frame), running_avg, alpha)`

Find absolute difference between the current smoothed frame and the running average
You apply the function `absdiff` between the smoothed frame and the running average

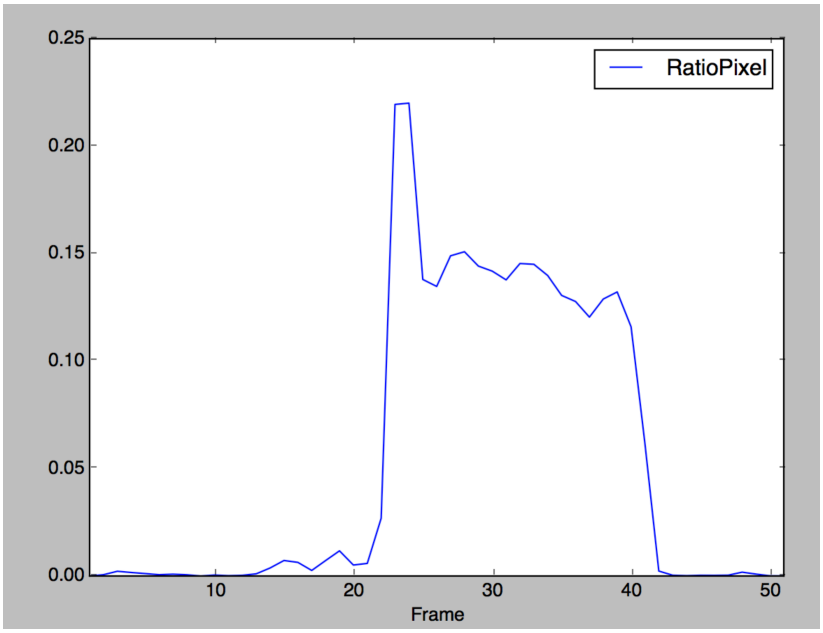
Finally you can apply the threshold function

Task 3: Something is moving on the video – Part 2 (50 pts)

The Task 3 gave us the code structure. We can start using some data analysis.
For each iteration of the loop, you can calculate the number of points that have been changed between two frames. You will divide this value by the total number of pixels. For this part, you can use the following ratio:

```
p.count_nonzero(subtracted)/subtracted.size
```

You will store this number for each frame. Then you will plot the number based on the frames.
You should find the following plot:

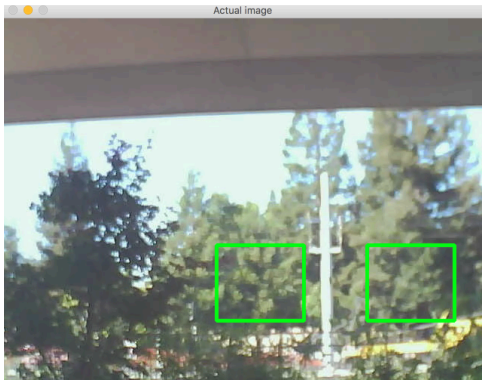


You can now determine when a train passes on the video.

Task 4: Where this train goes? (50pts)

Task 4-a: Define ROIs (20pts)

You will define two ROIs (Region of Interest) on the video to determine the direction of the train. You will display them on the video.



You can use the function

```
rectangle(current_frame,(X1,Y1),(X2,Y2),(0,255,0),3)
```

Task 4-b: USE ROIs and Apply the same function to detect movement (30pts)

Since you find the pixels that can help you to know the direction of the train. You will apply the function you previously use to see if there is a movement in one given ROI. If there is first a movement in the Left ROI, then a movement in the Right ROI, you can deduce that the train goes from Left to Right.

Task 5: For once, the word training set really means something (50pts)

You have now a code that can detect a train and its direction. Your model is ready to be tested on the video train_testing.mp4. For this video, you should change the ROIs since the angle is slightly different. This video has a lot trains and in different direction. Let's see if your code manages to find the number of the trains and their direction. You may need to play with the parameters.