Objectives

- Design a C/C++ or Python program that extracts and counts moving objects, e.g. people, cars and others using background modelling and detects pedestrians using cascades fully body detector.

Introduction

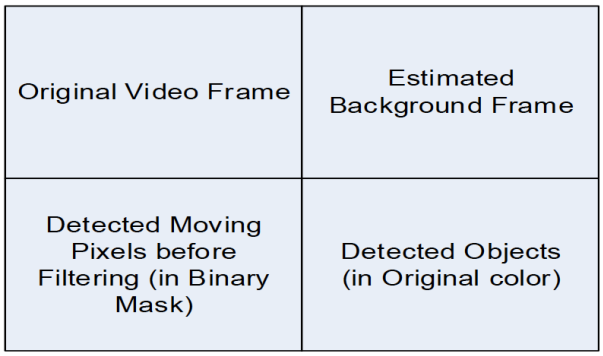
Extraction of moving objects and detection of pedestrians from a sequence of images or video is often used in many video analysis tasks. For instance, it is a key component in intelligent video surveillance systems and autonomous driving. In this assignment, you are required to develop a program in C/C++ or Python using OpenCV 4.5.2 to detect, separate and count moving objects from a given sequence of images or video captured by a stationary camera and to detect pedistrians. There are two tasks.

Task One (15%) – Background modelling

In this task, you are required to extract moving objects using Gaussian Mixture background modelling. There are three key steps involved in the extracting and counting moving objects:

1. Detecting moving pixels using background modelling and subtraction,
2. Removing noisy detection using morphological operators or majority voting and
3. Count separate moving objects using connected component analysis.
4. Classify each object (or connected component) into person, car and other by simply using the ratio of width and height of the connected components.

OpenCV 4.5.2 provides various algorithms for each of the steps 1 & 2. However, you MAY have to implement your own connected component analysis algorithm and classification algorithm. For simplicity, *you can assume that each connected component corresponds to one object*.



When running, the program should display the original video frame, estimated background frame, detected moving pixels after the background modeling and subtraction (before any noise removal) and the detected moving objects in a **single window** as illustrated above. The detected moving pixels before filtering should be displayed in black and white (binary mask). The detected object has to be displayed in its original RGB color (all background pixels should be displayed in black). At the same time, the number of objects or connected components should be output to the command window as illustrated below:

```
Frame 0001: 0 objects
Frame 0002: 0 objects
…
Frame 0031: 5 objects (2 persons, 1 car and 2 others)
Frame 0032: 6 objects (3 persons, 1 cars and 2 others)
…
Frame 1000: 10 objects (
…
```

Task Two (10%) – Detection and Tracking of Pedestrians

In this task, you are required to detect pedestrians using a OpenCV Haar Cascade detector for fully body, to track and display the detected pedestrians by providing same labels, i.e. $1, 2, \cdots, n$, to the same pedestrians across over times and to select the **(3)** pedestrians that are most close to the camera. Solutions to the tracking and selection of up to three **(3)** most close pedestrians must be described in the comments at the beginning of your source C/C++ or Python code. Note a trained haar cascades detector can be found in the subdirectory ".\etc\harcascades\haarcascade_fullbody.xml"

Original Video Frame	Video Frame with Overlapped Detected Bounding-Box
Video Frame with Detected and Tracked (Labelled) Bounding-Boxes	Video Frame with up to 5 Detected and Tracked Bounding-Boxes that are most Closed to the Camera

The program should display the original video frame, video frame with overlapped bounding-boxes of the detected pedestrians, video frame with detected and tracked bounding-boxes and video frame with up to three **(3)** closest objects to the camera. Display must be in a **single window** as illustrated above.

Requirements on coding

1. The program should be named as "**movingObj**" and shall take an option, either `-b` or `-d` and a video filename as the input, e.g. `movingObj -b videofile` or `movingObj -d videofile`. When `-b` is given, the program should perform Task One and when `-d` is given, the program performs Task Two.

2. No other third-party libraries should be used in the program except OpenCV 4.5.2 or OpenCV 4.5.2-Python (assuming that `numpy` and `matplotlib` packages exist). The code has to be either in C/C++ or Python.
3. Place your implementation in a single `.cpp` or `.py` file called `movingObj.cpp` or `movingObj.py`
4. Description of the solution to Task Two should be given as comments at the ***beginning*** of the `movingObj.cpp` or `movingObj.py`
5. No other third-party libraries should be used in the program except OpenCV 4.5.2. The code has to be in either C/C++ or python-OpenCV 4.5.2.

Marking Scheme

Zero marks may be graded if your code cannot run or the code does not meet the requirements

Task One

1. Program structure, comments and usability (1%)
2. Read and display of the video frames (2%)
3. Background modeling or estimation (2%)
4. Subtraction and display of the detected moving pixels (2%)
5. Remove of noisy or false detection (2%)
6. Connected component analysis and display of the moving objects (2%)
7. Classification of moving objects (2%)
8. Output number of objects or connected components (2%)

Task Two

9. Description of the solution (1%)
10. Display of the detection results (in bounding-boxes) (3%)
11. Display of tracking results (3%)
12. Display of up to three tracked pedestrians that are most closed to the camera (3%)