Bandwidth-Efficient Cross-Camera Video Analytics

Hyunwoo Jung
20183573
School of Computing
KAIST

Dongmin Kim
20184220
School of Computing
KAIST

Nick Heppert
20186505
School of Computing
KAIST

Taeksoo Kim
20193162
School of Computing
KAIST

I. Revised problem statement

Camera network system has become common these days. The advance in image recognition technology like deep learning and the decrease in the cost of IoT devices have lead the pervasiveness of the camera network system. One example of such camera network system is Amazon Go where there is no cashier but customers can do shopping without checkout. Customers just grab what they want to buy and leave the store then payment is automatically done because the video system tracks down who takes out what. Another example is video surveillance system in the metropolitan city like London and Beijing where a million of cameras have been deployed. However, it is very challenging to apply the technology used in Amazon Go to such large scale video surveillance system because it is expensive to have cameras equipped with compute devices or to wire all cameras to cloud systems that have powerful compute devices such as GPUs.

Recently, Jian et al. [4] have propose an algorithm for cross-camera video analytics called RexCam. The authors evaluate RexCam on the DukeMTMC dataset [6], which contains footage from 8 cameras located on the Duke University campus. The authors show the high degree of spatial and temporal locality of the footage. That is, the traffic between camera A and camera B is more likely than the one between camera A and camera C if camera A is located close to camera B but far away from camera C. Thus, they propose algorithms utilizing spatial and temporal locality to reduce compute cost and delay.

However, RexCam assumes a somewhat unrealistic environment. Some of the assumptions are that cameras are high-definition as well as that they are connected to each other and to a central powerful computation unit (most likely a cloud) over a high-speed local area network. RexCam also assumes that bandwidth between cameras and to the cloud is sufficient so that it can transmit video data in real-time with 30 frames per second. According to Zhang [8], however, existing network infrastructure is insufficient to accommodate many simultaneous high-definition video feeds such as Dropcam [2]. That is, the traffic between cameras and the centralized cloud system is bounded by the bandwidth of Wi-Fi, which results in the lack of scalability. In section II we will do an depth-in analysis of bottlenecks in the network. For this end, we propose resource-efficient distributed edge computing system for large scale camera network in Section III.

II. Design Considerations

A. Bandwidth of wireless network

Wired cameras are inefficient in terms of cost and effort due to their characteristics. Thus, many studies on video surveillance system have been done by assuming wireless network as in [8]. In this paper, we also assume that all cameras in our system communicate with a Wi-Fi network. If cameras send their video streams to the cloud through a Wi-Fi network, RexCam's approach is not realistic because of the bandwidth issue of Wi-Fi.

For example, DukeMTMC dataset, which is used by RexCam for experiments, consists of 1080p video recorded at 60 frames per second from 8 cameras. This means that each camera sends approximately 28.8 Mbps of video data, which is fairly large, to the cloud. Next, we will see if Wi-Fi Access Points (AP) receiving this amount of data can handle it. There are various versions of the Wi-Fi protocol and their bandwidths are different. However, recent studies including [3] have shown that most Wi-Fi models support up to 802.11g and 802.11n among many Wi-Fi standards. 802.11g can support only 54 Mbps and 802.11n can theoretically support 300 Mbps but it goes down to 100 Mbps in real world [7]. In this case, Wi-Fi APs cannot take video streams sent by only four cameras. Therefore, it is not feasible for all cameras to send their video streams.

B. Filtering cameras to stream

Our solution is to filter the camera to send video data to the cloud using the filtering techniques presented in Rexcam. RexCam builds spatio-temporal correlation matrix between cameras in the preparation phase to reduce computational overhead of identity tracking algorithms by picking up only relevant images to current object of interest among video streams. This is a reasonable choice because, from a locality point of view, there is a high probability that an object will be found again in an image close to the image where the object is already observed. We propose to dynamically filter which "cameras" would send their frames to the cloud, rather than filter "images" already sent to the cloud. That is, we suggest a system which make only the cameras adjacent to the camera that

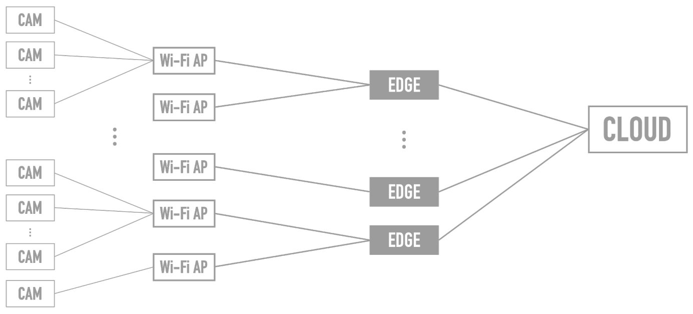

Fig. 1. Architectural overview of our new system.

is currently sending the video containing objects stream all frames of video, while other cameras stream partial frames of video. By doing so, we can naturally solve the bandwidth problem as well as reduce the computational overhead which is already achieved in RexCam.

III. Proposed scheme

In Section II we showed that the approach in [4], streaming all cameras to the cloud, is unrealistic in the real world. In Figure 1 we propose a new architecture which considers the previously outlined real-world requirements. To summarize our idea in one sentence from the clouds perspective our proposed system improves the following: instead of filtering out unwanted camera streams in the cloud we will utilize the edge in order to control which cameras stream is to the cloud.

A. Proposed Architecture

Similar to [4] we place multiple cameras in the environment but rather than letting stream every camera to the cloud and then do the filtering there we propose a another layer as shown in [5]. First, as analyzed in Section II we would assume that the amount of Wi-Fi cameras increases we therefore need to connect each camera to a Wi-Fi access point rather than using wired connections. Second, each access point is then further connected by Gigabit LAN to another node – an edge device or a so called “controller”. Eventually, each edge is connected to the cloud server where as in [4] the tracking application is run.

Please note that the Wi-Fi Access points and the edge device doesn’t necessarily have to be separated. Instead of our shown Gigabit LAN connection the Wi-Fi access points could also be integrated in the edge, be connected through a Peripheral Component Interconnect (PCI) connection or even USB. As the connection won’t be slower than the incoming connections at the Wi-Fi access point, we could neglect it and don’t raise any more attention to it.

B. Algorithmic Changes

With this updated architecture we need to modify the flow of streams in our system. We propose to shift the “filtering” step based on the spatial-temporal model in [4] on the edge. This will reduce the traffic which is sent to the cloud heavily as no longer every camera stream is uploaded to the cloud. There are no restrictions which could hinder us in doing so.

Please keep in mind that we only focus on the inference part as we assume that the spatial-temporal model is constructed in an offline pre-phase. For all analysis regarding amount of data needed for offline profiling etc. we point towards [4] where they extensively studied the offline model creation phase. As we don’t do any improvements here we assume the model as “given” in the following. Such a spatial-temporal model gives us information whether a camera is correlated spatially or temporally based on a defined threshold for both cases respectively.

We will still use the same model as described in [4] but we will move the camera filtering part on the edge similar to the proposed controller in [5]. The edge now takes over the role of filtering out the camera streams which are unwanted. As a consequence the cloud has to share its current across all edges. Sharing the state means sharing the currently used main camera c_q as well as the current frame index f_{curr} (following notations in [4]). Additionally, considering (see II) that one cannot stream all cameras to the edge, we need to modify the filter-operation. Therefore, we do not stream all edges constantly to the edge because our Wi-Fi Access Point has a limited bandwidth, but rather use the spatial-temporal model to find relevant cameras and then request explicitly the relevant camera streams. This reduces the transmitted camera streams to lowest minimum possible.

C. Reducing the Spatial-Temporal Model depending on Edge

Despite reducing the consumed bandwidth from the camera towards the cloud in the long run moving, the filtering on the edge has another benefit. In the current approach, there are two $N \times M$ matrices where N is the number of source cameras and M is the number of destination cameras. One matrix is for a spatial correlation between a source camera and a destination camera and the other is for a temporal correlation. When the filtering is done on the cloud for one specific camera, this camera needs to be checked against all other cameras in order to determine the correlation between them. This means one row or column for each matrix has to be iterated. Such big matrices are undesirable memory- as well as computation-wise. But luckily, the model is pre-computed and only a certain set of cameras are connected to one edge. Therefore, only a reduced model needs to be upload to the edge.

We argue that one can reduce the matrix by two independent operations. First, one dimension of the matrix can be reduced by removing all cameras that are not connected to the edge (this reduces the size of the destination cameras). Second, from the remaining cameras (source cameras) only a small subset is relevant which could be identified by setting the threshold of spatial and temporal correlation rather conservative. We therefore can further prune the other dimensions of source cameras and will be left with a small sub-matrix of the rather big model. This allows us to reduce hardware cost needed for the edge as we use less computing. In the scale of the Duke dataset [6] which was used in [4] this might not matter as there were only a total amount of 8 cameras. But with increasing amount of cameras up to a easily ten thousand (e.g. London underground [1]) this matrix becomes quite big to search for in real time at every frame.

D. Further Extensions

With this new architecture there are multiple extensions we could apply in order to make our system more autonomous or robust. Introducing them in an extended manner would be out-of-scope for the course, therefore we only give a brief overview of them.

1) Initial Query Discovery: Currently the system proposed in [4] does not provide any solution for doing an initial query. They say queries have to be issued by hand which is not scalable to a lot of cameras. To solve this problem there could be multiple solutions.

First, one could use a system as described in [5] where the authors use the cameras and edges resources efficiently in order to fulfill simple tasks (such as detection or tracking). Of course, in order to save hardware costs for more powerful cameras/edges, the advanced reidentification algorithm would be run on the cloud as described previously.

Second, if the previous solution is not accurate enough, one could leverage the cloud in order to get more stable results. Instead of selecting specific cameras to be streamed to the cloud, the bandwidth between cameras and the Wi-Fi access point is shared equally. Of course, this will result in a lower frame rate, but as we are currently not tracking anything in real time we could sacrifice this framerate drop. On the other hand when constantly streaming to the cloud at maximum capacity the costs for running the cloud server will increase.

To make a clear decision one has to study these two solutions in addition different methods. Doing so would be out of scope for this project we therefore kindly assume this as future work or an open issue in our new, proposed system and in the original RexCam [4] implementation.

2) Fast Replay Search: One problem we ignore in our project is how to conduct a fast-replay search as described in [4]. It cannot be carried out as proposed in the original paper though, because at the point where a fast replay is needed not all data is available at the cloud. Therefore, one has to provide the historical data one after another to the cloud for processing while caching all incoming data simultaneously. If we use the reasonable assumptions that the cloud can process streams faster than real-time (if in question just deploy more GPUs) as well as our upload bandwidth allows us streaming faster than real-time we can show that fast replay search is still possible even though not all data resides on the cloud but rather on the cameras itself.

References

[1] CCTV - British Transport Police. https://www.btp.police.uk/advice_an_information/safety_on_and_near_the_railway/cctv.aspx. Accessed: 2019-04-06.

[2] Dropcam website. https://www.dropcam.com/. Accessed: 2019-04-07.

[3] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-e, A. Bhartia, and D. Aguayo. Large-scale measurements of wireless network behavior. In ACM SIGCOMM Computer Communication Review, volume 45, pages 153–165. ACM, 2015.

[4] S. Jain, J. Jiang, Y. Shu, G. Ananthanarayanan, and J. Gonzalez. Rexcam: Resource-efficient, cross-camera video analytics at enterprise scale. arXiv preprint arXiv:1811.01268, 2018.

[5] S. Y. Jang, Y. Lee, B. Shin, and D. Lee. Application-aware iot camera virtualization for video analytics edge computing. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), pages 132–144. IEEE, 2018.

[6] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In European Conference on Computer Vision, pages 17–35. Springer, 2016.

[7] Z. Shah, S. Rau, and A. Baig. Throughput comparison of ieee 802.11ac and ieee 802.11n in an indoor environment with interference. In Proceedings of the 2015 International Telecommunication Networks and Applications Conference (ITNAC), ITNAC '15, pages 196–201, Washington, DC, USA, 2015. IEEE Computer Society.

[8] T. Zhang, A. Chowdhery, P. V. Bahl, K. Jamieson, and S. Banerjee. The design and implementation of a wireless video surveillance system. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15, pages 426–438, New York, NY, USA, 2015. ACM.