DoveDB: A Declarative and Low-Latency Video Database

Ziyang Xiao
Zhejiang University, China
22021206@zju.edu.cn

Dongxiang Zhang
Zhejiang University, China
zhangdongxiang@zju.edu.cn

Zepeng Li
Zhejiang University, China
lizepeng@zju.edu.cn

Sai Wu
Zhejiang University, China
wusai@zju.edu.cn

Kian-Lee Tan
National University of Singapore
tankl@comp.nus.edu.sg

Gang Chen
Zhejiang University, China
cg@zju.edu.cn

ABSTRACT

Concerning the usability and efficiency to manage video data generated from large-scale cameras, we demonstrate DoveDB, a declarative and low-latency video database. We devise a more comprehensive video query language called VMQL to improve the expressiveness of previous SQL-like languages, which are augmented with functionalities for model-oriented management and deployment. We also propose a light-weight ingestion scheme to extract tracklets of all the moving objects and build semantic indexes to facilitate efficient query processing. For user interaction, we construct a simulation environment with 120 cameras deployed in a road network and demonstrate three interesting scenarios. Using VMQL, users are allowed to 1) train a visual model using SQL-like statement and deploy it on dozens of target cameras simultaneously for online inference; 2) submit multi-object tracking (MOT) requests on target cameras, store the ingested results and build semantic indexes; and 3) issue an aggregation or top-k query on the ingested cameras and obtain the response within milliseconds. A preliminary video introduction of DoveDB is available at https://www.youtube.com/watch?v=N139dEyvAJk

PVLDB Reference Format:
Ziyang Xiao, Dongxiang Zhang, Zepeng Li, Sai Wu, Kian-Lee Tan, and Gang Chen. DoveDB: A Declarative and Low-Latency Video Database. PVLDB, 16(12): 3906 - 3909, 2023.
doi:10.14778/3611540.3611582

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/dovedb/DoveDB_MVP.

1 INTRODUCTION

Surveillance cameras have been extensively deployed in urban city to enhance public safety and intelligent transportation management. These cameras produce continuous flows of live video streams that constitute a massive video database, which is a potential gold mine yet to be exploited. Owing to the rapid advancement in AI technologies, object detection and tracking models have achieved significant breakthroughs and been successfully applied on video data to support various applications (e.g., pedestrian and vehicle tracking, traffic flow estimation and adaptive traffic light control). Nevertheless, these models are computationally intensive and consume enormous GPU resources. According to an assessment in [3], the popular YOLOv3 object detector can only process 100 video frames per second on the $10,000 NVIDIA Tesla V100 GPU. It is thus expensive to deploy them to support real-time object detection in an urban-scale surveillance camera network. For example, given a sampling rate of 30 fps (frames per second), there are 30,000 video frames generated per second from 1,000 cameras. Therefore, in order to support city-scale video analytics, there requires a systematic data management platform with high usability and low latency.

To address the efficiency issue, there has emerged a wave of research attention for scalable video query optimization and a noticeable number of video database systems has been proposed in recent years. Systems like NoScope [7] and TAHOMA [1] adopt the idea of designing a lightweight proxy model to replace the slow-running oracle model, with tolerable accuracy degradation. An alternative strategy, adopted by MIRIS [2] and OTIF [3], for performance speedup is to adopt downsampling to reduce the number of processed frames. Among these systems, OTIF is the only one capable of handling versatile queries, whereas the remaining systems are designed with tailored optimization for a particular type of video query. However, OTIF still requires expensive overhead to find the optimal configuration and its performance is not satisfactory with low sampling rate.

In this paper, we develop DoveDB as a more practical video database with high usability and low latency. Compared with OTIF, it is integrated with the following unique features:

1) Declarative query language. We devise a more expressive video query language called VMQL, with augmented statements to support model-oriented management and convenient deployment.

2) Lightweight ingestion scheme. We propose a lightweight ingestion scheme to extract semantic information from videos and construct spatial, temporal and visual indexes in real time.

We also notice the existence of recent demo papers on video database, such as Vaas [4] for workflow-based analytical task, GNOSIS [8] for video event processing, and SVQ++ [5] for object interactions in video streams. Compared with these works, our DoveDB is towards systematic video data management and query processing, with a comprehensive video-model query language, a complete workflow of ingestion, indexing and query processing, and broader demonstration scenarios for user interaction. We are the first to demonstrate the usability of VMQL for both model training and deployment. Furthermore, we have constructed an environment with 120 cameras to validate our efficiency, which is a significant

Proceedings of the VLDB Endowment, Vol. 16, No. 12 ISSN 2150-8097.
doi:10.14778/3611540.3611582

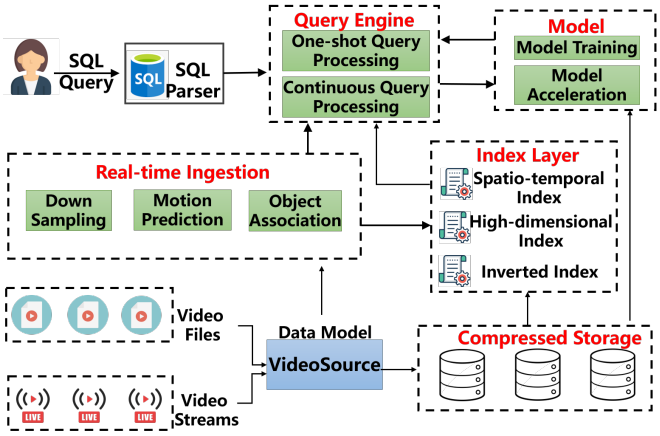


Figure 1: System framework of DoveDB.

improvement over previous demonstrations that were performed on single video clips.

2 SYSTEM OVERVIEW

The system framework of DoveDB is illustrated in Figure 1. Towards uniform management of data sources in the format of video files or live streams, we build an abstract data model called `VideoSource`, which is essentially inherited from Spark's RDD. A real-time video ingestion engine is developed to extract semantic information, including textual labels, visual features and spatio-temporal metadata. The output is then used to construct offline indexes to facilitate online query processing.

DoveDB provides SQL-like syntax to support convenient model training and query processing. Users can train a visual model on specified table columns, where we assume the annotations are available. The trained model can be conceived as a user-defined function and deployed on a target `VideoSource` for online inference. We also provide built-in model acceleration techniques such as neural network compilation and quantization to reduce computational complexity and accelerate inference speed. Our query processing engine, assisted by the constructed offline indexes, can support a diversified category of queries, including traditional selection and aggregation (which are referred as one-shot queries), as well as continuous queries deployed on video streams. In the following, we present the core modules in DoveDB.

2.1 Data Model

DoveDB is built upon Spark and supports queries against historical data and streaming data. We customize RDD to derive an abstract data model called `VideoSource` to uniformly manage data sources in the format of video files or live streams. The following SQL statement encapsulates a disk file as `VideoSource`.

```
-- create VideoSource from an input file
CREATE VideoSource traffic_a
    FROM FILE '/home/xzy/video-data/a.mp4';
```

In other words, `VideoSource` serves as the basic data structure in DoveDB to support query processing. It is a collection of videos that can be processed in parallel. The elements of `VideoSource` are processed from the first frame to the subsequent frames along the time axis through RDD transformation. This procedure is typically accompanied by calculation or memory persistence, which enables model inference, indexing and storage on each frame.

2.2 Query Language Syntax

Several video database systems have extended SQL to VQL and provided declarative language for video data management and query processing. An example of selection query is illustrated in the following.

```
SELECT frame FROM source_name
    WHERE CONTAINS(label, 'CAR')
        AND confidence > 0.8
    ORDER BY timestamp DESC LIMIT 10;
```

In this paper, we devise a more expressive language called VMQL, which augments existing VQL with functionalities for model-oriented management and convenient deployment. For instances, to train an object detection model for an out-of-vocabulary object type, we can use the following statement

```
CREATE YOLO model_name
    ON DATA_TABLE(img_column, label_column);
```

Here, we assume the training data have been annotated and stored in the database. The trained model is named and stored in our model corpus. It can be conceived as a user-defined function and applied on a `VideoSource` for visual inference:

```
CREATE MONITOR_EVENT event_name
    USING model_name
    ON DATA_SOURCE source_name;
```

The above example shows how to create a monitoring event on a target `VideoSource` using our trained model.

2.3 Real-time Ingestion

The advantages of OTIF [3] essentially come from its multi-object tracking (MOT) model to extract the tracklets of all the moving objects and index them to support selection and aggregation queries. In DoveDB, we devise a more advanced MOT model that works well with low sampling rate. In other words, DoveDB can achieve the same query accuracy level with OTIF by ingesting much fewer number of video frames. In our implementation, we adopt the popular tracking-by-detection paradigm in the family of existing MOT models. It first applies an existing detection model (e.g., YOLOX) on the sampled frames to generate bounding boxes for the detected objects. The second step is to associate those bounding boxes to recover the tracklets of moving objects. A common strategy is to apply Kalman filter to predict the future position of a bounding box in the next frame. As such, a detected object is assigned to an existing tracklet if its spatial matching distance is small. In DoveDB, we devise a sampling-resilient MOT model, with accurate motion prediction and robust bounding box association. To predict the position of an object in the next sampled frame more accurately, we propose an enhanced Kalman filter with more informative state

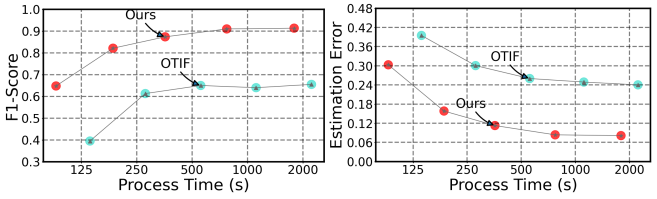
representation and parameter matrix update mechanism. The state representation is augmented with acceleration speed to capture non-linear motion pattern. The estimated and process covariance matrix can be dynamically updated according to the divergence between observation and internal state prediction. In addition, to associate the detected bounding boxes more robustly, we propose a comprehensive similarity metrics that integrates multiple spatial matching clues, including overlap, center point distance and aspect ratio of the bounding boxes. For more details of sampling-resilient MOT, readers can refer to our technical report[1], where we also compare our approach with many recent MOT models.

In the following, we briefly present the experimental comparison between DoveDB and OTIF using GeForce RTX 3090 Ti as the hardware environment and Jackson Town as the test dataset. In Table 1, we vary the sampling ratio and report MOTA and IDF1, which are two popular performance metrics to measure tracking accuracy. When we use lower sampling ratio, the inference time can be significantly reduced but the accuracy of tracking also drops. Under the same ratio, DoveDB is much superior over OTIF in terms of both efficiency and accuracy.

Table 1: Tracking performance of DoveDB and OTIF.

	Sampling Ratio=1/8			Sampling Ratio=1/16			Sampling Ratio=1/32		
	MOTA	IDF1	Time	MOTA	IDF1	Time	MOTA	IDF1	Time
OTIF	55.5	69.6	1113.5s	51.5	62.3	556.7s	42.3	53.1	278.3s
DoveDB	**74.6**	**74.7**	**752.1s**	**70.8**	**67.4**	**353.7s**	**65.2**	**59.9**	**181.2s**

As mentioned, we can leverage the results from MOT models to answer selection and aggregation queries. In Figure 2, we compare DoveDB with OTIF in terms of the trade-off between efficiency and query accuracy, by varying sampling ratios. The selection query retrieves all the video frames containing at least one vehicle. The aggregation query estimates the number of vehicles that appear in the video clip. We use F1-score and Mean Average Error to measure the quality of these two queries, respectively. With the same video ingestion time, our DoveDB achieves significantly more accurate retrieval performance than OTIF. Due to space limit, more details are presented in section 4.9 of our technical report.



(a) Selection Query (b) Aggregation Query

Figure 2: Query processing performance of DoveDB and OTIF.

2.4 Offline Index Construction

For each frame of video, we employ visual models(i.e., YOLOX in DoveDB) to extract a set of spatio-temporal meta-data, including bounding boxes outlining detected objects, as well as their corresponding visual features. Additionally, we capture meta-data such as camera locations, frame timestamps, and the precise positions of identified bounding boxes. We use OpenGauss[2] to store this data and create the necessary indexes for them to facilitate a wide range of advanced queries, such as interesting location-aware queries or mining tasks.

2.5 Visual Model Management

In DoveDB, a visual model is conceived as a user-defined function to perform a specified inference task, such as object detection, image classification, image segmentation, etc. DoveDB is integrated with built-in functions to train a new model from scratch or fine-tune an existing visual model with additional annotation data. In order to support model training and inference, the system integrates PyTorch Lighting as the deep learning framework and exposes batch data feeding API to support other machine learning frameworks.

2.6 Query Processing Engine

With the textual, visual, and spatio-temporal indexes built from the extracted tracklets, online queries can be processed with very low latency. For instance, to retrieve video frames containing ambulance and firetruck, we can simply perform an intersection between the inverted lists of these two labels. As another example, to estimate the traffic flow with a time period, we can leverage the spatial-temporal index to identify the relevant video frames and aggregate the number of distinct objects within these frames. If a query cannot be answered via the offline index and requires online inference, we also provide model acceleration techniques, including network compilation and quantization, to reduce computational complexity and boost inference speed. Finally, our real-time ingestion scheme can be naturally used to support continuous object tracking queries.

3 DEMONSTRATION SCENARIOS

3.1 User Interface

In Figure 3, we show the user interface of DoveDB for declarative video query processing. The layout contains five main components, marked as zone A, B, C, D, and E, respectively. In zone A, users are allowed to select a target camera to view its streaming video frames through the drop-down box. If the camera is deployed with monitoring events or continuous queries, the output (e.g., the detected bounding boxes for the multi-object tracking event) will be displayed as well. Zone B contains the visual models that have been trained and stored in the database, such as object detection model YOLOX and our proposed multi-object tracking models. Zone C is a map panel, deployed with 120 surveillance cameras for the demonstration purpose. The videos are generated using Carla simulator[3], which has also been used in Visual Road [6] to generate benchmark datasets for video database. Zone D is a result panel for selection and top-k queries to show the list of retrieved video frames. Zone E is a command-line panel to accept user's VMQL input.

3.2 User Interaction Scenarios

Scenario 1: Model training and deployment. In this scenario, we illustrate the utilities of model training and deployment with VMQL. Specifically, we train a lightweight model to detect red vehicles and

[1] https://github.com/dovedb/DoveDB/blob/main/SR-Track.pdf

[2] https://opengauss.org/

[3] https://carla.org/

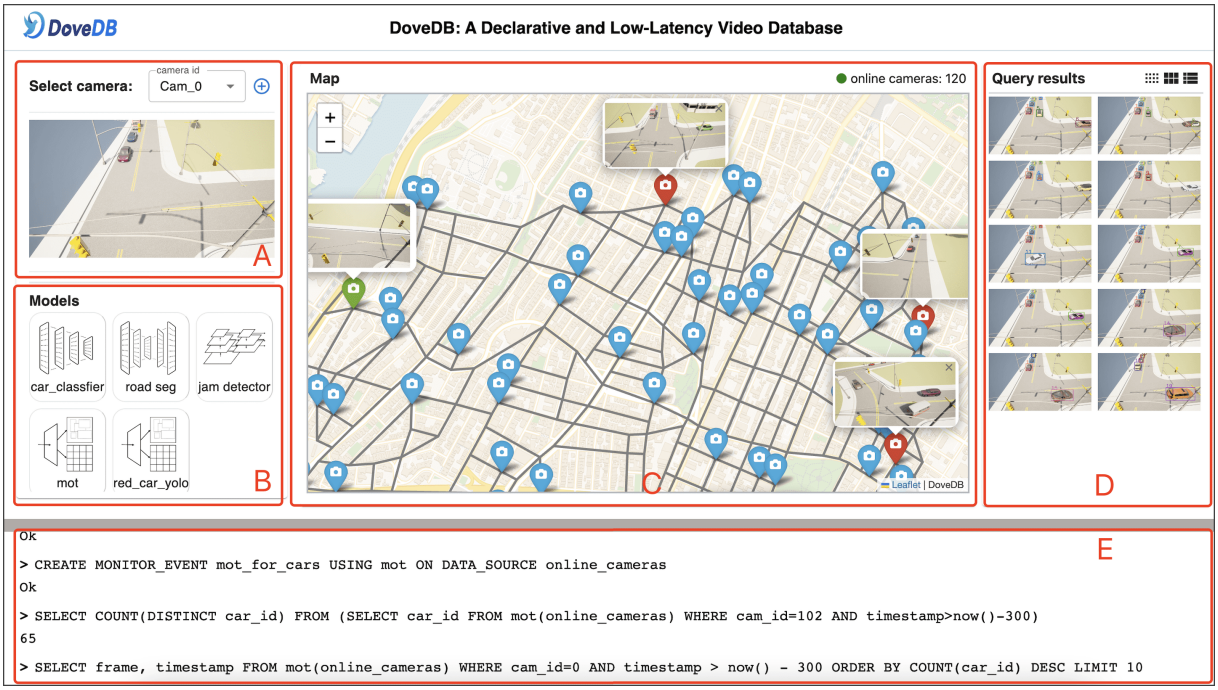


Figure 3: User interface of DoveDB.

deploy it on 30 cameras for real-time object detection on streaming data. To train the model, we first harvest training labels without human intervention, i.e., we use expensive object detection models to generate pseudo labels. Among the detection results, we filter the objects whose class labels are not related to vehicle and leverage prior knowledge on RGB color space to retain red vehicles. Afterwards, we train a lightweight object detector using the network structure of *YOLO nano* whose model size is only around 4.0MB. To accelerate training speed, we also adopt the mixed-precision training scheme provided by NVIDIA. When the training is finished, the model specifically designed for red vehicle detection will be added to the model panel in Zone B. Using the following VMQL, we deploy it on 30 cameras for real-time detection of red vehicles. In the demonstration, once a target vehicle is detected, a dialog box is popped from the camera to show the related video frame.

```
CREATE MONITOR_EVENT red_car_event
    USING red_car_detector
    ON DATA_SOURCE  online_cameras WHERE cam_id < 30;
```

Scenario 2: Multi-object tracking and index construction. This scenario illustrates the efficiency of multi-object tracking, which can be viewed as a continuous query on video streams. Similar to the above VMQL statement, users can create a monitoring event to deploy our sampling-resilient MOT model on multiple target cameras. After the model has been deployed, users can click the camera icons on the map plane to find the real-time tracking output. In addition, the detected results will be collected to build indexes and support queries that will be demonstrated in Scenario 3.

Scenario 3: Aggregation and top-k queries. This scenario illustrates the low latency of query processing. As illustrated in the panel of command line in Figure 3, we use aggregation and top-k queries for demonstration. The aggregation query obtains the number of vehicles in a target camera within the specified time period. The top-k query retrieves the most dense video frames and the result frames will be displayed in Zone D. During the demonstration, since we have built offline indexes for the target camera, these two queries can be answered instantly.

4 ACKNOWLEDGMENTS

This work is sponsored by CCF-Huawei Populus Grove Fund.

REFERENCES

[1] Michael R. Anderson, Michael J. Cafarella, Germán Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *ICDE*. IEEE, 1466–1477.
[2] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael J. Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *SIGMOD*. ACM, 1907–1921.
[3] Favyen Bastani and Sam Madden. 2022. OTIF: Efficient Tracker Pre-processing over Large Video Datasets. *SIGMOD* (2022).
[4] Favyen Bastani, Oscar R. Moll, and Samuel Madden. 2020. Vaas: Video Analytics At Scale. *Proc. VLDB Endow.* 13, 12 (2020), 2877–2880.
[5] Daren Chao, Nick Koudas, and Ioannis Xarchakos. 2020. SVQ++: Querying for Object Interactions in Video Streams. In *SIGMOD*. ACM, 2769–2772.
[6] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2019. Visual Road: A Video Data Management Benchmark. In *SIGMOD*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 972–987.
[7] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Deep CNN-Based Queries over Video Streams at Scale. *Proc. VLDB Endow.* 10, 11 (2017), 1586–1597.
[8] Piyush Yadav, Dhaval Salwala, Felipe Arruda Pontes, Praneet Dhingra, and Edward Curry. 2021. Query-Driven Video Event Processing for the Internet of Multimedia Things. *Proc. VLDB Endow.* 14, 12 (2021), 2847–2850.