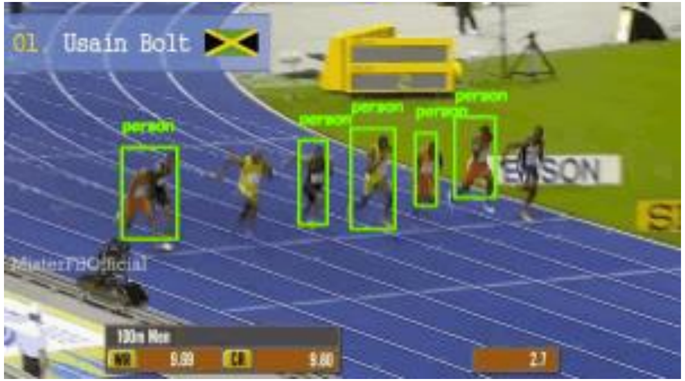
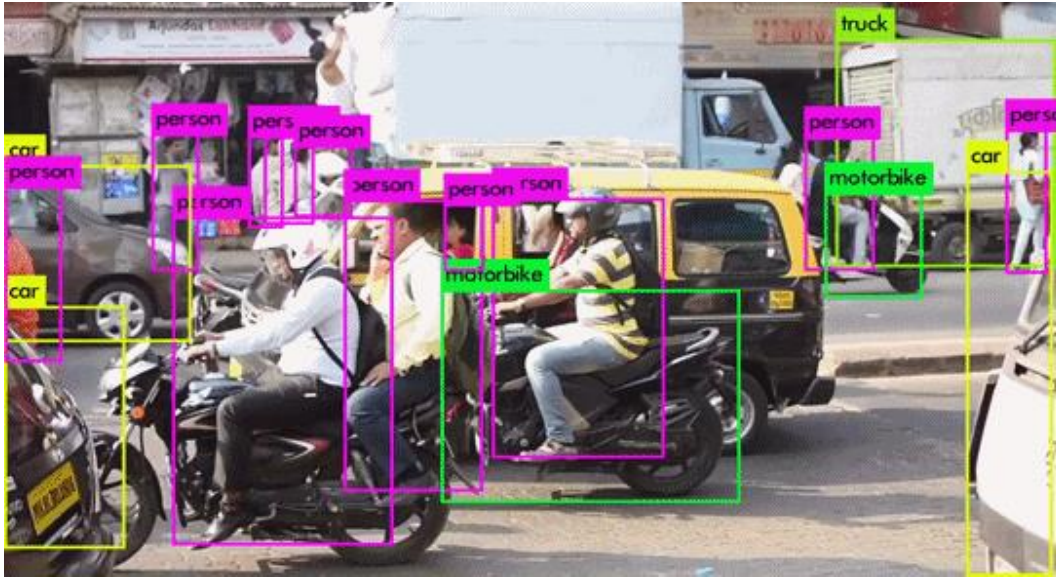
Internship Report

Research Subject

Research subject

Real-time multi-object detection and tracking while preserving objects' IDs using (Detection, Tracking and Matching).

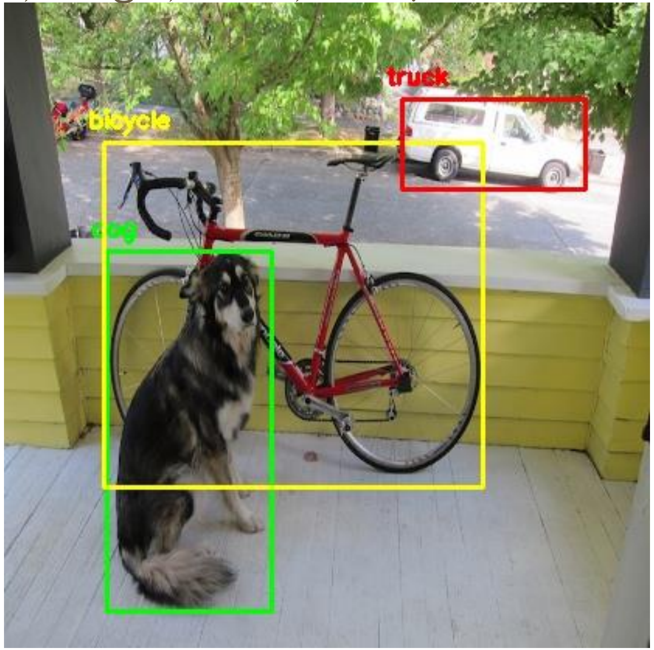


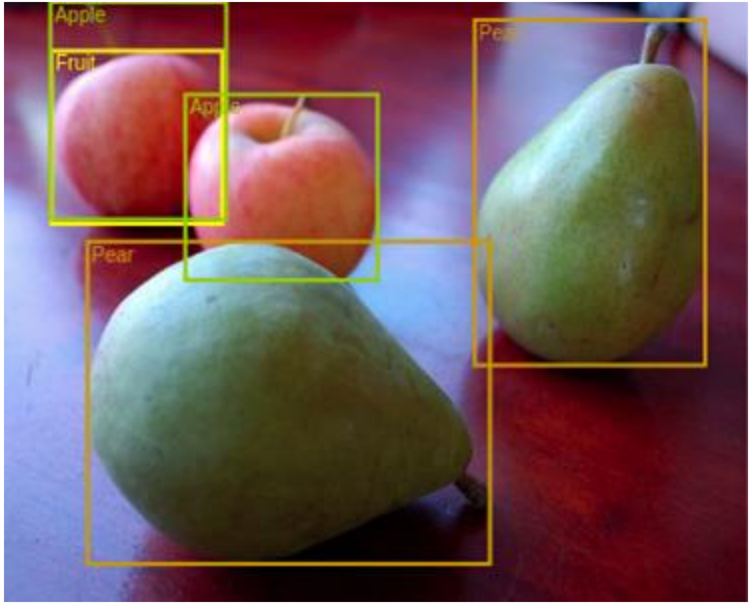


Definitions

Object detection

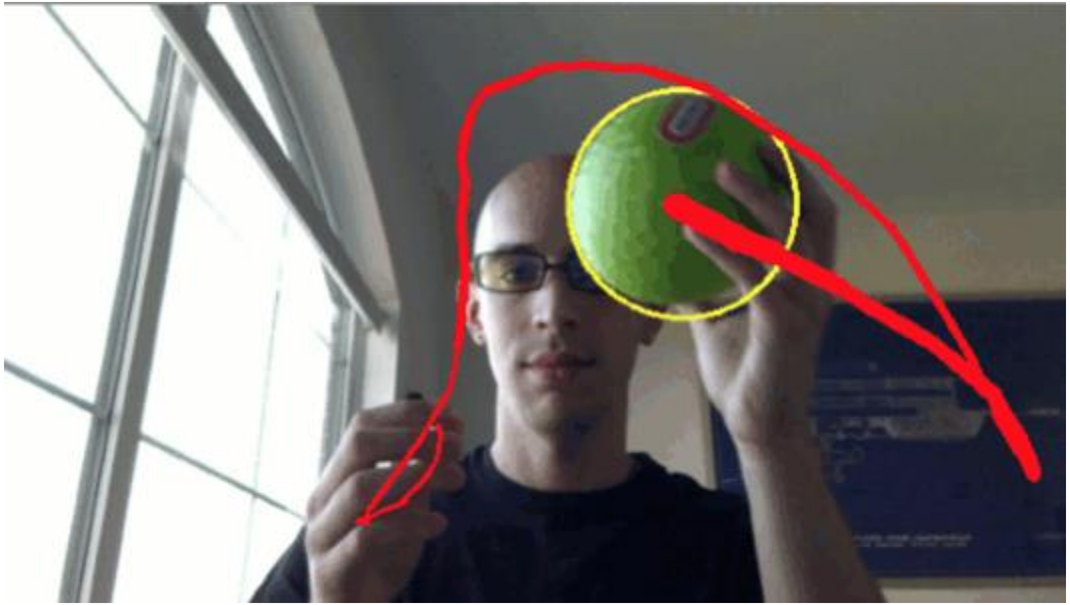
Detecting objects is to recognize and locate them within a frame (objects such as human, dogs, cats, etc.).





Tracking

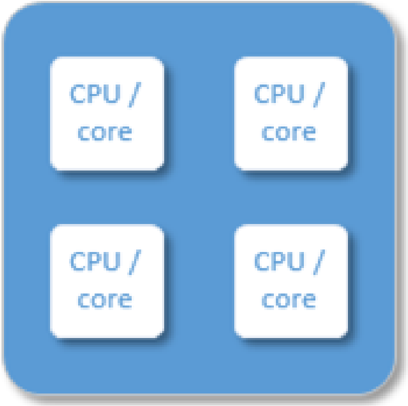
Track a specified object over a sequence of images.



Parallel Programming

Time is important!

Parallel programming is to save time by dividing the work between more than one processing unit (core).

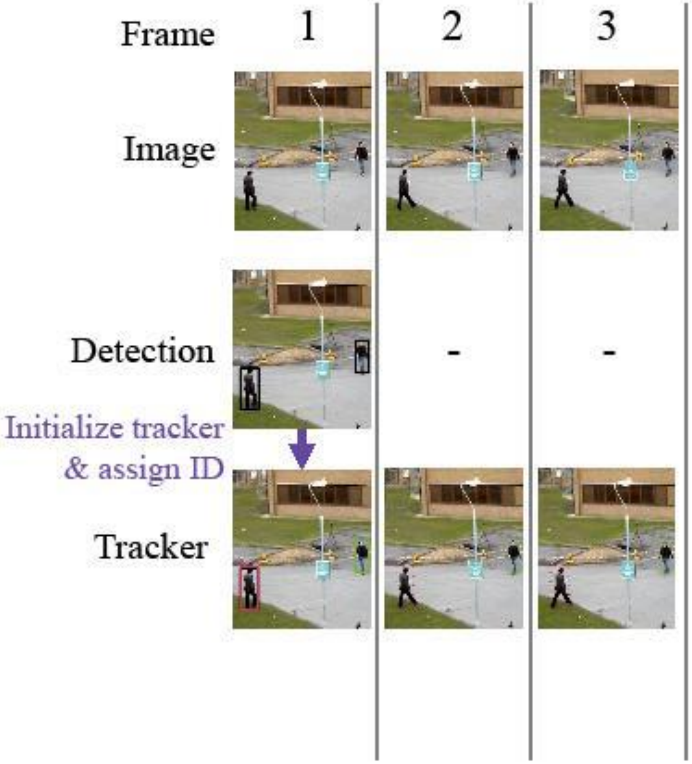


Why multi-object is hard to achieve?

- Object Occlusion
  - Hard to preserve IDs if objects overlapped
- Variations in object shapes
  - Object shape and brightness may change over the frames
- Heavy computations
  - Hard to achieve good accuracy
  - Hard in real time
  - Balancing speed with accuracy is challenging so there is a trade-off

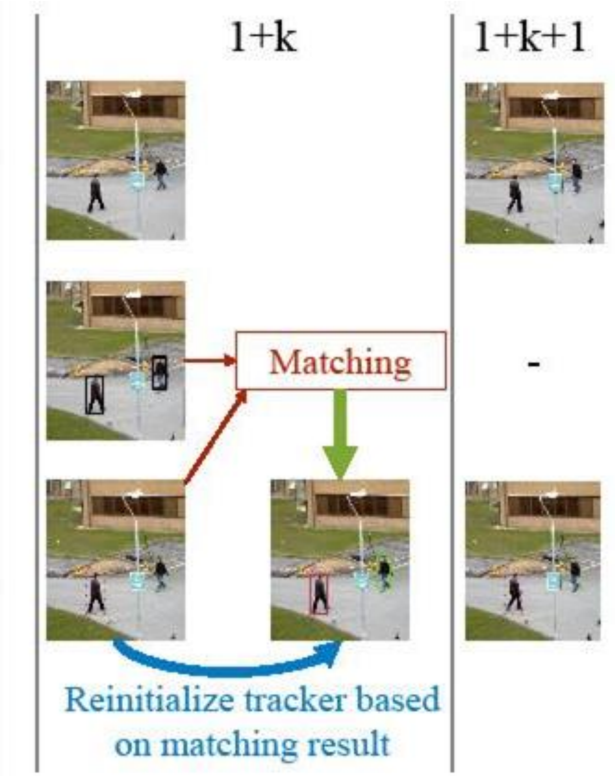
What is the plan?

- Detection for the first frame only.
- Tracking for following k frames using **KCF tracker**.
- Increasing speed of tracking using parallel programming.

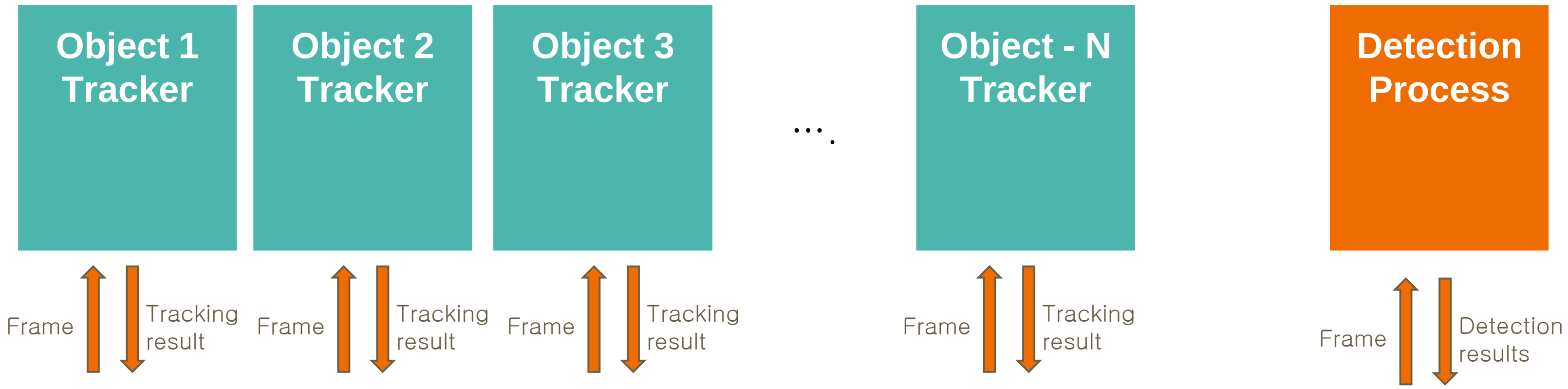


What is the plan?

- Detection again for the frame K.
- Match results from detection to results from tracking using **Hungarian algorithm**.
- Track again for another K frames.
- Repeat previous steps



Parallel Design



<u>Main program</u>

- **First frame:**
  - Create detection process - Get first detection results - Create a child Tracker for every object
- **For every frame:**
  - Get new frame from Camera - Send frames to child trackers
  - Draw detection results (If tracking has not finished draw the nearest result)
- **After K frames**
  - **Send frame for the detection process**

Parallel Design results

Computer Specifications

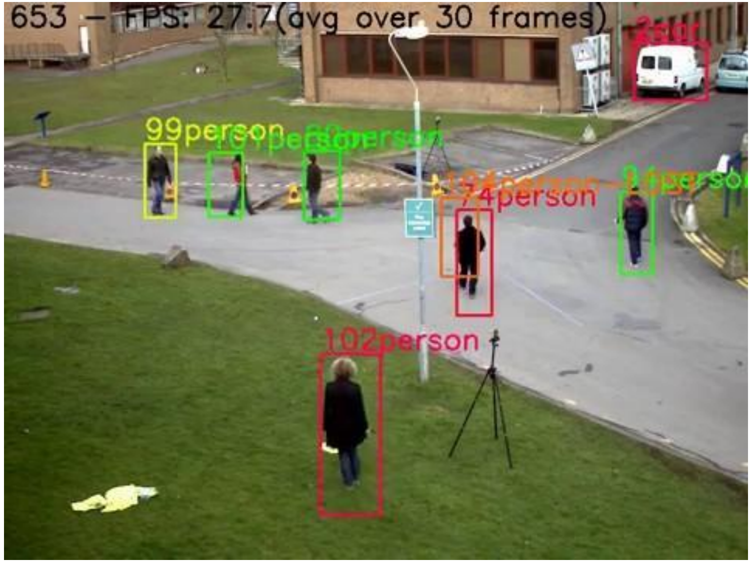
RAM: 32 GB

Processor: Core i7-5960X

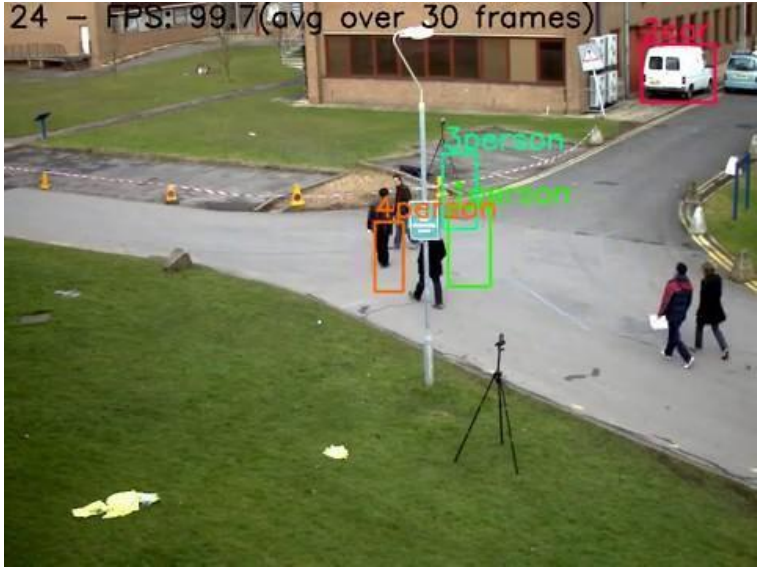
GPU: Nvidia GTX Titan X

PETS09-S2L1 Dataset - using GPU

Original
frame_per_second = 24.0

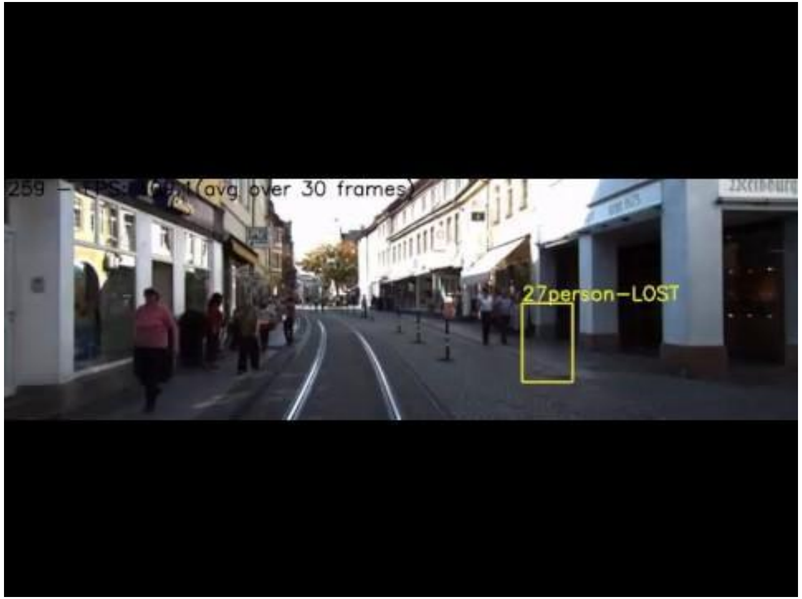


Parallel
frame_per_second = 12.6

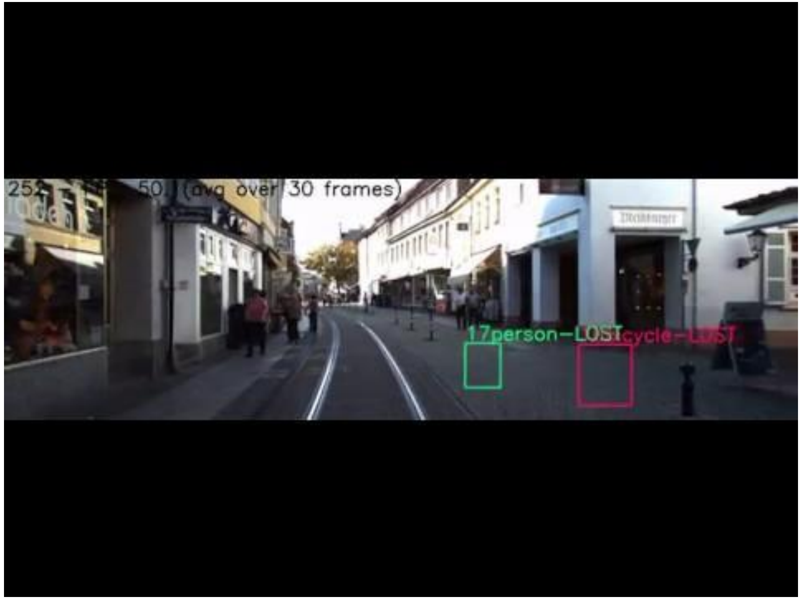


KITTI-13 Dataset - using GPU

Original
frame_per_second = 57.7



Parallel
frame_per_second = 39.0



KITTI-17 Dataset - using GPU

Original
frame_per_second = 50.7

Parallel
frame_per_second = 32.7

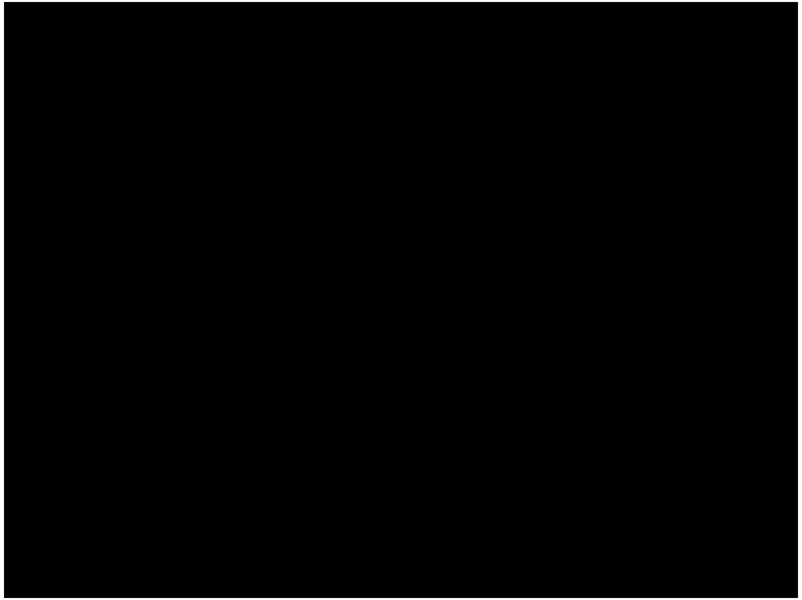
78-00 Dataset - using GPU

Original
frame_per_second = 13.8

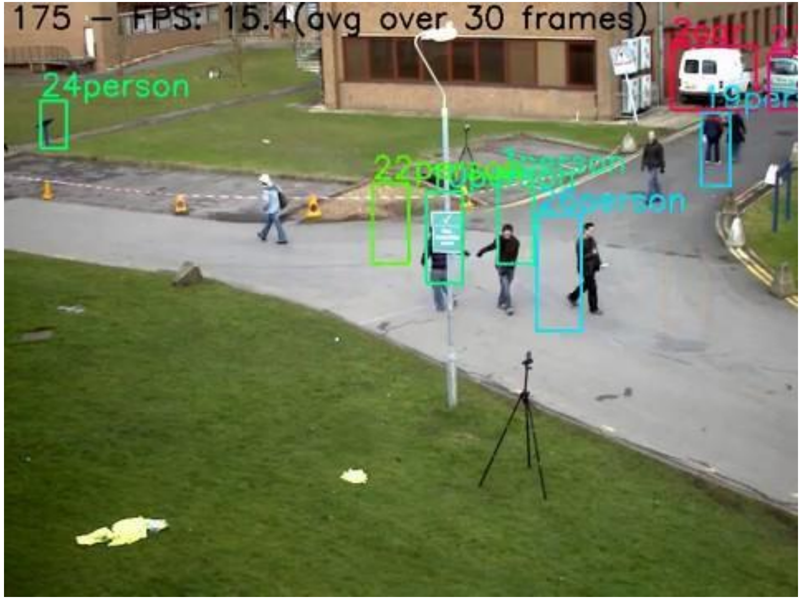
after:
frame_per_second = 5.8

PETS09-S2L1 Dataset - without GPU

Original
Frame_per_second = 9.16

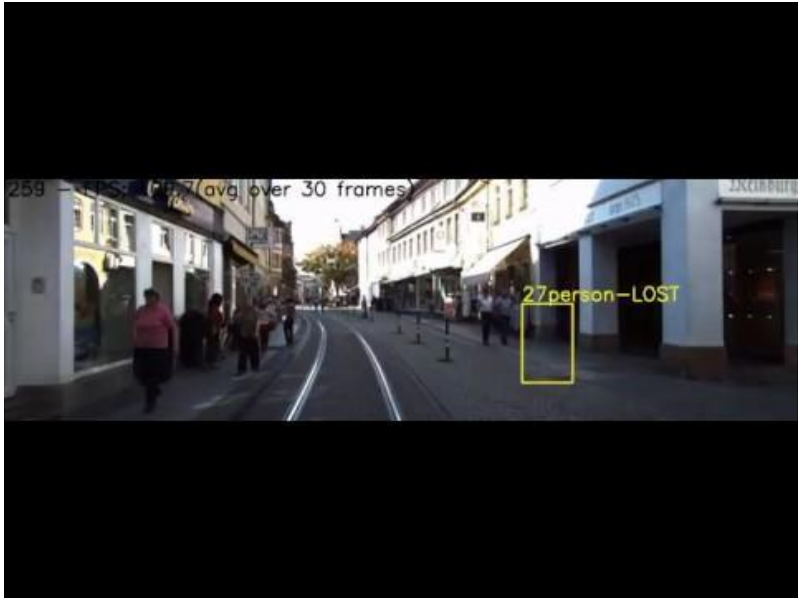


Parallel
Frame_per_second = 13.05



KITTI-13 Dataset - without GPU

Original
Frame_per_second = 12.19



Parallel
Frame_per_second = 76.2

KITTI-17 Dataset - without GPU

Original
Frame_per_second = 11.95

Parallel
frame_per_second = 66.8

78-00 Dataset - without GPU

Original
Frame_per_second = 7.22

Parallel
frame_per_second = 8.2

Notes on results

- In experiments with GPU, Sequential results is faster because GPU has huge computational power so, parallelization may have some overheads that makes it not best option when the used GPU is powerful.
  - *Implementing parallel in limited resource device (such as NVIDIA TX2) may increase the speed of multi-object detection*
- In experiments without GPU, Parallel code is faster and more reliable to use as not every devices can have GPU to fast the program. As detection is running in parallel we don't have to wait until detection produce some results.

Parallel programming is not always the best option otherwise we would have used parallel in all programs!