Location-Aware Distributed System for Cameras in a Resource-Efficient Way

Hyunwoo Jung
20183573
School of Computing
KAIST

Dongmin Kim
20184220
School of Computing
KAIST

Nick Hepper
20186505
School of Computing
KAIST

Taeksoo Kim
20193162
School of Computing
KAIST

I. Introduction

Recently, the increase in use of smart devices, also known as IoT devices, has led to the connection between those devices. Due to the limitation of resources for IoT devices, it is usual to offload compute-intensive workload to cloud computing systems to utilize their plentiful resources. However, network latency holds stops developers from handling everything including time-sensitive task in the cloud. To this end, edge computing has recently arisen to solve this latency problem. Edge computing is the emerging concept to deal with the trade-off between computation time and network latency. Unlike cloud systems, edges are decentralized and network-oriented with relatively smaller resources than a cloud but responses faster than the cloud. Thus, IoT devices can leverage such edge systems as an alternative to cloud systems according to the characteristics of tasks by deciding where to put the computation – on the IoT device, the edge or the cloud.

Video analysis is an exemplary task for edge computing because it requires relatively expensive resources such as GPUs and real-time responses (low latency). One concrete example for video analysis is processing a surveillance cameras video stream – also known as closed circuit television (CCTV). Traditionally, they continuously transmit all the scenes recorded to the control center in which security officers monitor the video for emergency cases such as robbery and murder. The detection of abnormal situation is done by humans who always monitor the video stream. It is onerous for humans to always watch streams for rare occurrences. To automate such demanding tasks, a system should automatically determine whether the currently streamed situation is abnormal or not. Also, to catch the criminals more efficiently, it should distribute the results to all neighbor cameras. We suggest such a new system that detects abnormalities fast using edge computing and distributes it to proper cameras.

II. Related Work

A. Crime Detection

In order to automate the onerous work of monitoring video streams, a wide range of techniques has been explored. One possible solution is anomaly detection. Anomaly detection is the identification of abnormal observations which raise suspicions by differing significantly from the majority of the data. There exists a variety of methods to implement this detection [9] ranging from principal component analysis (PCA)-based approaches [11], sparse combination learning [13], autoencoders [6] to deep neural networks [17]. Each method has its pros and cons. Deep neural networks outperforms other methods in terms of accuracy but to do so they require powerful compute devices such as graphics processing units (GPUs), which is expensive and large for IoT devices to equip. In contrast, a PCA-based approach is computed seemingly immediately just by using central processing units (CPUs), which are cheaper than GPUs. When deployed in IoT devices though PCA-based approaches are not as accurate as deep neural networks. For example, Lu et al. [13] propose a sparse combination learning method and achieve 15% frame-level Equal Error Rate (EER) on the UCSD Ped 1 dataset [15] while Sabokrou et al. [16] propose a deep neural network called Deep-Cascade and achieve 9.1% frame-level EER on the same dataset. However, the sparse learning method is much faster with running at 143 fps (frames per second) on a 3.4 GHz CPU whereas the deep neural network is merely reaching 15 fps on a 2.4 GHz CPU even though it gets as fast as 130 fps on an NVIDIA GT 620 GPU. Thus, there is a trade off between using fast/less accurate methods and slow/more accurate methods.

B. Edge Computing Application for Video Analysis

Research to apply these advantages to real-time video analysis has already been done in various fields. For example, in order for a drone to operate, it is necessary to handle a lot of images which are too many for the drone itself but need to be processed in real-time at the same time [3]. Beside drones, virtual reality (VR) is also a representative example that requires a large amount of image processing and requires real-time response. To accomplish this goal, research on applying edge computing to VR is also underway [10]. In this project, we will study a distributed system that facilitates the anomaly detection by using edge computing while taking advantage of the location-aware network.

In order to justify edge computing we want to shift the focus on [4] where the authors conducted a small study in order to validate the assumption that offloading computation for video analysis on the edge is a valid approach. Similar previous work was already done in [2]. Both analysis have in common that they assume the end device is a phone. Recent low cost IoT cameras [18] with limited processing capabilities (e.g. motion/sound detection) have similar hardware specification. Therefore we can transfer the reinforcing arguments for offloading computation on the edge from [4] and [2] to our approach.

In [14] an extensive study is conducted comparing different architectures and their potential speed up. These insights as well as the analyzsis of related work in Section II-A can be used in order to choose in what form and how we could offload video analysis tasks from the IoT camera to the edge.

C. Location-Aware Networks

Despite video analysis a second component in our system will be passing the information to different cameras (nodes). The problem of passing a message to multiple other nodes in a location-aware manner already has been studied a lot in the past with multiple techniques proven to be good [12], [1], [8], [5]. In the following we will briefly cover some of the main differences in the approaches and how we could use some of the aspects from them.

In our humble opinion we think there are two ways we might structure our network. We either use a location aware publish-subscriber framework, which requires an additional middleware broker(s) [5]or each camera is connected to a set of other cameras - a peer-to-peer network [12], [1], [8]. First, we analyze the publish-subscriber setup. Second, we will take a look at peer-to-peer-network solutions.

1) Publish-Subscriber Network - Indirect Communication: As previously mentioned this would mean we need to add additional hardware in order to setup middleware brokers to reduce the additional publish-subscriber overhead on the IoT devices (cameras). In [5] a very simple is proposed by the authors. As a middleware a simple webservice is used which matches the different clients. Both subscribers and publishers define a range in which they want to receive and send events. Therefore, the middleware only delivers the message, if the ranges overlap for a subscriber-publisher-pair.

2) Peer-To-Peer Network - Direct Communication: On the other hand a peer-to-peer network removes the need of an additional middleware broker. In order to construct such a network various methods [12], [1], [8] already has been proposed we could use as a baseline. The algorithm proposed in [12] efficiently constructs a peer-to-peer-network without the need of any global information by location-aware matching technique. It does not consider only regional locations but also cuts slow connections. Such techniques are important as it might happen that IoT devices are close in a physical sense but have a high latency between them. Luckily, this shouldn't be the norm case which allows us to use the technique based on Delaunay triangulation as described in [1]. The authors of [8] try a different approach by dividing the environment in dynamically changing areas. Their method also assigns different roles to each peer, which is only realizable in our case by additional hardware.

D. Scaling video analytics systems to large camera deployments

In short, our research goal is to build a location-aware distributed system in a resource-efficient way. To do this, we need to build the system in such a way that the accuracy of the video analysis is maintained without any significant increase in cost by utilizing offloading computation to an edge as well as spatial correlations (location-aware) cameras. Jain et al. [7] present an outline for such a system and point out what today's camera system structure is missing for such an implementation.

One missing component is a cross-camera correlation database which contains the spatio-temporal correlations among cameras. It has the information about the way cameras are inter-connected in terms of their sight. Others are peer-triggered inference. When analyzing multiple videos, a video analyzer should be aware of which camera the video comes from and selectively utilize it to reduce costs. We consider that the video analyzer runs on a computation-power-limited edge device to reduce latency between the camera and the video analyzer. Therefore it is desirable to not analyze all received videos. For example, one camera's video stream analysis can be adjusted to be done frequently or seldom if there is already relevant camera with it. Let's suppose some detected anomaly keeps its position in one camera where cannot be observed in other cameras' position. Then the other cameras' video streams don't have to be analyzed continually so we can retrench costs by reducing frequencies of those video stream analysis. In an opposite case where a criminal is running away very fast, a predicted trajectory could be used to in order to inform cameras along his/her direction for a rapid detection. Despite peer-triggered inference, the authors also introduce pipeline composition which means to combine inference model across cameras or train model with other model's result to raise prediction accuracy.

As the authors do not actually implement their system but rather just designing it we will implement their peer-triggered inference. However, we will also implement an alternative solution as described in III.

III. Proposed Approach

Our approach focuses on the implementation of the system described in Section II-D. We assume we can analyze videos directly at the camera slower and with less accuracy than on the edge device. On the other hand we assume that it is not feasible to stream the

data from all cameras to the edge device. We argue for that assumption with two reasons. First, streaming all cameras at one device will reduce in a bandwidth overload. Second, the edge device only has a limited computation power and therefore can't process all video streams in a timely manner. As a solution we will build a distributed system which is able to dynamically allocate video stream pipelines of cameras to the video analyzer running on the edge in order to avoid redundant inference and get rapid response when targeting moving objects.

More specifically, we have devised two methods for coordinating cameras in a location-aware way and how they communicate with each other. First one is direct communication that you can see in the top of Figure 1. Each camera already knows its adjacent cameras so it spreads its detection of anomalies by itself. In general this is how [7] imagined their peer-triggered inference. We also suggest a second approach by adding a middleware which is in charge of spreading information through the spatial network. This would reduce the overhead (message routing, multiple peer-to-peer connections etc.) on the IoT cameras itself. Cameras would only need one connection to the middleware.

In either options, cameras become active or inactive depending on whether they have received a signal from the surrounding cameras or the middleware. When a camera is considered as active it connects to an edge to make use of the faster and more accuracte detection algorithm.

We will first conduct a case study in order to get familiar with current camera hardware and network connectivity to precisely simulate a real world. Second, as [7] didn't do any experiment we can compare against, we will implement the two approaches we explained previously on top of this simulated world.

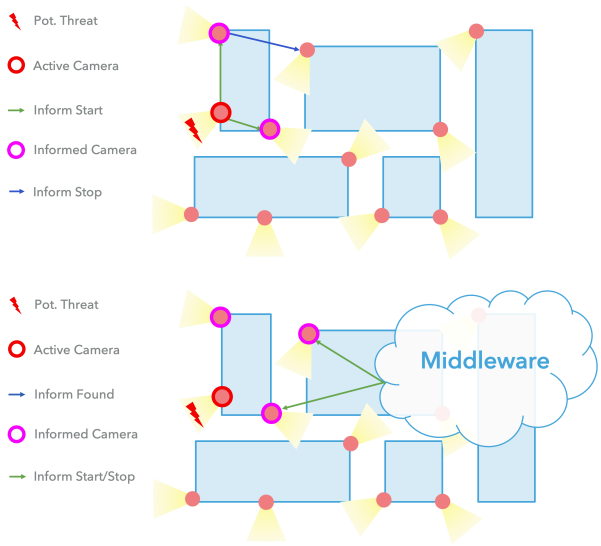

Fig. 1. Direct (Top) and indirect (Bottom) communication among cameras. For clarity the connection from active or informed cameras to the video analyzer is left out.

References

[1] F. Araújo and L. Rodrigues. Geopeer: A location-aware peer-to-peer system. In Third IEEE International Symposium on Network Computing and Applications, 2004.(NCA 2004). Proceedings., pages 39–46. IEEE, 2004.

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services, pages 49–62. ACM, 2010.

[3] J. Dick, C. Phillips, S. H. Mortazavi, and E. de Lara. High speed object tracking using edge computing. In Proceedings of the Second ACM/IEEE Symposium on Edge Computing, page 26. ACM, 2017.

[4] J. Dolezal, Z. Becvar, and T. Zeman. Performance evaluation of computation offloading from mobile device to the edge of mobile network. In 2016 IEEE Conference on Standards for Communications and Networking (CSCN), pages 1–7. IEEE, 2016.

[5] P. T. Eugster, B. Garbinato, and A. Holzer. Location-based publish/subscribe. In Fourth IEEE International Symposium on Network Computing and Applications, pages 279–282. IEEE, 2005.

[6] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis. Learning temporal regularity in video sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 733–742, 2016.

[7] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling video analytics systems to large camera deployments. In Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications, pages 9–14. ACM, 2019.

[8] Y. Kaneko, K. Harumoto, S. Fukumura, S. Shimojo, and S. Nishio. A location-based peer-to-peer network for context-aware services in a ubiquitous environment. In 2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops), pages 208–211. IEEE, 2005.

[9] S. K. Kumaran, D. P. Dogra, and P. P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. arXiv preprint arXiv:1901.08292, 2019.

[10] Y. Li and W. Gao. Muvr: Supporting multi-user mobile virtual reality with resource constrained edge cloud. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), pages 1–16. IEEE, 2018.

[11] S. W. Liu, H. Y. Ngan, M. K. Ng, and S. J. Simske. Accumulated relative density outlier detection for large scale traffic data. Electronic Imaging, 2018(9):1–10, 2018.

[12] Y. Liu, L. Xiao, X. Liu, L. M. Ni, and X. Zhang. Location awareness in unstructured peer-to-peer systems. IEEE Transactions on Parallel and Distributed Systems, 16(2):163–174, 2005.

[13] C. Lu, J. Shi, and J. Jia. Abnormal event detection at 150 fps in matlab. In 2013 IEEE International Conference on Computer Vision, pages 2720–2727, Dec 2013.

[14] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 19(3):1628–1656.

[15] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 935–942, June 2009.

[16] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. IEEE Transactions on Image Processing, 26(4):1992–2004, April 2017.

[17] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette. Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. Computer Vision and Image Understanding, 172:88–97, 2018.

[18] I. Wyze Labs. Wyze Cam camera specification. https://www.wyze.com/wyze-cam/. Accessed: 2019-03-23.