분산환경에서 비용 효율적인 다중 카메라 사용자 트래킹 시스템

Kevin Oh, 김재훈
아주대학교

kevinoh@ajou.ac.kr, jaikim@ajou.ac.kr

MCPT: Multi-Camera People Tracking on a Budget with Distributed Systems

Kevin Oh, Jai-Hoon Kim
Ajou University

Summary

This paper presents a study on achieving multi-camera people tracking on a limited budget through the use of distributed systems. The main idea is to divide the total number of cameras among different machines, aiming to maximize the number of cameras each machine can handle. An object detection model is utilized over each camera stream, to detect individual people the results of which are sent to a central server to be used to calculate the movement of people across multiple camera views.

I. Introduction

Object detection is a well-explored fundamental task in computer vision. But computer vision generally trends towards larger, deeper networks. Better performance often hinges on training larger networks or assembling multiple models together. Running and training these complex models comes at a high computational cost, requiring better, more expensive hardware to match. Modern real-time object detectors like YOLO [1] have come a long way delivering a balance between speed and performance.

In this work we focus on how to deliver a scalable solution for handling a multi-camera people tracking task. Say we have a public place like an airport or supermarket and we are interested in tracking the movements of people in those areas to provide better service. Such places are also likely to already have ample security camera coverage of the location.

II. Related Works

A Distributed Approach for Real-Time Multi-Camera Multiple Object Tracking [2] is a similar work that also proposes a distributed approach for multi-camera tracking, but they are handling tracking across multiple viewpoints of the same area. This work deals with tracking across a large space covered by multiple cameras.

Deep Multi-Camera Tracking [3] proposes an end-to-end tracking pipeline to provide a system for multi-camera tracking in real time.

III. Proposed Algorithm

To tackle this task, we introduce a layered distributed system. See Fig. 1 for an overview of the system architecture of MCPT. The top layer consists of the IP cameras used to stream video of the space we will be tracking the movement of people in. The second layer consists of the worker nodes which act as compute nodes. Each worker connects to the streams of a defined number of cameras it is responsible for, running the object detection model over each stream to detect people. The results of which are periodically sent to the central node. This node uses the data along with knowledge of shared camera coverage through the use of a camera calibration program to resolve tracking of people across multiple cameras. When the cameras are first installed to cover a space, snapshots will be taken by the cameras with distinctly marked objects at the edges of camera views. Shared objects in the snapshots will be used with the calibration program to calculate which cameras have shared coverage. This node is also responsible for storing all the data to a database.

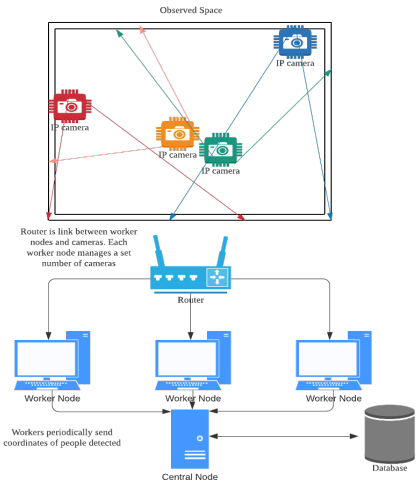

Fig 1: System architecture of MCPT

To handle tracking we use an implementation of the YOLO [1] object detection model to detect people. To

expand detection to tracking we id tag detected persons and use an algorithm to compare the motion of people between frames of video and match the ids.
Tracking the movement of people across views covered by different cameras will be done by the central node. The system assumes cameras will be setup at some height looking downwards at an angle towards the space and that the position of the cameras is fixed. Upon setup an image of the area covered by the camera with reference items at the edges of the view will be taken to be used by a camera calibration program to compute the overlap in camera coverage by the multiple cameras. This will be used so that when a person leaves the view of one camera the system can figure out which view she came from.

IV. Performance Evaluation

The goal of the system is to maximize the size of the space we can track people across. For simplicity's sake we will assume a best-case scenario where the space to be observed is a wide open area with minimal crowd density and no obstacles to obstruct camera views and the people are moving around this space at an average walking pace, about 5 km/h. The system must be real-time in that video from the cameras will be streamed and not recorded so detection must be done directly on the stream. Concerns of privacy have lead many countries to pass privacy laws limiting the use of tracking software, a common requirement is that no video is saved and subjects are not personally identified.

More cameras mean more space covered but each camera comes at a cost. To know how much open space each camera can theoretically cover we need to know how small in pixels a person can take up in an image and still be detected. Another important note is the impact of image resolution on the computational cost for running detection, similarly affecting the number of cameras each worker node can handle. We will assume an image size of 480x360 to reduce load while remaining detailed enough to adequately detect people in our proposed scenario. Image will be captured at camera's natural resolution which would vary depending on camera model, and the resulting capture can be scaled to a set resolution. Also note that some overlap is required between coverage of each camera for the camera calibration program to work.

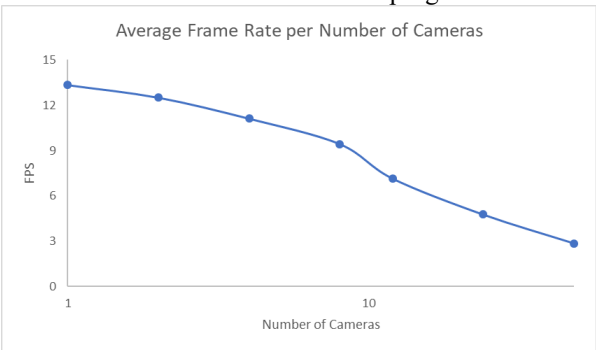

Fig 2: Estimed average FPS per number of cameras for a single worker node

Worker Nodes: This is the main bottleneck of the system; each worker node will need to be equipped with a GPU to run the detection model on video from cameras. To limit costs, we need to maximize the number of cameras each node can handle as cheaply as possible by minimizing the computational cost per image. We will define a real time frame rate goal at 10fps. A relatively low frame rate to minimize computation cost while remaining fast enough to capture movement at the assumed walking pace.
The burden on computation time of higher frame rate and resolution becomes heavier as the number of cameras grows. An important bottleneck for model runtime is the copying of data to the GPU by the CPU. There might not be a large difference in runtime between a single 1080p image versus 360p but when the number of images increases, the CPU will quickly become the bottleneck. Thus it is proposed to use several worker nodes each with a cheaper GPU, than to use a single worker with a powerful GPU.

Under these conditions testing with a GeForce GTX 960 GPU and i5-6600 CPU a single worker node can support a maximum of 6 cameras. Fig. 2 shows a comparison of the effect of multiple cameras on estimated average fps.
Central Node: The central node will periodically receive detection data from each worker node. For each frame of video, the coordinates of each detected person, as the center of the detected bounding box will be packaged to be sent periodically to the central node. As the central node only needs to know the coordinates of detected people, rather than the image data itself, the size of data sent will be small but the concern is how often this data will be sent. If each worker node

V. Conclusion

Real time multi-camera tracking is a tricky, unsolved problem. Doing it on a budget per real world business scenarios is an even trickier task. In this paper we explore ways to balance out cost for performance to tackle this task.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2018R1D1A1B07040573)

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934, 2020

[2] Previtali, Fabio & Bloisi, Domenico & Iocchi, Luca. A distributed approach for real-time multi-camera multiple object tracking. Machine Vision and Applications. 10.1007/s00138-017-0827-5, 2017.

[3] Quanzeng You, Hao Jiang. Real-time 3D Deep Multi-Camera Tracking. arXiv:2003.11753, 2020.