

Geometric Image
Processing Lab

People Counting System

By: Ido Galil, Or Farfara

Supervisor: Yaron Honen

Sponsor: Technion's libraries



Problem Statement

- This project comes to supply the Technion's libraries with a people counting system, to monitor the amount of people in the libraries at any given time.
- This goal is achieved by the combination of two components: a detector component using a Convolutional Neural Net (YOLO) detecting people on the frame, and a tracking component utilizing a tracking algorithm (CSRT) which updates those people positions on the next frames.



Project Scope

Background

- In the near past, the ability of identifying people and tracking them in a video and doing so in real-time was very challenging: it required a lot of computational resources and was immensely hard to develop.
- With the rapid development of neural networks in recent years, this task became more plausible for general use, and could be applicable to solve many problems.



Project Scope

Background

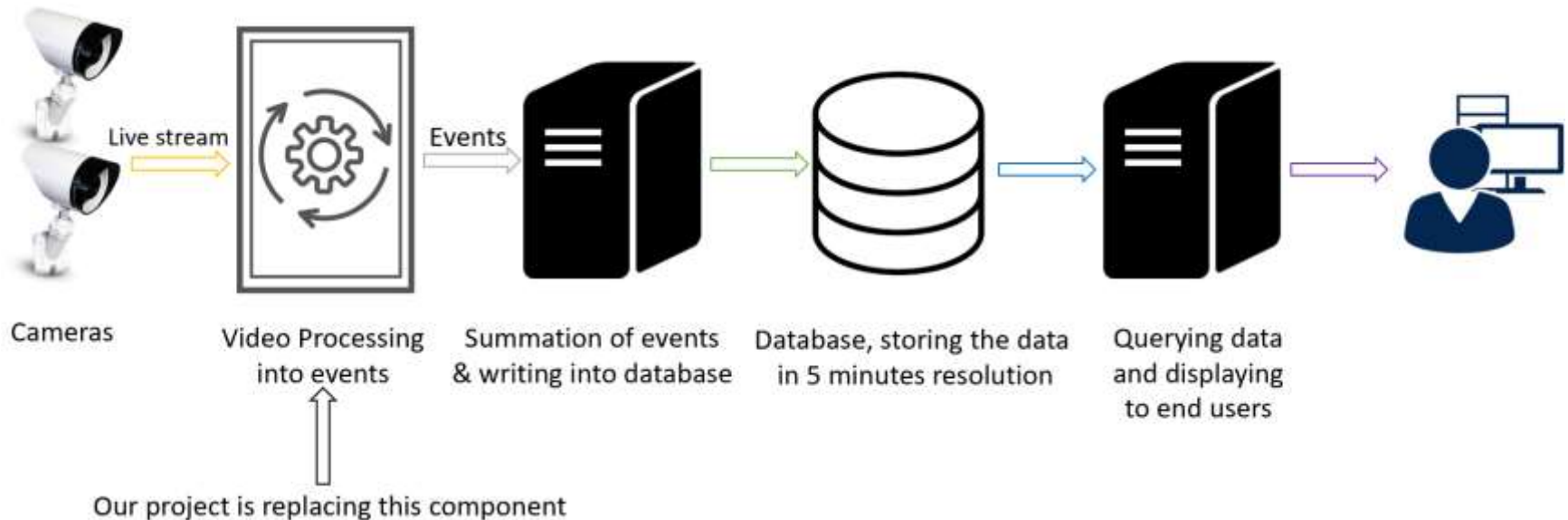
- One such problem is counting the amount of people entering or exiting a building through specific entrances.
- A counting system intended for this problem could add value for various different scenarios (knowing the amount of people in a building in case of an emergency, allocating an area's resources according to how populace each building or floor in it is etc...).



Project Scope

Motivation

- The library today performs the people counting task using the following architecture:



Project Scope

Motivation

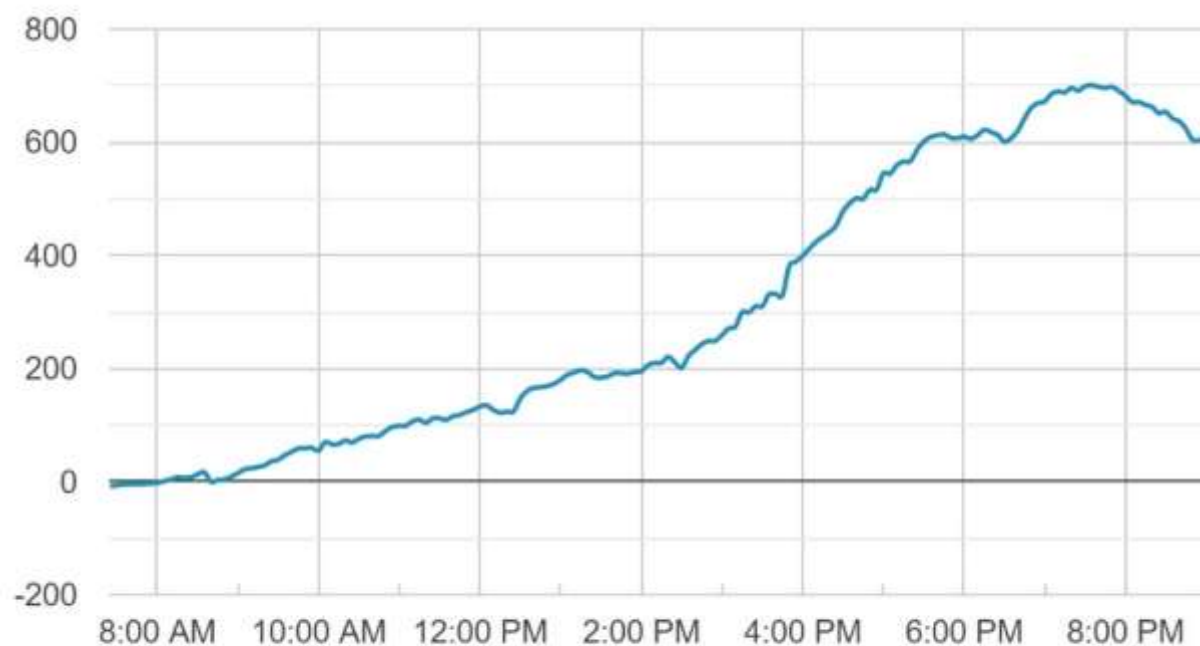
- 1) IP cameras are located overhead in the libraries entrances and send live stream video
- 2) A video processing component (already existing today) gets this live feed from the cameras, processes it and translates it into count events: ins and outs.
- 3) The events are sent to a summation system, that sums the ins and outs and sends them to the database every 5 minutes
- 4) The database stores the data in 5-minute resolution
- 5) The data is made available for queries and displayed to the end users



Project Scope

Motivation

- The current state of the architecture suffers due to large inaccuracies in its counting capabilities, as can be shown in this daily counting graph:



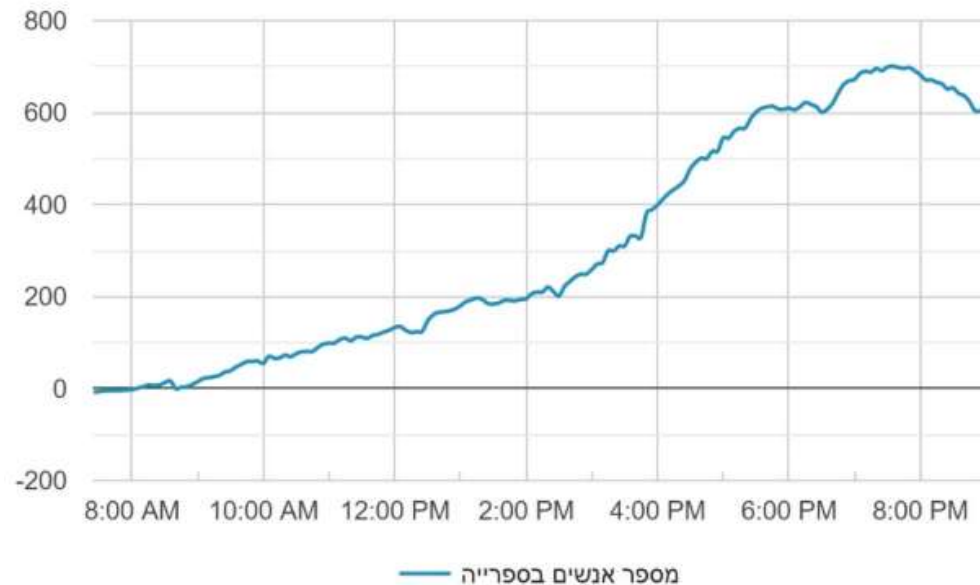
מספר אנשים בספרייה



Project Scope

Motivation

- The blue line represents the number of people within one of the Technion's libraries, from its opening until its closing (when there are no more people left within it). As can be seen, while the library should be empty and contain 0 people, the system believes there are around 600 – which goes to show the inaccurate counting of the video processing throughout the day compounding to this level.



Project Scope

Hypothesis

- Using a combination of a Deep Neural Net to identify people on the frame (every few frames) and a good tracking algorithm to keep track of the detected people on the frame should provide with good accuracy.

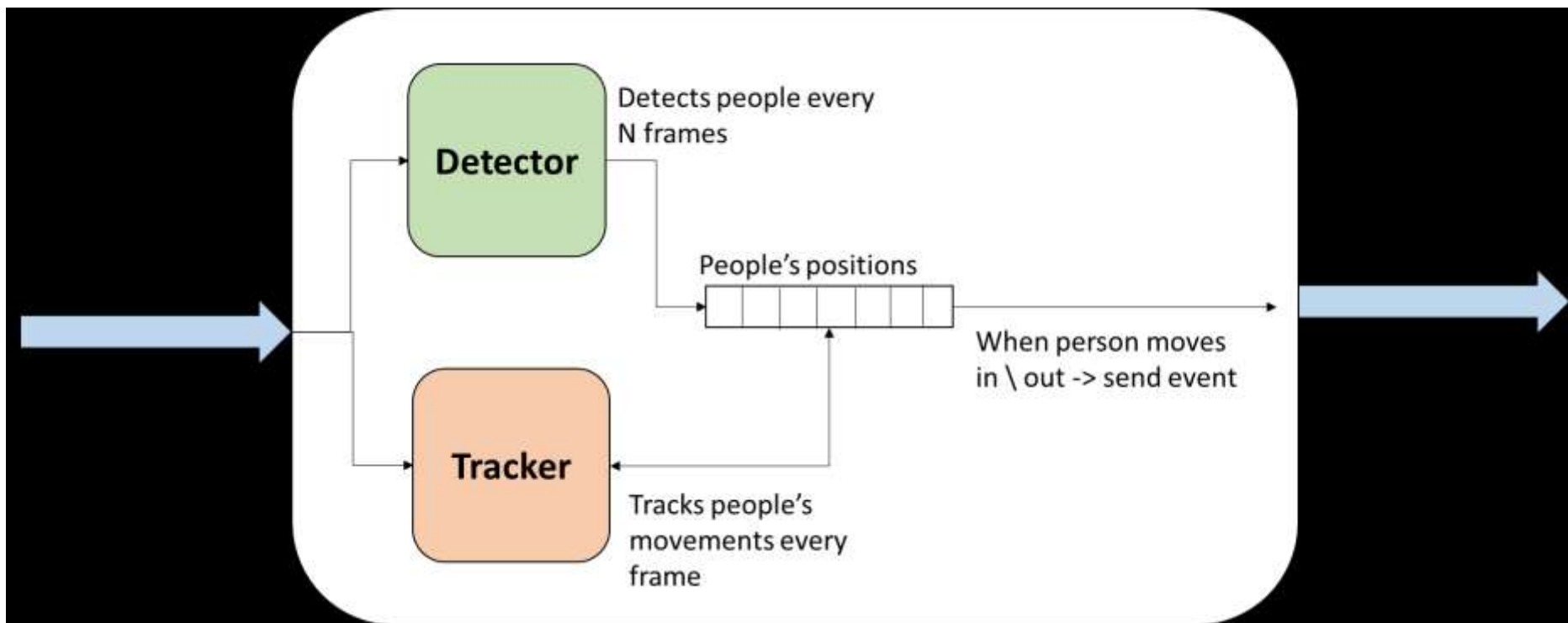


Project Objectives

- The creation of a people's counting system, able to count people entering and leaving an entrance with precision.
- The system must be easy to set up for new entrances to serve future expansion plans by the Technion's libraries.
- The system must be able to work in real-time, given the basic hardware (strong PC with GPU) to function.
- The system should be able to work over the internet, receiving its feed from live stream IP cameras.



Block Diagram



Solution Procedure

The basic structure of the solution

- Input & Output: The system receives a video stream from the cameras, and outputs events whenever a person was detected entering or exiting the entrance (with the corresponding “in” or “out” event).
- Internally, the system has two main components:



Solution Procedure

The basic structure of the solution

- Detector: a component detecting objects it classifies as human (implemented with YOLOv3), and either identifies them as objected already detected in the past (in which case their positions will be updated) or as new objects (and then adds new objects to the people's positions array).
- The detector attempts to make detections only every 'N' frames in order to lower the computational cost of the system.



Solution Procedure

The basic structure of the solution

- Tracker: this component tracks the locations of already existing people (previously detected by the detector) and updates their positions in the array. Implemented by Open-CV's CSRT tracker algorithm.
- In our implementation, each person has his own tracker object.
- Every time a person has changed his position from the entrance exterior region to the interior region (which is marked by a line or polygon), a counting event occurs and an “in” or “out” event is sent from the system.



Solution Procedure

System improvements: developing from experiments

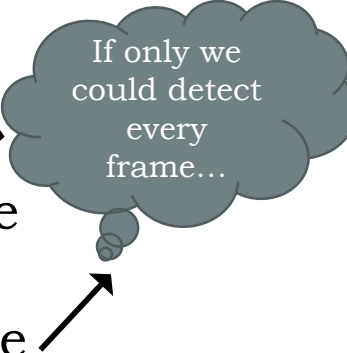
- We have discovered the basic structure of a detector component and a tracker component do a reasonable job, but not achieving high enough accuracy as required.
- Lots of small improvements were added to the system, trying to solve accuracy issues.



Solution Procedure

System improvements: developing from experiments

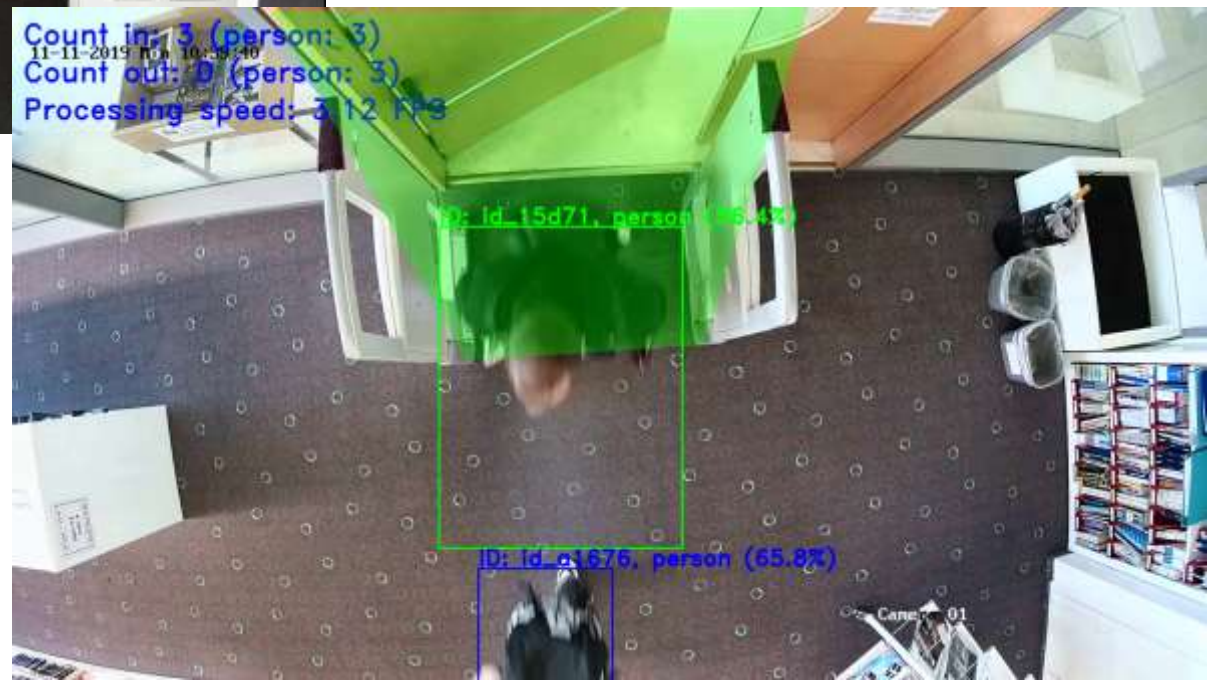
- The most significant issues: Detector & Tracker failures
 - Detector:
 - 1) Missing the best frames to detect a person's movements due to interval
 - 2) Confidence threshold being too high for edge cases
 - Tracker: losing tracked people without having the detector to fallback upon due to interval
- **Solution:** “*Detection slowdown*” mode
- **Results:** raising test results to 100% accuracy



If only we could detect every frame...

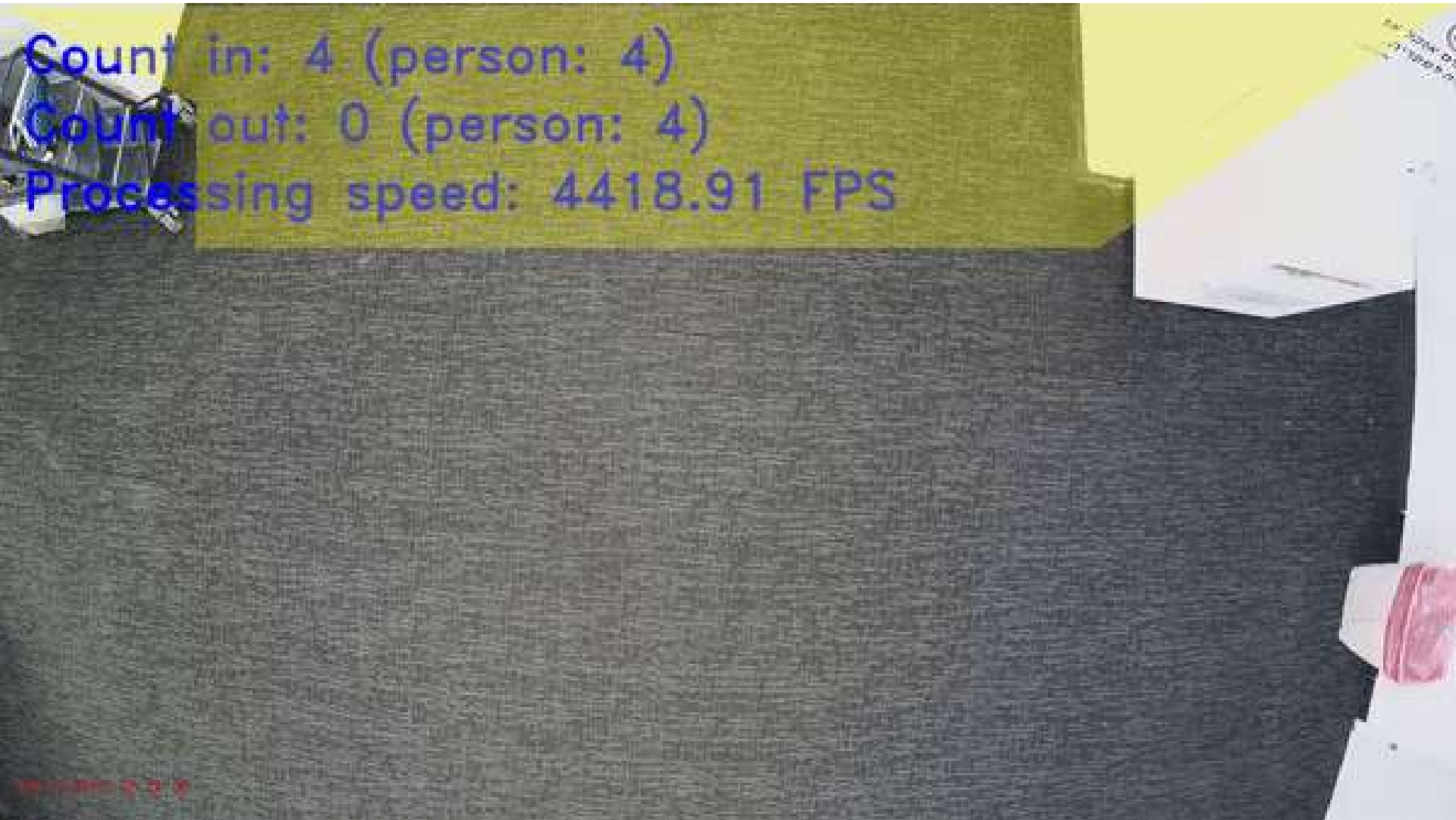


Results



Count in: 1 (person: 4)
Count out: 3 (person: 4)
Processing speed: 108695.65 FPS





Results

Library	Ins	Outs	Total Accuracy
Total	38/40	28/28	97%
Central	8/8	4/4	100%
Architecture	6/6	3/3	100%
Mechanical	6/6	3/3	100%
Electrical	6/6	7/7	100%
Medical	12 / 14	11/11	92%

In total: 97% accuracy

*And we suggest simple ways we believe will boost accuracy **even more!**



Results

Speed

- We used the Central library test video to measure the system's speed, which is a 320 seconds video at 12 FPS as received.
- With doing nothing more than streaming the modified video (adding the counting line and bounding boxes on it) **and recording** (to the file-system, a costly operation), it took **229.5**~ seconds.
- **Without recording** the modified video, it took **192** seconds.
- Without even showing the modified video, only counting the people in the video, it took **182.5** seconds.



Conclusion

- The system achieves high accuracy without being heavy on computational resources (relatively to the task). If we were to summarize all people entering and exiting in all of the test videos, we would stand on 97% accuracy in counting.
- We avoided overfitting to any specific library by limiting the amount of factors we could change (the most significant one being Maximum Continuous Detector Failure).
- Our benchmark test videos were 12 FPS, but higher FPS should provide **better accuracy** (though it might be slow the system and will change the optimality of the parameters we've found).



Works Cited

- Mask R-CNN paper: <https://arxiv.org/abs/1703.06870>
- YOLOv3 paper: <https://arxiv.org/abs/1804.02767>



Future Work

- Increasing FPS of videos from cameras and adjusting the system's parameters accordingly to increase accuracy even further.
- Changing cameras positions to center bottlenecks. Optionally, adding a physical bottleneck if the entrance lacks any.
- Use even a better detector or tracker in the future when such will emerge. The system was built in a modular way to allow the easy switching of models.

