Multi-Camera Re-identification and Tracking

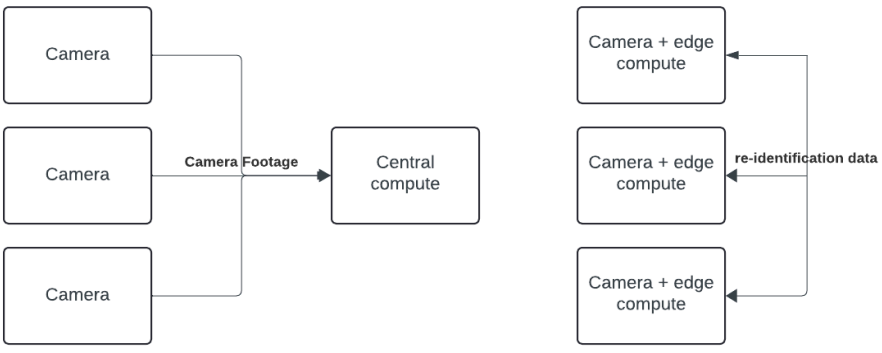
Eric Ye <ericye@stanford.edu>
Ryan Rumana <ryan.rumana@gmail.com>
Code link: https://code.stanford.edu/ericye/ee292d-project (requires Stanford sign-in)

Background and Motivation

Busy areas such as plazas, malls, streets, may be surveilled by security cameras. It can be useful to track the movement of a single person or people across an area with many people in it, such as for public safety or marketing purposes. However, the cameras covering an area are usually not networked together, and tracking the movement of a single person across these cameras is a challenging manual task, requires camera calibration, or is not possible at all.

This problem of tracking people or objects individually across a single video is known as multiple-object tracking (MOT) and is a well-known problem with many solutions in computer vision. A separate but related problem is how to identify the same person from multiple angles. This problem is called reidentification (ReID) and it, too, is a well-known problem in computer-vision. We aim to make it possible to track individual people and objects across the field of view of multiple cameras. Typically, this is done by streaming each camera's feed into a powerful central server and having that one server do both MOT and ReID. While this method is simpler to implement, it is bandwidth intensive as streaming video over the network is costly and cost increases for every camera added to the system.

Our project will attempt to solve this problem by combining tracking and ReID algorithms to a distributed set of cameras that overlook the same or similar areas from different angles, also including non-overlapping areas. This method allows us to avoid the cost of streaming the raw video to a centralized server to process. Our goal is to have a relatively low amount of bandwidth required for this algorithm to pass data rather than the video stream between cameras. See diagram below for a comparison of the two approaches.



Methodology

The methodology we adopted is to pair Yolov8 nano (Ultralytics, n.d.) to create bounding boxes around people in each camera frame and then to pass those bounding boxes into ResNet-50 (He et al. 2015) for person ReID. Once this is completed, the camera frame, number of embeddings generated for that frame, and the embeddings themselves are saved to a ramdisk on the device where it is served via HTTP and made accessible by other devices on the network. We use cosine similarity to determine if embeddings match each other, allowing a central server to correlate one camera's output to another without needing to see or process the video frame, reducing network overhead substantially, especially as the number of cameras in a deployment increases. This identification is extremely lightweight, and could be run on one of the Raspberry Pis in the camera network if desired, however the debugging UI we designed would not be accessible for the typical headless operation of edge devices.

Datasets

We ended up using the pre-trained version of Yolov8 nano which is trained on the Common Objects in Context (COCO) dataset which is a large dataset that contains 80 object categories and 5 captions describing the scene. This was sufficient for our purposes since the model is already adept at identifying and creating bounding boxes for people in still images. For ResNet-50, our ReID model, train and evaluate on the Market1501 dataset (Zheng et al. 2015) which is a pre-assembled collection of 1501 people captured from multiple different cameras and camera angles. This dataset is a very standard choice for ReID and is optimized for full body images like those captured by security cameras.

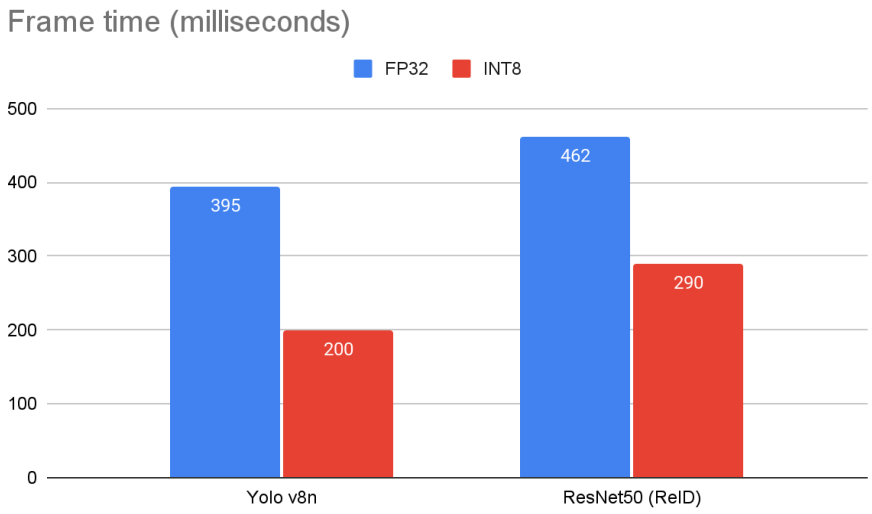
Model Optimization

To run both the object detection model and the ReID model efficiently on the Raspberry Pi, we quantize both of them from their fp32 original weights to int8-based quantized models. For the object detection model, this was straightforward since the Ultralytics YOLO software we used supports quantization out-of-the-box and includes helpful examples on usage, even including some of the preprocessing and postprocessing code implemented for quantized models.

For the ReID model, this was more challenging. We chose the ResNet-50 architecture as a simple starting point that had decent performance in accuracy (mean average precision (mAP) of 79%) and runtime. The ReID model was implemented in PyTorch (Pytorch authors, n.d.), which currently (as of this writing) offers three unstable ways to perform quantization: eager mode quantization, FX graph mode quantization, and Pytorch 2 export quantization. Of these, only eager mode quantization worked, and this only worked after significant troubleshooting and manually fusing layers to make them compatible with quantization. We calibrated the quantization with 100 training images in the Market-1501 dataset. Surprisingly, the mean average precision was unchanged after quantization (79%).

Before we were able to quantize our ResNet50-based ReID model, we attempted to use the YOLO v8 XL classification model as our ReID model. The motivation was that we knew the Ultralytics YOLO software had a well-defined path for quantization, if we were able to train YOLO as a ReID model, quantization might have been easier to do. However, we were only able to achieve approximately 40% mAP, even with the largest YOLO model available, on the evaluation set, despite training to nearly 100% accuracy in the training set. We suspect that the performance issue was due to lack of a bottleneck layer to reduce the feature vector width used for embeddings, which is present in the ResNet50-based model and helps avoid overfitting.

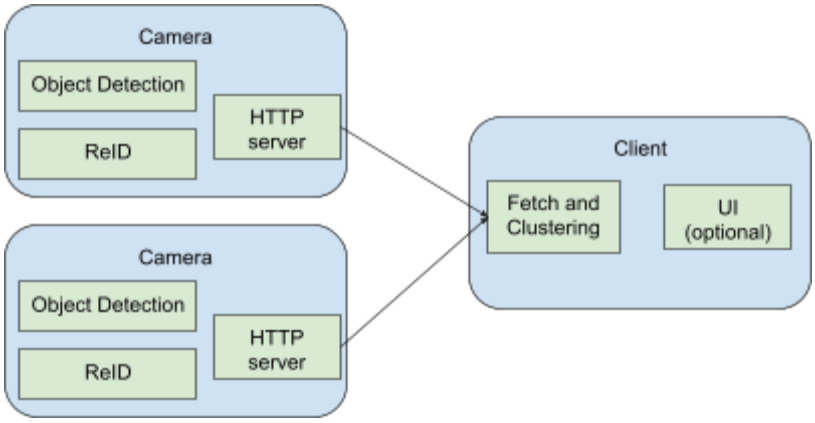
Quantization improved runtime of the object detection algorithm by approximately 49% and the ReID algorithm by approximately 37%. Since the overall runtime depends on how many people are detected in-frame, there is no general value for the overall frame time improvement.



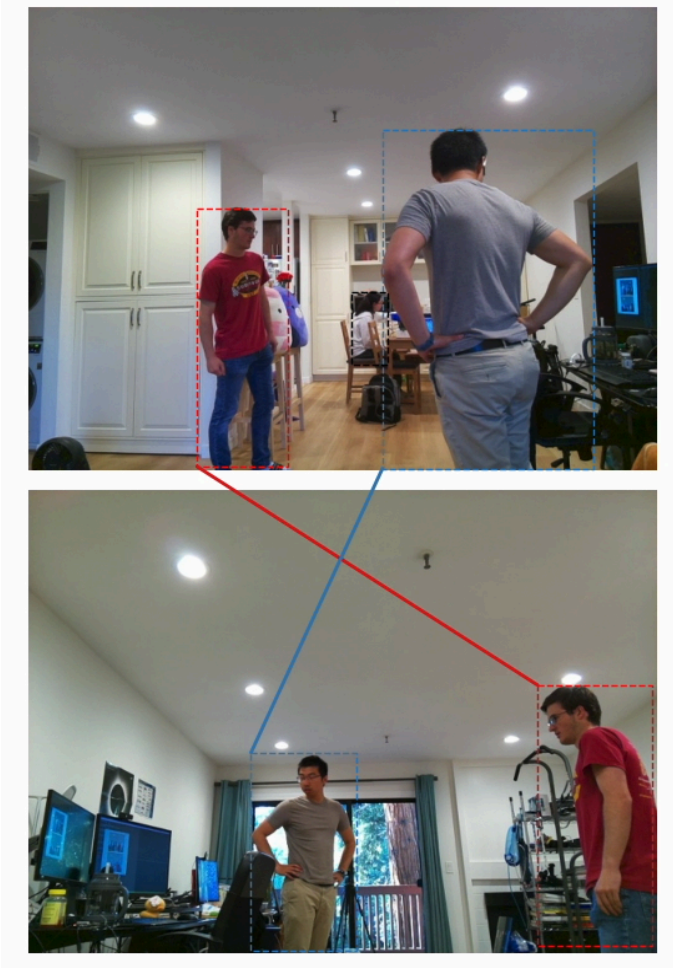
Model runtime performance

Demonstration Software Architecture

We capture complete frames from the Raspberry Pi camera, resize them to 640 by 480 pixels, then run them through the quantized object detection model to detect people. For all the people detections, we run the cropped image through the quantized ReID model. We save the full frame, each cropped image, the bounding boxes, the ReID embeddings, and some debug information into a directory, which is then served via an HTTP server. The central computer / monitor can then fetch all of this data from each camera on the network to cluster people with similar embeddings together without having to stream the entire captured frames from each camera or run the ReID model locally. This scales better with the number of Raspberry Pi cameras, since the heavy object detection and ReID algorithm runs on the device rather than centrally, and the central computer only has to perform clustering, which can be as simple as calculating the cosine similarity between embeddings.



Overall system architecture



Software interface, showing 2 people being correctly detected and associated with another. Note that the cameras are from opposing fields of view (180 degrees apart), and that a third person, boxed in black in the upper image, has no associated position in the bottom image since she is behind the bottom camera. Video link: https://tinyurl.com/5eaujycu

Discussion and Results

As mentioned above, the quantized ResNet50-based ReID model ran with the same accuracy (mAP of 79% on the Market-1501 dataset) as the full-precision model with a 37% lower runtime on the Raspberry Pi. Qualitatively, this results in a model that is able to perform the correct detection/association most of the time with overlapping cameras, although we notice that if one person is cropped or occluded from one camera, detection often fails. The increase in framerate we observed enabled platform viability of the Raspberry Pi for this application. See the above video link for a demonstration of this software working. We believe that having a stable embedding and Kalman filtering/tracking on each camera individually may improve this as we will discuss below.

Future Work

Stable Association over Time

In the live and video demonstration, we do not store ReID embeddings across video frames. Instead, each frame from each camera is treated as totally new and embeddings are re-associated every time. Creating a stable embedding that allows associating the same person across frames and when people leave the field of view of the image improves the usability of the software by allowing for non overlapping cameras and re-association when the same person returns. It may also improve robustness by allowing the algorithm to have multiple embeddings of the same person, from different points of view.

Per-camera Kalman Filtering

In addition to tracking the same person within one camera by their ReID embeddings, a Kalman filter can also be used to track the same person as they move across a single video stream. This can provide even more information to improve associating across cameras and helping disambiguate similar-looking people.

Improvements to the Association Algorithm

The algorithm used to associate different ReID embeddings can be improved to take into account the position and motion of the people in-frame (see Kalman filtering, above) and other properties such as uniqueness (one person shouldn't be associated with two different people in another camera frame).

Camera Differences and Profiles

Different cameras may have different color and exposure settings, which produce different embeddings even for the same person. Training an algorithm to either be robust to this or to learn cameras' profiles may improve performance.

Runtime Performance Improvements

Instead of quantizing the models and running on CPU, the model may be rewritten to run on the GPU. There are reports of being able to run Yolov8, our object detection model, on the Raspberry Pi's GPU via ncnn (Allan 2023), which would likely perform better than our current approach of quantizing and running on CPU. Similarly, our ReID model may also be replaced with a network such as MobileNetv2 which may be faster than the current ResNet50-based model.

Simultaneously Localizing People and Cameras

Although we set an explicit requirement that the cameras need not be calibrated, it should be possible to solve for the relative positions of camera to each other based on the positions and velocities of commonly-detected people. This would allow for a "top-down" transformed view of the people, which may present an interesting/useful application.

Societal Impact

The Panopticon is a well-discussed hypothetical prison invented by Jeremy Bentham in the 18th century. It is designed so that a single prison guard can see all the prisoners but none of the prisoners can see the guard in the middle. The prison encourages the prisoners to police themselves, since they never know when they are being watched, so they must assume they are always watched.

In the modern day, it is difficult to go about our daily lives without some sort of surveillance. However, we can generally find comfort in the idea that the many CCTVs we see are generally not linked and even if they are monitored by people, a single security guard will only be responsible for a few dozen CCTVs at most. Additionally, video storage is expensive, and businesses are unlikely to continuously record video for long periods of time. You are unlikely to be tracked for a very long amount of time and leaving the field of view of the CCTVs will generally end your state of surveillance.

With this ReID technology, it may be possible to track people persistently, cheaply, and automatically, with very few humans in the loop, across large areas and timespans. Storing embeddings over time is orders of magnitude cheaper than storing raw footage, so they can be stored for much longer. If someone is to be tracked, they could be easily searched for in a long-running database of embeddings, a tireless digital panopticon. Development of this technology should consider not just the explicit privacy impact of being able to track people across but the self-policing effect of making the technology widely available and known.

Citations

Allan, Alasdair. 2023. “Benchmarking Raspberry Pi 5.” Raspberry Pi. https://www.raspberrypi.com/news/benchmarking-raspberry-pi-5/.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Deep Residual Learning for Image Recognition.” arXiv. https://arxiv.org/abs/1512.03385.

Pytorch authors. n.d. “pytorch/pytorch: Tensors and Dynamic neural networks in Python with strong GPU acceleration.” GitHub. Accessed June 10, 2024. https://github.com/pytorch/pytorch/.

Ultralytics. n.d. “ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite.” GitHub. Accessed June 10, 2024. https://github.com/ultralytics/ultralytics.

Zheng, Liang, Liyue Shen, Lu Tian, Shengjing Wang, Jingdong Wang, and Qi Tian. 2015. “Scalable Person Re-identification: A Benchmark.” Market-1501 Dataset. https://zheng-lab.cecs.anu.edu.au/Project/project_reid.html.