Multi-viewed Live Video Streaming System for UAV Swarms

Hongpeng Guo and Xiaocong Yu

Abstract

The technology advancement has made Unmanned Aerial Vehicle (UAV) swarm a promising method to achieve complicated missions that a single UAV cannot support. UAV swarms are thus usually applied to take and stream multi-viewed live videos in many event driven scenarios. However, the wireless uploading link is easily get saturated when large number of drones stream their high quality videos concurrently. A more scalable and bandwidth efficient video streaming must be developed.

In this work, we propose a multi-viewed live video streaming system, which makes use of the spatial redundancy among the frames from different perspective cameras. We also design a drone-edge collaboration solution to best utilize their computational resources to facilitate the video processing computation. We also implement a prototype and design experiments on real world multi-viewed dataset to justify the performance of our protocol. The experiment results suggests that our designed system could save the uploading bandwidth by 18% to 32%. However, we did not achieve the real-time video processing features in this semester long project. Several future improvement solutions is proposed in this paper.

1 Introduction

The Unmanned Aerial Vehicles (UAVs), also known as "drones", are aircrafts that fly without humans on board. UAVs are controlled either by remote controllers or on-board computer systems autonomously [6, 7]. Advances in technologies allow drones to carry cameras, sensors and communication units on board. They are thus applied to take and stream videos for many surveillance or event driven scenarios, such as football games, Rock concert and farm observation [9–11]. To achieve better observation of the target event, the drones within a swarm will take videos from different angles and provide viewers with video streams with multiple viewing angle options. Through the multi-viewed streaming, viewers can have a more comprehensive picture of the live event. As in a football game, the viewers could watch different players through live videos taken from different angles.

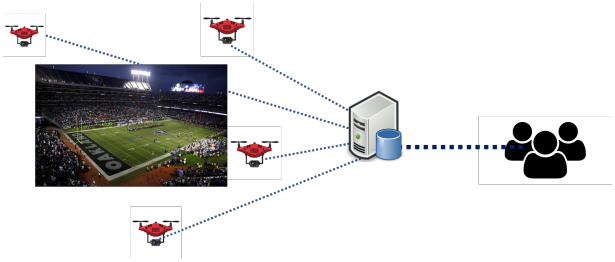

Figure 1: Overview of the Multi-Viewed Video Streaming System

However, Continuous video transmission from a swarm of drones places severe stress on the wireless spectrum [9, 10]. Hulu estimates that its video streams require 13 Mbps for 4K resolution and 6 Mbps for HD resolution using highly optimized offline encoding [5]. Live streaming is even less bandwidth-efficient, A 25 FPS HD video stream occupies bandwidth of 10 Mbps. Only 50 drones transmitting HD videos continuously will saturate the theoretical uplink capacity in a 4G LTE cell which covers a large area, regardless many mobile phone users will further occupy the limited bandwidth.

In this work, we designed a bandwidth efficient system to stream multi-viewed live videos from a swarm of drones. As shown in figure 1, all the UAVs will upload their real-time video feeds to a nearby edge server through wireless connections. The edge server will further distribute those videos to the viewers all over the world leveraging Ethernet infrastructure and CDN architecture. In this project, we only focus on the first part, optimizing the wireless uploading bandwidth from drones to the edge server.

We observed that most of multi-viewed videos taken from different drones have large overlapping on the background scenes. These background scenes are redundant to be uploaded by every UAV. By doing video analysis on the drone

videos, we aim at eliminating redundant information being uploaded and thus greatly reduce the upload bandwidth. After collecting videos from each angle, the edge server will be able to recover the cropped video frames by smartly *copy-pasting* the removed area leveraging the frames in another view. Overall, we plan to develop a novel swarm-edge collaboration solution that can jointly optimize what part of content being uploaded by each drone to the server by taking into account drone location, camera orientation, wireless link strength and the computational resources on board drones and servers, to provide the best quality of service for users. We conclude two major challenges to realize key updates as follows.

- **Real-time Requirement.** To achieve live video streaming, the frames cropping and recovering must be handled by the drones and edge server within the duration between two consecutive frames, which is 40 ms in case the frame rate being 25 fps. This requirement is pretty challenging. Many neural network involved video analysis method cannot finish inference on a single HD frame within 40 ms [10], regardless handling the frames from all angles.
- **High Quality Requirement.** The quality of the videos should not degrade too much after the cropping and recovering process. As we reduce the pixels of a frame being transmitted and wish to recover them using redundant information, we must ensure the recovered video is close to the original one and will not cause severe quality degradation.

In this project, we present the design and implementation of the multi-viewed live video streaming system, which remove the spatial redundant information among the multi perspective video frames, and thus greatly reduce the bandwidth for video uploading. At the beginning, all the drones obtain their camera calibration matrices by themselves. The drones will then send their calibration matrices to the edge server, so that the edge server could determine their overlapping area and make the cropping decisions for the drones based on the calibration information. The edge server will then send the cropping decisions to the drones, according to which the drones will crop their frames. Finally, the drones upload the cropped frames to the server, and the server recover the frames using redundant information. We outline the major contributions as follows:

- We design an bandwidth efficient multi-viewed video streaming solution to optimize the content to be transmitted leveraging the spatial relation among the frames in a multi-viewed scenario, which is novel and not studied in the literature.
- We propose a drones-edge collaboration architecture to best utilize the computational resources of drones and the edge server, and provide a straightforward way to reach consensus cropping decision with the central edge server.

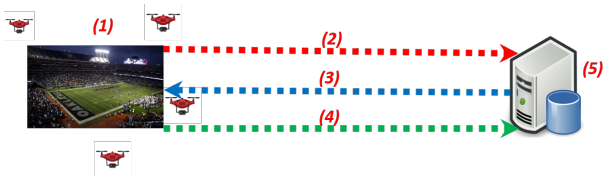

Figure 2: Overview of the System Workflow.

- We build a prototype based on real-world multi-viewed data set [2]. The experiment results suggest we achieve substantial bandwidth saving for the video streaming of 4 views.

However, there are still many problem we did not fully solve in this semester long project. For example, our current prototype system cannot achieve real-time video streaming. The recovered video quality is not as high as the original video. We will provide the improvement plane and future works in this report.

The rest of this report is organized as follows. Section 2 will show the details of our system design. We present the experiment and demo in section 3. Finally, we discuss the advantages, disadvantages and future works in section 4.

2 System Design

2.1 System Overview

As shown in figure 2, the overall workflow of our multi-viewed video streaming system contains five steps as follows.

- Drones calculate their calibration matrices by themselves.
- Drones send the calibration matrices to the edge server.
- Edge server calculate the overlapping area of drones and send the cropping decisions to drones .
- Drones crop their video frames and send the frames to the edge server.
- The edge server recover the cropped content for each frame using the spatial redundant information of other drones.

We present the technical details in the rest of this section.

2.2 Drone Camera Calibration

Camera calibration has been a mature technology in the area of computer vision. After camera calibration, it is easy to obtain a homographic transformation matrix from one perspective view to another. One of the most famous camera calibration instance is *chessboard calibration* as shown in

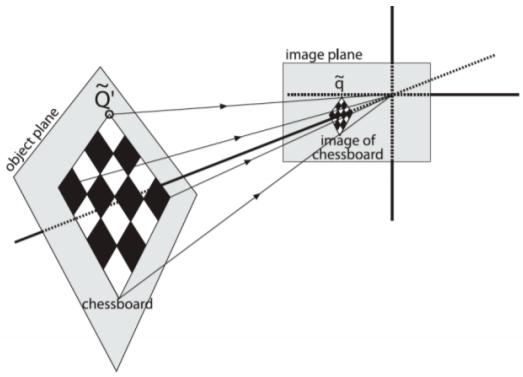

Figure 3: Chessboard calibration.

figure 2.2. Chessboard is used for camera calibration because there are many Connors in the board for easily detection. Similarly, in our football game scenario, many boundary lines and Connors are easy for drones to detect and refer to for calibration.

In this report, we will not provide the technical details of camera calibration. After the calibration process finishes, every drone will obtain a homographic calibration matrix which could map its perspective view to a common 2D plane. All the drones will send their homographic matrix to the edge server for further overlapping detection.

2.3 Overlap Detection & Crop Decision

After obtaining the homographic matrices from the drones, the edge server will calculate the overlapping areas for the perspective views of all the drones. Note that the homographic matrix of each drone will transform their view to a common 2D plane. The server will simply project the drone perspective to the universal plane leveraging the homographic matrices of each drone. In the overlapping detection step, the server will require nothing but a 3×3 matrix from each drone.

We present the overlapping detection using example as shown in figure 4 and 5. Figure 4 presents two views of a laboratory, which are two frames extracted from two perspective videos. The edge server will project the two perspective to a common plane, the overlapping area will immediately display as shown in Figure 5. When the overlapping area is detected, the edge server will make the decision, which view keeps the overlapping area and which drop the overlapping area. In our design, we simply choose the view with high pixel rate to be preserved. In this way, there will be less quality drop when doing homographic transformation from a high pixel density view to a less pixel density area.

2.4 Human Detection & Frame Cropping

Object detection and segmentation is a hot researched area where many advanced models using deep neural network have reached considerably high accuracy in those related

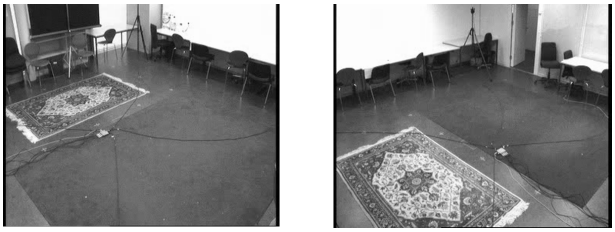

Figure 4: Two Views of a Laboratory.

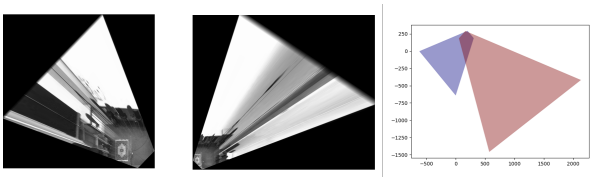

Figure 5: Overlapping Detection Demo.

challenges and made impact with real life application. We use the Mask-RCNN [3], off-the-shell model from torchvision library, here to achieve the goal of identifying areas that we want to preserved during the transmission for overlapped area. The model is built up on Faster-RCNN, which is able to predict the bounding box and class score of potential objects in the image, and it adds another branch for Masks of instances along the training and prediction process.

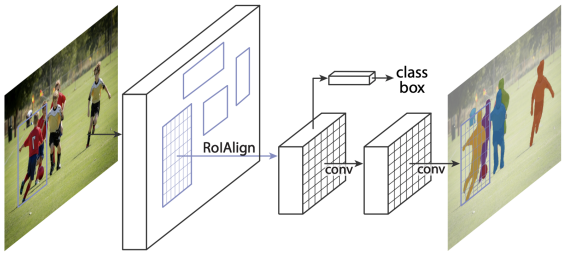

Figure 6: Mask R-CNN framework for object segmentation.

The model, pretrained with a subset of COCO dataset which contains 91 different classes is initialized during environment preparing stage and is used when processing each frame of the input videos. Masks representing segmented instances are returned, so that we are able to generate the bandwidth-reduced output for a single client by applying the segmentation mask and reverse mask of overlap mask with original frames using logical "AND". The second row of Figure 7 show these intermediate states of transmission. The bandwidth-reduced output serves the major functionality of reducing the overall bandwidths of all remote clients.

2.5 Video Recovering

The Video recovering involves two parts. In our senarios this stage happens on the main server which gets high com-

puting resources and is capable of achieving any intensive jobs. After receiving frames frrom remote clients, the first is to transform the frames from most prioritized view according to the transform matrix between current view and that one, which is also updated with control channel. The transformed frames will be cropped and fitted within the vision of current view with openCV library. The second is to overlap the bandwidth-reduced output of client onto the transformed frames to recover the original frames from current view and the frames are written to designated output file at the same frequency of the original videos in our experiments. The third row of Figure 7 are output of current version of the program.

3 Experiments & Demo

In this section, we will present a demo to illustrate how our system saves the bandwidth of multi-viewed video streaming. The following of this section contains three parts. Section 3.1 introduces the multi-viewed video dataset we used in the experiment. Section 3.2 presents all the intermediate steps using processed frames of the test dataset. Finally ,we provide the measurement results of bandwisth saving and discuss about the result we got in section 3.3.

3.1 Dataset Description

We test our experiment using the EPFL multi-viewed video dataset [2]. This dataset provides four different perspective cameras taking videos of a laboratory. These four videos are all of length 157 seconds and have frame rate as 25 fps. The frame size of these videos is 288 pixels *times* 360 pixels. This dataset is originally used for human detection from multi-viewed videos. Several human go around in the laboratory, which increases the difficulty for us to crop and recover the video in our experiment.

The EPFL dataset provides the calibration matrices for the four cameras, we thus start our experiment from the step of overlapping detection. For the simplicity of experiments, we transformed the four videos into gray-scale and only deal with the first minute of the four videos. In the experiment, we crop the video frames and later recover the cropped area using spatial redundant information. All the code and generated videos can be retrieved from our project Github link [4].

3.2 Demo Walk Through

We show all the intermediate results of our video processing in Figure 7. We pick one frame from each of the four original videos listed in the first row of Figure 7. We remove the redundant overlapping area of view 2, 3 and 4 but keep that area in view 1. The four reduced frames are shown in the second row of Figure 7. It is easy to observe that large area of the background scene is blacked out. The corresponding frame size is thus greatly reduced. The reduced frames should be directly created on board drones. Due to limited of time and resource, we implement everything with our local machine. After the reduced frames are generated, they will be transmitted to the edge server for recovery. The third row of Figure 7 shows the four recovered video frames. The third row of Figure shows the four recovered video frames. For view 2, 3 and 4, the missing background information is retrieved from view 1. After the corresponding homographic transformation, the overlapping area will be applied to the other three views.

Although the recovered frames are close to the original view, there are still several problems associated with the frame quality. It is easy to observe that the background pasting is not seamless, the tables and chairs at the edge of the frames have clear distortion. Human in view 1 is also distorted on the floor like a shadow. We will further discuss the corresponding solutions in our future works of section 4.

	View 0	View 1	View 2	View 3	Total
original video sizes	15.17 MB	20.35 MB	16. 27 MB	22. 59 MB	74.38 MB
cropped video sizes	15.17 MB	15.23 MB	13.72 MB	17.85 MB	61.97 MB

Table 1: Reduction of video sizes

	View 0	View 1	View 2	View 3	Total
original video sizes	96 KB	114 KB	106 KB	108 KB	424 KB
cropped video sizes	96 KB	54 KB	72 KB	68 KB	290 KB

Table 2: Reduction of sum of frame sizes

3.3 Measurement and Discussion

Besides generating the video demos, we also make two set of measurements to justify (1) how much bandwidth can we save comparing with the naive video transmission method. (2) how much time will cost in our video processing pipeline.

As the video frames going through network will only be the cropped ones. Therefore we compare the data being transmitted between the cropped videos and the original videos. We further compare our video streaming method with the naive one on two matrices as follows.

- Compare the size of four original 1 min videos with the four cropped 1 min videos.
- Compare the average frame size of the four original videos with that of our four cropped videos.

The reason for these two metrics is as follows. Because modern video compressing method, like MP4, AVI, will compress the video size to be far less than the simple sum of all the frame sizes, the real-time video streaming usually send videos feeds in the form of several compressed frame batches. In our current implementation, we did not build the whole

Figure 7: Reduced frames of four views.

streaming system. But the real bandwidth saving must stay in between the results of these two metrics.

The comparing results for the compressed videos are presented in Table 1, and the results for sum of the frame sizes are presented in Table 2. For the compressed videos, we save about 18% of total data being transmitted. For the submission of frame sizes, we save about 32% of total data. Overall, our multi-viewed video streaming system can save bandwidth significantly.

We also measuring the processing time cost of our video processing pipeline. We measure the computational cost to generate 2 seconds of cropped video from an original video. The detailed mapping between time elapsed and specific action are shown as follows. For a single frame, the time for overlapping area detection about 40 ms, while the time for human detection using Faster-RCNN is around 4.3 s. Overall the performance cannot achieve the real-time requirements. However, we may change our human detection model to some weaker but faster model to achieve more efficiency. Also, as we do not use any advanced hardware for this task, it will be possible to speed up the overall video streaming pipeline by introducing GPUs or other accelerators.

4 Conclusion

In this project, we design and implement a prototype for multi-viewed live video streaming system for UAV swarms. By observing the spatial redundancy of the multi-viewed video frames. We designed a drone-edge server collaboration solution which can greatly reduce the bandwidth usage for uploading drone videos to edge servers. Our prototype video clearly illustrated our proposed method and justify our design goal of bandwidth saving.

However, there are still following points that we will continue work on as our future works.

- Try some weaker neural network model smartly to achieve comparable performance as Faster-RCNN in our application scenario. Some possible reference literature includes [10], [9] and [1].

- Try to optimize the last layer of prediction model according to the task scenarios by supporting only limited specified class types to avoid unnecessary detection and generation of masks

- Try to improve the recovered image quality using some image prior method, such as super resolution and inpainting. Some possible reference literature includes [12] and [8]

- Experiment is carried out using real UAV swarms and real streaming system is built to achieve more convincing measurement results.

References

[1] Zhou Fang, Dezhi Hong, and Rajesh K Gupta. Serving deep neural networks at the cloud edge for vision applications on mobile platforms. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 36–47. ACM, 2019.

[2] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 2007.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. arXiv:1703.06870, 2017.

[4] Xiaocong Yu Hongpeng Guo. Project code for multi-viewed live video streaming for UAV swarms. https://github.com/xyu335/SemanticSeg, 2019.

[5] Hulu. Internet speed requirements for streaming HD and 4K. https://help.hulu.com/en-us/requirements-for-hd, 2017. [Online; accessed May-16-2017].

[6] Mohammad Mozaffari, Walid Saad, Mehdi Bennis, Young-Han Nam, and Mérouane Debbah. A tutorial on uavs for wireless networks: Applications, challenges, and open problems. *IEEE Communications Surveys & Tutorials*, 2019.

[7] Kamesh Namuduri, Serge Chaumette, Jae H Kim, and James PG Sterbenz. *UAV Networks and Communications*. Cambridge University Press, 2017.

[8] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9446–9454, 2018.

[9] Deepak Vasisht, Zerina Kapetanovic, Jongho Won, Xinxin Jin, Ranveer Chandra, Sudipta Sinha, Ashish Kapoor, Madhusudhan Sudarshan, and Sean Stratman. Farmbeats: An iot platform for data-driven agriculture. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017.

[10] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018.

[11] Xiufeng Xie and Xinyu Zhang. Poi360: Panoramic mobile video telephony over lte cellular networks. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*. ACM, 2017.

[12] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. Neural adaptive content-aware internet video delivery. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 645–661, 2018.