D2V: Drone data ingest mechanism for Video DB

Bharati Khanijo*, Harshil Gupta and Yogesh Simmhan

Department of Computational and Data Sciences
Indian Institute of Science, Bangalore 560012 INDIA
Email: bharatik@iisc.ac.in,harshilgupta@iisc.ac.in, simmhan@iisc.ac.in

***Abstract*— With advances in UAV & computer vision, automated management & analysis of video data captured by UAV mounted cameras is an area of growing interest. Video & metadata collected by a fleet of such UAVs need to be stored at a backend server to answer historical queries. Managing video data is known to be challenging due to its volume. But the videos captured by a fleet of drones are additionally plagiarized by visual information of varying level of detail & shorter duration due to which efficient analysis (by a single model) can be challenging, as well as rules out traditional techniques to reduce volume of video by downscaling it to a predetermined static resolution. Here, we analyzed existing methods to measure the concept of level of detail in the literature, proposed & built a data (video & metadata) processing pipeline to scale videos captured by UAVs dynamically (using metadata captured by drone sensors) so as to meet user specified level of detail configuration on average. We implemented the above pipeline on a heterogeneous edge cluster & observed that proposed preprocessing can reduce average turn around time to ingest data from multiple clients to a central server by 65.97 %for the evaluated workload. To better manage the load across devices we integrated & evaluated 3 load balancers with the above pipeline & observed that VS (video & device heterogeneity aware) load balancer reduced the makespan for observed load by 2-32% on average when compared to out of the box Round Robin load balancer in Apache NiFi.**

I. Introduction

Drones ease video data collection due to their mobility & easy deployment, thus capturing high resolution videos from diverse viewpoints. The videos are also accompanied by dynamic metadata, capturing temporally varying location & orientation of the drone & static metadata about the camera's intrinsic properties. The video & metadata collected can be stored & used later for answering historical queries. This can be achieved with the help of video databases that can query & fetch video(s) or a segments of it, not only based on the spatial location of camera [1], [2] & time, but also based on expected region that the video is looking at in space [3], [4] & the semantic objects present in the video [5], [6], [7]. With advances in deep models, the task of semantic object detection can be accurately carried by deep models. But the performance of deep models varies for different sized objects (in pixel) [8], [9]. A collection of short videos collected from a fleet of drones with different resolution cameras, flying at different altitudes & having different orientations is expected to have large variation in level of detail thus in turn in the size (in pixel) of the semantic objects, making it difficult to choose a single model to achieve desired performance for semantic object detection at the time of analysis/query. The analyst could overcome this by a maintaining a collection of models & map which video(s) will be analyzed by which model, which is a cumbersome task. Alternatively, the analyst could also pre process the videos to achieve similar level of details, so that the object sizes are similar (in pixels) across videos.

For static cameras this can be done manually as a one time effort, but due to mobile nature & short trips (or videos) of drones, this can become a labor intensive task if done manually for each video. Thus we design a system in which the user can specify a single configuration parameter specifying the expected level of detail in a video & the system should be automatically able to select appropriate scaling factor & convert the visual information into an appropriate resolution so that the video's mean level of detail matches the user specified level of detail parameter. The details about this parameter & integration into the system are discussed in sections II & III respectively.

This pre-processing of video data & computation of secondary derived metadata like the camera footprint of the video can be computed using the near edge resources, to save time to upload data corresponding to a trip, compared to uploading raw data from the drone IV-B. We have developed an ingest pipeline using Apache NiFi [10] that performs the above pre processing by efficiently using the edge accelerators (eg. Jetson [11]) that are specialized for lightweight computing & video processing.

In real world, we cannot assume that the edge accelerators will have homogeneous compute capability [12] , so it becomes necessary to have a load balancer that will schedule the above pre-processing based on the compute capacity of the device. To address this we have integrated static load balancer(D) (fig. 1) to distribute the load across heterogeneous near-edge devices.

Specifically, our technical contributions are as follows

1) We propose controlling the variation of level of detail in drone video data by dynamically down scaling the video as per the user specified level of detail.
2) We have developed a drone data (video & metadata) processing pipeline for the same over a heterogeneous cluster of edge accelerators. The stages of the pipeline are designed to utilize the benefit of pipelining to speed up data processing
3) We have integrated & evaluated 3 load balancers (RR, Wt. RR & VS) IV-B with the above pipeline. We have

reported makespan time & the compute overhead for the load balancer to schedule the requests.

II. Related Work

Level of Detail : The concept of level of detail (LOD) of visual information available has been modeled differently in different domains. In vision domain, it is modeled by the resolution of the video, i.e. more resolution corresponds to more level of detail. But this model does not work for a collection of videos captured by drones. This is because for the same resolution, based on altitude & orientation of the drone camera, the level of detail of information captured about the objects on ground can be different. Another model commonly adopted by the geospatial community considers only the spatial information, in terms of area of the camera footprint [13]. According to this model, if the area of camera footprint is more, then less level of detail of information will be captured. The above model does not work in the case cameras of different resolutions are used to capture different videos. In our work we plan to use the concept of *Ground Spatial Distance (GSD)*, which models the geometric resolution or the distance on the ground for a given pixel represented in the image. There are numerous applications that utilize the concept of LOD [13], [14], [15], but to the best of our knowledge this is a first work to utilize this concept as a video pre-processing step in an edge based pipeline, for drone videos.

Video Processing near the Edge : Video pre processing systems near the edge have been typically developed to support standing queries which focus on using limited compute on edge to (partially) analyze video based on a predefined query & reduce the compute & network overhead at backend [16], [17]. One technique commonly used is to filter out frames based on query [17], [18]. Application specific queries like person re-identification [19] have also use edge resources to achieve a target latency.

It is to be noted that the above systems requires the task to be known before data is ingested in the system, which may not be always possible like in case of *historical queries*. Systems that support latter [4], [5], [20] typically assume that the video data is available at the backend server at fixed low resolution (To reduce its volume) [21] although limited systems do exist utilize compute near the edge to reduce compute overhead at backend [6], [22], [23], [24] . These are based on the assumption that all possible classes that need to be detected are known at the time video data is ingested into the backend server & models to accurately detect them are also available. But with fast advances in deep learning, the model performance improving at a rapid pace [8], [25], [26], [27], [28], [29], [30], [31], [32], this assumption can prove as a drawback. So instead we rely on the assumption that the user can specify the level of detail & pre processing (video scaling) is done near the edge in our system, allowing SOTA models/systems to be used at query time.

Load balancers : Load Balancing algorithms play a pivotal role in efficiently utilizing the capacity of distributed systems. A single load balancer (LB) can be categorized as static or dynamic based on the time when load balancing takes place [33]. Latter involves dynamically redistributing workload to less utilized resources, but frequent redistribution adds to additional overheads like task migration etc. Thus static load balancing strategies need to be efficiently balance majority of the load based on the requests. Traditional load balancing algorithms [34] include Round Robin (RR), selecting a server with fewest active connections (based on compute capability), selecting a server based on fastest response time & least active connections, hash based algorithms & Random with 2 choices algorithm [35]. These techniques either statically balance the load, or use the state of the server to send the request. Common assumption for above LBs was that requests are expected to contribute similar load to the system. This assumption is not valid in the proposed pipeline that considers drone video & metadata as a primary payload, whose processing time can significantly vary based on duration of video etc. There has been work in literature that considers predictive methods over similar workloads & tasks, like media transcoding [36], motivating us to integrate & evaluate VS algorithm III.

III. Implementation

The proposed system is based on the assumption that drones will transfer video & temporally varying metadata to clients when they land on a base station, say to charge. These clients then send request to our system as depicted in Figure 1. The request sent by client includes temporally varying 3D location , orientation of the drone & ftp link(s) to download the video(s) from the client. The request lands on the primary node of the edge cluster. Finally the request to upload trip data is sent to Load Balancer (RR) to be uploaded to spatio-temporal DB. At the same time, request is sent to load balancer (D) to process video data & associated metadata by the pipeline marked in orange (ingest pipeline) in fig 1. At the time of splitting, we assume that multiple videos might be recorded during the trip of UAV, that can be processed in parallel by different devices to speed up time turn around time (of uploading the requested trip to the server). As the most compute & network heavy part of the system is expected to be this pipeline, we try to optimize & evaluate data processing in this part of the system.

We first split the trip data (if multiple videos are present) thus utilizing data parallelism. We then integrate & evaluate different techniques for load balancing on this pipeline to efficiently distribute the load on heterogeneous edge devices. In order to further speed up the processing , we reap the benefits of pipeline based parallelism. The 4 stages (in orange) in fig 1, can work in parallel on data associated with different videos (for instance when multiple clients request to upload data) . It is to be noted that the design of the stages is such that different stages are expected to predominately use different hardware capability in the stages network (download to device & upload from device) & hardware accelerated encoding & decoding [37].

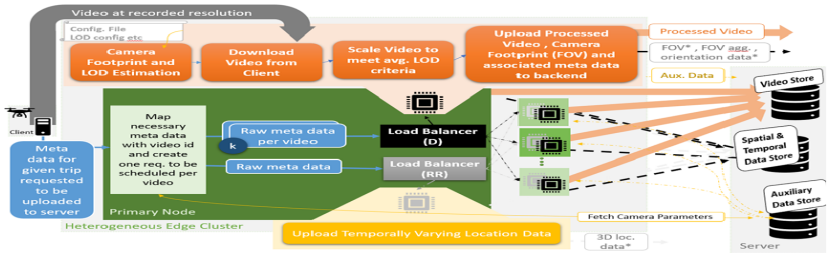


Fig. 1: Architectural diagram of the system having edge devices of variable compute capacity. The data marked with * varies with time & its size is arrow's thickness. The load balancer (D) is connected to the pipeline in orange which is responsible for majority of pre-processing of video data & represents the performance of the entire system.

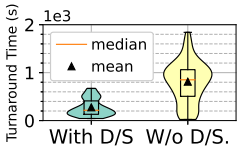


Fig. 2: Comparison of time to process trip via pipeline with and without downscaling (d/s).

Stages of Pipeline: The 1st stage of the pipeline, calculates temporally varying camera footprint (modeled as a quadrilateral [38]) and scaling factor. For cameras looking near the horizon, the proposed system has specified a parameter (maxGSD) that will automatically limit the extent of polygon to represent region expected to be visible. Further for each of the above camera footprints, GSD for center of this region is calculated & is assumed to be the representative GSD for the given footprint. In order to calculate the representative GSD for the entire video, the mean of all the representative GSDs associated with the video is taken. The scaling factor for the video is then calculated from the above representative GSD for video & the user specified LOD parameter. The accuracy of the above level of detail metric is dependent on the correctness of metadata provided by the drone. It may be noted that we are mapping the worst level of detail possible which is bounded by the above metric of GSD. It is possible that a given video has better visual details due to presence of objects nearer to the camera occluding the regions with poorer level of detail.

The 2nd stage fetches the video from the client to the device which will be processing it in the next stage. In the 3rd stage, the video is scaled by factor computed in the 1st stage. It is to be noted that if the scaling factor suggests upscaling the video, this stage will skip the upscaling part as the actual detail of the visual information cannot be enhanced by simple upscaling. The final stage in the pipeline, uploads processed video to video store, temporally varying camera footprint & aggregate footprint of the video to spatial & temporal DB [39], [40], [41] & deletes video data once it has been uploaded to the server.

Load Balancing : Our baseline LB is RR offered by NiFi out of the box. Next we implement a load balancer that is aware about the compute capacity of the heterogeneous devices (Wt. RR). The information for relative compute capability is provided by benchmarking the pipeline on each device & measuring the median execution time for the benchmark workload.

Finally we integrate a load balancer that is not only aware of the compute capacity of the device, but also the load associated with the request (VS) . This addition was based on the observation that processing time in the pipeline was highly correlated with video duration. To rectify this we introduce a parameter that takes into account the video duration as well. We calculate the median time taken ($M(ps)_i \; \forall \; i \in D \; where$

Algorithm 1 Wt RR Load Balancing

```
procedure FindNode
    Set D ← {EdgeDevices}
    for Device d ∈ D do
        LoadonDevice L_d ← FromCacheServer(d)
        MedExceTime M_d ← MedianExpectedExecutionTime(d)
            ▶ Find device, video should get processed in the least time
        picked_node ← min(L_d + M_d)
    end for
    SendToNode(picked_node)
    AddToLoadCache(picked_node, M_d)
            ▶ Reduce load in cache when video processing completes
end procedure
```

$D = \{edgedevices\}$) to process a second of a video from the benchmark done above. This $M(ps)_i \times$ (Video Duration) gives the expected execution time E for that video. We use E in load L_i, thus modeling load better.

IV. Experimental Evaluation

A. Experimental Setup

Datasets : We performed detailed evaluation of the three load balancers using two drone video datasets, Zurich Drone Dataset [42] & Au-air Dataset [43] both having full HD videos. Zurich Drone Dataset has metadata for a single flight of the drone. In order to increase the number of trips ingested, we created a synthetic workload that used, 2D spatial location of the trips from Geolife Trip dataset [44] & temporally synced altitude, orientation & video data from the Zurich drone dataset. A subset of 99 trajectories were then selected from Geolife dataset with trip duration ranging from 6:43 mins to 24:44 mins. Video data was accordingly trimmed to match the trip duration. Finally to simulate the case of a single trip having multiple videos, the videos & corresponding orientation data, associated with a trip are chunked into segments of max duration of 5 mins. This synthetically created workload will be referred to as Zurich Dataset in the rest of the paper.

Au-air on the other hand has 8 trips of different duration & each trip has 1 video along with associated metadata. We replicate these 8 trips to simulate a longer experiment.

System Setup : Our experimental setup consisted of a cluster of 5 edge-accelerators of variable compute capacities (Nvidia Jetson [45] AGX Orin , Jetson TX1, Jetson TX2, 2 Jetson Xavier NX). We make AGX Orin as the primary node in NiFi because of the extra cores. The cache server is running on the Jetson Orin & other devices submit their cache updation requests to this node. All the devices: Edge,

clients, DB server are connected to the same switch to get a cross-plane bandwidth of 1Gbps.

Simulation Setup : We also performed simulation experiments between the 3 LBs. To benchmark the simulation selected 25 videos (with associated metadata) from the Zurich Dataset such that their duration is uniformly distributed, & logged time spent by them in each phase (without contention). Using these data points we applied linear regression to estimate time taken to process a video in for each phase, given its duration. These times were then used as proxy values as the time spent in the pipeline for processing for a single video metadata combo.

Default Workload : In total 120 trips were sent from 5 clients. We kept a wait of 20 secs between firing of next request from each client. This was done to simulate the scenario of landing of drone & uploading the video(s) it had recorded after it had completed the set of tasks that were assigned to it.

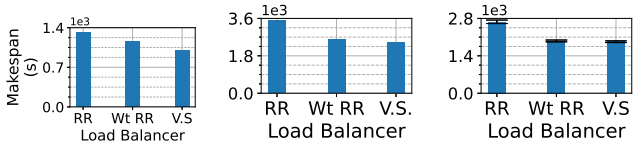
B. Experimental Results

Video processing near the edge: We evaluated the design decision of downscaling then uploading the video near the edge against uploading the full resolution video directly. In order to simulate scenario of large number of videos being concurrently uploaded to a central server, we added 12 edge accelerators (AGX Orin,TX1,TX2,4 NX,5 Nano) to the cluster setup that process trip & video data from 5 clients sending trip data frequently every 20 seconds. In total, 100 trips from Zurich Dataset were ingested in this experiment.

As shown in figure 2 , we observe that even after incorporating time overhead for downscaling the video near the edge, we see significant average reduction of 65.97 % in turnaround time of a trip in pipeline, as compared to uploading videos at the resolution at which they were recorded. It may be additionally noted that for the above workload, the downscaling factor, for a side, ranged between 3.9 & 2.3 to achieve the user specified mean LOD.

Load Balancer : We observed that both the implemented algorithms outperformed the default baseline of RR LB for both the workloads in terms of improvement in makespan. Makespan here is the end to end time to process the workload (aka run the experiment). In terms of load balancing overhead at runtime (decision logic), on average there was 0.23 s reduction compared to RR (0.56 s).

Evaluation using Au-air dataset : We ran the experiment with the experimental setup as given in section IV-A. We report observed Makespan (in s) across different load balancers in figure 3a. It is expected that better load balancing strategy should reduce makespan, by efficiently balancing the load. We observed that V.S. outperform Wt. RR by 13.5 % (155 s). This can be attributed to the characteristics of the workload that has one video per trip & the wide variability in video duration (2.57 mins to 23.29 mins)



(a) Makespan for AU AIR (b) Makespan for simulated experiment (c) Makespan for live Experiments

Fig. 3: Experimental evaluation of Load Balancer

Evaluation using Zurich dataset : As observed in figures 3a 3c 3b , VS and Wt. RR consistently out performs RR across workloads and setups. But on running the Wt. RR and VS multiple (4) times, with Zurich workload fig. 3c, we observed that there was variability in make span of both. On further investigation we observed that there was variability in the relative order in which reduce load requests reached LB compared to add load (FindNode) requests. This variability may be attributed to variation in time to process a request and the variation in time that this information reaches LB.

We also performed a simulation experiment to control the above variability and used the same workload and decision making logic of the 3 LBs. We simulated the actual execution time of the videos from the linear fit estimated during benchmarking.

We observed that VS load balancer was able to reduce make span by $\approx$ 180 s as compared to Wt. RR.

For the non-simulated (aka live) setup, fig. 3c, maximum improvement of 154 s was observed between Wt. RR and VS. It was also observed that the mean difference between the 2 LBs, was very less. On further investigation we observed that this may be attributed to multiple factors, one of which being the variability in makespan as observed above. The other was that in the given workload, due to the presence of multiple videos in a single trip (in contrast to Au-Air workload), available bandwidth decreased compared to benchmark setup. This was because, on average more than one video was fetched from the client in parallel (by different devices), thus increasing the download time compared to benchmarked download time. Thus reducing the possible overestimation of processing time by Wt. RR LB.

V. Future Work

We expect to extend this work more accurately by modeling the load (eg. sampling rate, video size etc) & system level variation like bandwidth contention. We plan to control the level of detail at a more granular level (video segment) to handle temporal variation in altitude etc. in the video & empirically observe the savings. We also plan to numerically estimate the scaling property of the proposed pipeline by varying the no. of devices in the cluster.

VI. Conclusion

In this paper we proposed a system to control the variation of LOD across drone videos by scaling it to meet user specified criteria. We also experimentally observe that processing the video data if done near the edge can reduce average turn around time to upload a trip by 65.97 % , thus making the data available faster at backend server for querying. Our evaluation of load balancer(D) fig 1 over the existing system indicated that both VS & Wt RR were able to reduce the make span better compared to inbuilt RR.

VII. Acknowledgements

We acknowledge the work done by Sharath Suresh Bhargav & Rounaq Choudhuri for their contribution to parts of development & documentation of the proposed pipeline. We would also like to acknowledge Prashanthi SK & Aakash Khochare for their reviews.

References

[1] Y. Creators, "How to tag your location on videos & live streams," Available at https://www.youtube.com/watch?v=7wB5z8dLtNo (2019/05/21).

[2] A. Khochare, A. Krishnan, and Y. Simmhan, "A scalable platform for distributed object tracking across a many-camera network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1479–1493, 2021.

[3] Y. Cai, Y. Lu, S. H. Kim, L. Nocera, and C. Shahabi, "Querying geo-tagged videos for vision applications using spatial metadata," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, pp. 1–18, 2017.

[4] S. A. Ay, R. Zimmermann, and S. H. Kim, "Viewable scene modeling for geospatial video search," in *Proceedings of the 16th ACM international conference on Multimedia*, 2008, pp. 309–318.

[5] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: optimizing neural network queries over video at scale," *arXiv preprint arXiv:1703.02529*, 2017.

[6] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying large video datasets with low latency and low cost," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 269–286.

[7] S. H. Kim, A. Alfarrarjeh, G. Constantinou, and C. Shahabi, "Tvdp: Translational visual data platform for smart cities," in *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 2019, pp. 45–52.

[8] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[9] L. Cui, "MDSSD: multi-scale deconvolutional single shot detector for small objects," *CoRR*, vol. abs/1805.07009, 2018. [Online]. Available: http://arxiv.org/abs/1805.07009

[10] Cloudera, Apache Software Foundation, "Apache NiFi," https://nifi.apache.org/, March 2021.

[11] Nvidia, "Nvidia Jetson series," https://www.nvidia.com/en-in/autonomous-machines/embedded-systems/, March 2021.

[12] J. Jiang, Y. Zhou, G. Ananthanarayanan, Y. Shu, and A. A. Chien, "Networked cameras are the new big data clusters," in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019, pp. 1–7.

[13] F. Gilboa-Solomon, G. Ashour, and O. Azulai, "Efficient storage and retrieval of geo-referenced video from moving sensors," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 404–407.

[14] J. Höhle, "Oblique aerial images and their use in cultural heritage documentation," *Proc. Int. Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 5, p. W2, 2013.

[15] Pix4D, "Ground sampling distance (gsd) in photogrammetry," Available at https://support.pix4d.com/hc/en-us/articles/202559809-Ground-sampling-distance-GSD-in-photogrammetry (2021/06/12).

[16] J. Emmons, S. Fouladi, G. Ananthanarayanan, S. Venkataraman, S. Savarese, and K. Winstein, "Cracking open the dnn black-box: Video analytics with dnns across the camera-cloud boundary," in *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019, pp. 27–32.

[17] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 359–376.

[18] J. Wang, Z. Feng, Z. Chen, S. A. George, M. Bala, P. Pillai, S.-W. Yang, and M. Satyanarayanan, "Edge-based live video analytics for drones," *IEEE Internet Computing*, vol. 23, no. 4, pp. 27–34, 2019.

[19] A. Khochare and Y. Simmhan, "A scalable and composable analytics platform for distributed wide-area tracking," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, ser. ICDCN '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 506. [Online]. Available: https://doi.org/10.1145/3288599.3299753

[20] D. Kang, P. Bailis, and M. Zaharia, "Blazeit: Optimizing declarative aggregation and limit queries for neural network-based video analytics," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, 2019.

[21] Seagate, "Video surveillance storage: How much is enough?" Available at https://www.seagate.com/in/en/solutions/surveillance/how-much-video-surveillance-storage-is-enough/ (2021/07/02).

[22] G. Ananthanarayanan, V. Bahl, L. Cox, A. Crown, S. Nogbahi, and Y. Shu, "Video analytics-killer app for edge computing," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 695–696.

[23] Microsoft, "Microsoft rocket for live video analytics," Available at https://www.microsoft.com/en-us/research/project/live-video-analytics/ (2021/06/25).

[24] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 253–266.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[28] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2349–2358.

[29] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 408–417.

[30] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9217–9225.

[31] H. Zhu, H. Wei, B. Li, X. Yuan, and N. Kehtarnavaz, "A review of video object detection: Datasets, metrics and methods," *Applied Sciences*, vol. 10, no. 21, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/21/7834

[32] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[33] K. Pradeep and T. P. Jacob, "Comparative analysis of scheduling and load balancing algorithms in cloud environment," in *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE, 2016, pp. 526–531.

[34] NGINX, "What is load balancing ?" Available at https://www.nginx.com/resources/glossary/load-balancing/ (2022/09/23).

[35] Netflix, "Rethinking netflix's edge load balancing," Available at https://netflixtechblog.com/netflix-edge-load-balancing-695308b5548c (2022/09/23).

[36] J. Guo and L. N. Bhuyan, "Load balancing in a cluster-based web server for multimedia applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 11, pp. 1321–1334, 2006.

[37] Nvidia, "Accelerated gstreamer user guide," Available at https://developer.nvidia.com/embedded/dlc/l4t-accelerated-gstreamer-guide-32-1 (2022/09/30).

[38] Y. Lu and C. Shahabi, "Efficient indexing and querying of geo-tagged aerial videos," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2017, pp. 1–10.

[39] PostGIS, "Spatial and geographic objects for postgresql," Available at https://postgis.net/ (2021/07/03).

[40] Computer & Decision Engineering Department of the Université Libre de Bruxelles (ULB), "MobilityDB - An open source geospatial trajectory

data management and analysis platform," https://www.mobilitydb.com/, December 2020.
[41] E. Zimányi, M. Sakr, and A. Lesuisse, "Mobilitydb: A mobility database based on postgresql and postgis," *ACM Trans. Database Syst.*, vol. 45, no. 4, dec 2020. [Online]. Available: https://doi.org/10.1145/3406534
[42] A. L. Majdik, C. Till, and D. Scaramuzza, "The zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269–273, 2017.
[43] I. Bozcan and E. Kayacan, "Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8504–8510.
[44] Microsoft, "Geolife gps trajectories," Available at https://www.microsoft.com/en-us/download/confirmation.aspx?id=52367 (2022/09/19).
[45] Nvidia, "Nvidia Jetson series," https://developer.nvidia.com/embedded/jetson-modules, March 2021.