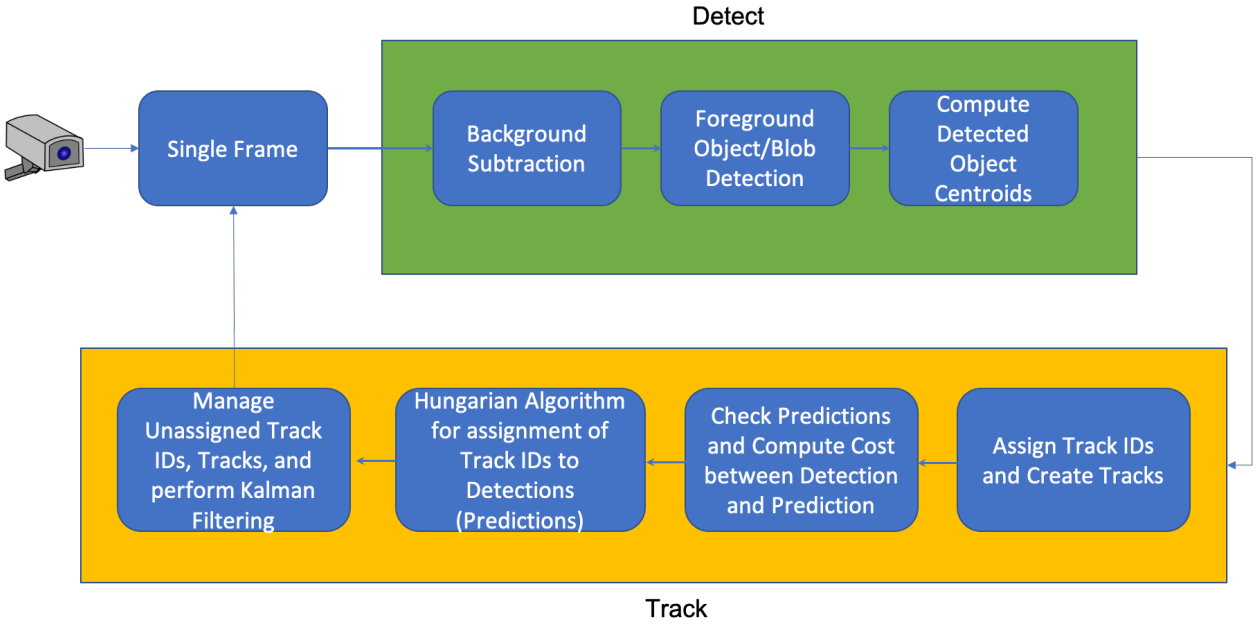
HomeWork-4: Object Detection and Tracking

Due Date: 11:59 PM, 6th May 2020

Object detection and tracking is a common requirement for a range of applications in computer vision. In this assignment, we will leverage the broad understanding we have developed throughout this semester, specifically as it related to background modeling for foreground object detection, Kalman filtering for prediction of object location, and jointly, for object tracking.

This homework assignment will focus on the design of an object tracking solution that is implemented under a 2 module framework; ***a module for detection of foreground objects, and a module for tracking individual detected objects while ensuring consistent object id association using distance as a metric.***

The overall framework can be seen in the figure below:



You are given a solution template that includes the main function that implements the overall object tracking solution as shown above. In addition, the **Track** module is also available that already includes implementation of the Kalman filter that uses the centroid of detected objects as its state parameter. The focus of this assignment will be on implementing the **Detect** module. In doing so, you will also be required to implement appropriate **Background Modeling** approach to perform Background Subtraction and Foreground Object Detection. Specific assignment steps are:

1. **Background modeling (5 Pts.)** - In the given BGModel() class (background_model.py), you are to implement the following background subtraction methods:

a. Background model estimated by computing the mean value of observed image intensity for each pixel
b. Background model estimated by computing the median of observed image intensity for each pixel
c. Background model estimated by modeling each pixel's intensity as a Gaussian (mean and variance)
d. Background model estimated by modeling each pixel's intensity as a Mixture of Gaussians (can use cv2. createBackgroundSubtractorMOG2())
e. Background model estimated by modeling each pixel's intensity based on Nearest Neighbor estimate of observed intensities (can use cv2. createBackgroundSubtractorKNN())

Note that the specific computation of the foreground image after background subtraction should be performed in the compute_fgmask() method

2. **Object detection (20 Pts.)** - In the given Detectors() class (detectors.py), you are to implement approach to detect objects for tracking and their centroid values. The detector should allow for use of one of the background subtraction methods and get the foreground image. This should be further processed to detect objects worthy of tracking. Your choice of processing to detect objects and compute their centroids should be implemented in the Detect() method within the class. (Please note that this does not require classifier based detection of objects)

   You should not modify the following files:
   cv_hw4.py, kalman_filter.py and tracker.py files.

3. **Report (5 Pts.)** - Write a report outlining your approach for background subtraction and object detection.

General Instructions:

1. Submit code on Github
2. All coding must be done using **python**
3. Use tools available in OpenCV.
4. Your solution needs to ensure that it executes using the following commands:

```
python cv_hw4.py -v QIL_orig.mp4 -b mean -d 0
python cv_hw4.py -v QIL_orig.mp4 -b median -d 0
python cv_hw4.py -v QIL_orig.mp4 -b gaussian -d 0
python cv_hw4.py -v QIL_orig.mp4 -b mog -d 0
python cv_hw4.py -v QIL_orig.mp4 -b knn -d 0
```

5. Result image will be written to the folder *output/*.

Submission Instructions:

1. Submit only the source code files.
2. Include a readme file to describe any required set up, like location of image files that was hard coded, or any other information that is required to run the code
3. Make sure your final submission is running on circleci. The TA will use CircleCI output and your github code for grading. TA will not be able to grade if the code does not run on circle CI.

   Common reasons for failure:

- Do not use any 3rd party libraries or functions.
- Do not display images in your final submission. Example, cv2.imshow(), cv2.waitkey(), cv2.NamedWindow will make the circle ci fail.
- Do not read or write images with cv2.imread and cv2.imwrite, make sure you delete them in the final submission.
- Files not to be changed: requirements.txt and .circleci directory,