Challenges and Opportunities for Autonomous Vehicle Query Systems

Fiodar Kazhamiaka*, Matei Zaharia, and Peter Bailis

Stanford University

ABSTRACT

Autonomous vehicles (AVs) collect and process rich visual and dimensional data as they operate. Across a fleet of AVs, the collective data forms an up-to-date partial snapshot of the physical world. The ability to query this data has many use-cases, from finding open parking spots and checking the line outside a coffee shop, to monitoring the usage of urban spaces and road conditions. Managing AV data has unique challenges with respect to data volume, bias, and privacy issues. In this paper, we take steps to describe the design and research opportunities for a new class of data management systems: AV query systems.

1 Introduction

Several generations of sensing devices – from smartphones to UAVs and inexpensive satellites – have unlocked new opportunities for querying the world with data. Smartphones revolutionized the mobile sensing landscape with their ubiquity, array of sensors, and processing power. However, despite their pervasiveness, smartphone sensor data provides limited insight into the state of the surrounding world. Camera and microphone sensors, which could theoretically be leveraged to provide observations of the local environment, are kept off per the user's expectation of privacy. Instead, visual sensor networks in the form of static camera networks, satellites, and camera-mounted drones are deployed to let users query a wide variety of physical phenomena, such as monitoring traffic [6] and store parking lots [39], streaming wave cameras for surfing [15], and detecting events such as oil spills [14].

However, there are many more examples of useful queries about the physical world that cannot be answered by existing approaches. This is especially true for queries requiring high-resolution sensing, wide coverage, and real-time response. A commuter might want to know how many people are currently in line at every coffee shop on their way to the office; road-way patrols want to monitor road conditions in real-time; a city planner could ask how a particular public space is used, or what kind of vehicles are using a particular road segment over the past year; delivery truck drivers want to know about open parking spots closest to their destination; investors look for creative "alternative data" queries to gain insight into investment opportunities.

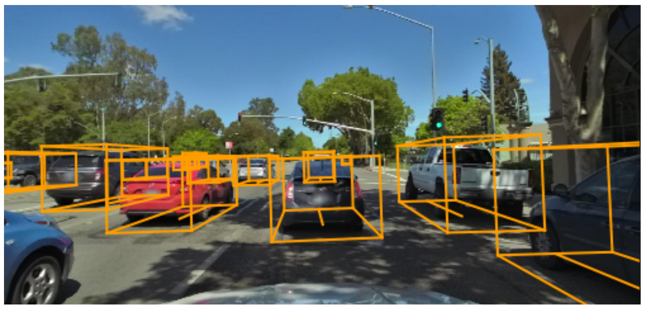


Figure 1: AV camera image with vehicle bounding-boxes [40]

We believe that autonomous vehicles (AVs), with their array of visual and laser sensors, powerful onboard computing hardware, and mobile coverage, will bring about the next revolution in both the quantity and quality of data obtained on the outside world by leveraging their hardware to form a mobile sensory and computing network. This is made possible by the ongoing proliferation of sweep sensors on cars, and by advances in computer vision algorithms for physical object detection through video and other sensors; crucially, these algorithms are already deployed as part of the car's driving task.

Contemporary cars with self-driving systems, such as Tesla's Autopilot [16] and Cadillac's Super Cruise [8], capture high-resolution visual and structural data of the world they navigate. This data is processed to mark location, class, and velocity of objects such as pedestrians and vehicles, as in Figure 1. Consider the data collected and processed by a fleet of several hundred vehicles driving through a city; at any moment in time, the combined data forms a partial present-state snapshot of that city. This data can be processed on-board, and/or transferred to one of a network of local data warehouses that answer queries, as shown in Figure 2. In its most ambitious form, data will be collected across millions of cars from around the world, providing unprecedented coverage and detail of physical space.

In this paper, we argue that realizing the opportunity for large-scale analysis and storage of AV fleet data will require a new class of data management systems: AV query systems. AV query systems must tackle several challenges that differentiate them from existing data management systems:

*Correponding Author: fiodar@stanford.edu

Data volume. AV sensors generate terabytes of data per hour. A fleet of AVs generates several orders of magnitude more data than any existing widely-deployed sensor network. This volume is both a blessing and a curse. The data provides a detailed view of the physical world surrounding the AV; however, it is impractical to upload and store in entirety across a large fleet of AVs. Classic techniques such as sampling every n^{th} frame and inter-frame compression algorithms are insufficient, and the vast majority of sensor data will have to be discarded. This leads to several research questions: *How should the system balance data storage and transmission costs with the querying power that is lost when data is discarded? How can the system be designed to support different query modes, for example, real-time queries, historical queries, and queries on arbitrary data?*

Sampling bias. Unlike most sweep-sensor networks, AVs are mobile, and the collected data is tied to when and where their passengers need to travel. This results in a sampling bias that affects queries on statistics related to transient events, for example, a query to identify the road segment with the heaviest bike traffic within a city. *How can AV data be de-biased to provide accurate query answers?*

Privacy. The data collected across a fleet of AVs can expose both AV owners and observed individuals to privacy intrusions. If not handled carefully, the data may be used for invasive purposes such as tracking individuals. *How can AV data be made queryable while preserving the privacy of the car owner and observed persons?*

In the rest of the paper, we describe our vision for AV query systems, discuss several emergent technical research problems, and introduce a proposed system architecture (Figure 2). We present empirical evidence from existing AV datasets, propose initial directions for several of these problems, and discuss design trade-offs meriting further investigation by the data management community.

2 Motivating Queries

Queries about the physical world are typically *spatio-temporal*, meaning that a target geographical location and time are specified or are part of the result. The queried subject is an object such as a vehicle, a geographical location, a time range, or a set of these. Below, we list four examples of queries that an AV query system could answer.

Q1: Where are the 3 closest currently-open parking spots to a particular location? This query can come from anybody looking to park their car, and is especially useful as a tool for "last-mile" logistics, where parking a delivery vehicle is a major hindrance to efficient delivery [2]. Camera networks and satellites cannot answer this query for arbitrary locations; both systems are limited by their coverage, and satellite imagery is not updated frequently enough to provide useful results. AV data can identify locations where cars are often parked, and the recent observations of each spot reveal their availability. Alternatively, if the query is performed ad-hoc (e.g., by an app that finds parking for its users) it can be executed by directly querying the status of known parking spots.

Q2: Which locations currently contain debris on the road? This query is useful for roadway patrols, who currently rely on drivers that can safely phone in to report debris [3]. AVs can detect debris and upload its location and image, and a stream query allows for prompt response and

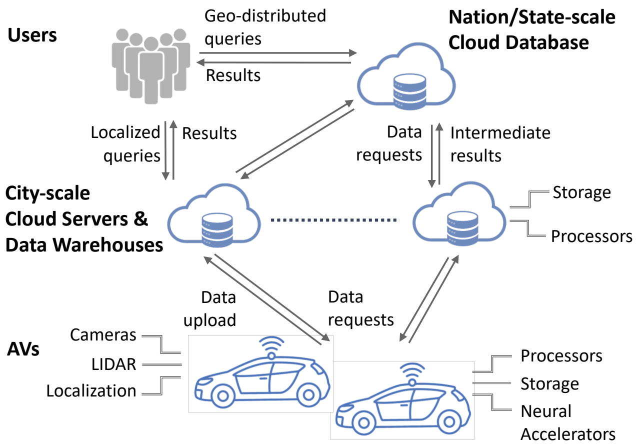


Figure 2: AV query system architecture

removal. Existing sensor systems, such as camera networks and satellites, are unable to monitor the road with enough coverage and detail to detect debris.

Q3: What percentage of the time between 8 am and 8 pm is there a cyclist on the road near a particular intersection over the past year? This query could be asked by urban planners who are looking for data on urban space usage and traffic calming measures – in this case targeting streets with high cyclist traffic for a new development of cyclist-friendly lanes. This query requires observations made of the target intersection over the past year. Existing approaches to answering such queries involve physically monitoring the road, or setting up cameras to monitor specific parts of the road, which is tedious and delays the collection of statistics.

Q4: How many people are currently in line at a store's locations? This query may be asked by the general public to check on wait-times, or by stock traders as a proxy for estimating sales before they occur. The query could be formally specified by first identifying the areas in which people might line up around every store that is visible from a road (e.g., Tartine bakery in San Francisco), and counting the number of stationary people observed within; more complex execution strategies can cross-correlate lines observed across similar stores. While smartphone location data could be used to estimate this query, many people keep location tracking off by default on their personal device.

In summary, AV query systems have the promise to answer these queries using data that is collected and processed as a byproduct of AV driving, which allows results to be obtained efficiently and at scale compared to existing approaches. Other queries may require additional processing, for example classifying a vehicle's make and model to estimate demographics [30]. In such cases, the accuracy of downstream classification models can be improved by using data from multiple sensors, such as camera and LIDAR [42]. The key challenges lie in the efficient, accurate, and privacy-preserving execution of these AV queries.

3 Challenge: Data Volume

Data volume represents a major challenge in AV query systems. Each AV is expected to generate 3-6 TB of raw data per hour of operation[1] depending on the sensors installed. We focus primarily on visual data in the form of video frames collected by upwards of six high-resolution (2K) and high-frequency (at least 30fps) cameras, and comprises the majority of this volume. In this section, we discuss the problem of managing this data, the implications on the queries that can be supported by the system, and how data volume can be reduced.

3.1 Keeping the Data in Entirety is Impractical

Consider an AV that drives an hour a day generating data at a rate of 5 TB/hour, and assume that the data is downsampled from 30 Hz to 2 Hz and compressed by a factor of 10×, down to 33 GB/hour. Uploading this volume of data is impractical across a fleet of vehicles; 5G networks are expected to support up to 10 Gbps (4.5 TB/hour) rates in ideal conditions, but in practice, speeds are much lower [41]. Even if transmission bandwidth is not a concern, the cost of uploading data at \$0.05/GB [5] is \$12.50/hour, and storing an hour of data in Amazon Glacier at \$0.004/GB per month [7] costs \$1.6 over a year. The resulting yearly data volume is 12 TB, which costs \$1200 to upload and store for a year, and presents a significant cost burden when scaled to hundreds of thousands of vehicles. This problem requires algorithms that reduce the raw data volume by an order of $10^4\times$, which will impact both the cost and querying power of the system.

There are many approaches to reducing data volume. In the following, we discuss how data can be reduced to support specific AV query modes, content-driven data selection, and efficient data formats.

3.2 Query Modes

Ideally, only the data that will be used to answer queries should be stored; this context-aware data reduction would minimize transmission and storage costs without sacrificing querying power. To explore this idea further, we discuss different query modes that can be supported by the system: real-time, historical, pre-specified, and arbitrary queries, and their implications on data volume.

Query scope: Historical versus Real-Time. Queries can pertain to present (real-time) or past (historical) states of the physical world. For real-time queries, only the required data needs to be uploaded. Data can be discarded after the query result is obtained to avoid the accumulation of storage costs over time. Queries on the present state often require low latency, implying that AVs must upload their observations in real-time over a potentially expensive channel. The latency of a real-time query is most affected by how long it takes for AVs to pass by the target locations, and on processing time; contemporary computer vision models process an image in tens of milliseconds on a server-side GPU [17] and sub-second on mobile phone-level hardware [37]. Historical queries require data to be collected and stored. The relevant data be uploaded opportunistically, for example over WiFi + fiber rather than a cellular data connection. However, AV on-board storage is limited, and data should be off-loaded when possible to avoid being over-written.

[1]For example, consider a 30 fps 2K video camera, where each frame consists of 2048×1080 pixels across 3 channels with 16-bit precision, for a total of 12.65 MB per frame, or ≈ 1.3 TB/h per camera.

Table 1: Pedestrian and Vehicle Observations in AV Data

Dataset (Camera FPS)	NuScenes (2)		Lyft (5)
Location	Singapore	Boston	Palo Alto
Images Analyzed	92,184	112,710	136,080
Instance Annotations	261,756	623,264	637,993
Avg. Instances/Image	2.84	5.53	4.69
Unique Instances	14,567	32,791	17,190
Avg. Unique Instances	1.89/sec.	3.49/sec.	3.82/sec.
Sampled images, Alg 1	5,432	9,754	12,052

Expressivity: Pre-specified versus Ad-hoc. The system can mandate that queries must be submitted before any relevant data is collected. To support pre-specified queries, only the data required by each query needs to be collected. Any real-time query is inherently pre-specified, while queries on past observations, such as Q3, would have to be specified "over the next year" and then wait while data is gathered. Ad-hoc queries, including those on past observations, are the most challenging to support. They require the collection of all data that might be touched by a query, which is impractical given the volume of AV data and the diversity of possible queries. A best-effort approach is needed to filter out data that is less likely to be useful. In particular, data featuring dynamic objects, such as pedestrians, vehicles, and animals could be collected more often than data featuring objects whose state changes very slowly, such as building facades.

3.3 Content-Driven Data Selection

Several promising approaches for reducing data volume are summarized by the following two principles: 1) Additional observations of an object add a diminishing amount of new information, 2) An object's state can often be represented in a low-volume data format.

Intuitively, static objects will not change much across consecutive observations, and additional image frames are often redundant and may be discarded without loss of information. For example, a parked car may require only a single image along with a set of annotations marking every time that particular car is observed to be in the same parking spot. For dynamic objects such as pedestrians, a passing AV collects and processes video and LIDAR data to estimate their velocity and location. The velocity and location of the pedestrian can be described using several bytes of data, and captures much of their state without requiring the entire video clip encoded in millions of bytes. Ideally, the image corresponding to each object is chosen in a way that supports further object classification. After reduction, the visual data uploaded by an AV over the course of a trip no longer resembles a smooth video stream, but rather an album of images featuring observed dynamic objects.

3.3.1 Empirical Study of AV Data Content

To better understand the information density of AV data, we analyze the dynamic objects observed in real driving scenes from the NuScenes [24] and Lyft Perception [40] datasets, which sample AV sensors at 2 Hz and 5 Hz, respectively. We focus on the visual data samples, which are composed of 6 images coming from 6 cameras that provide 360° visual coverage. The data is organized in 1080 separate 20-25 second-long scenes located in Boston, MA, Palo Alto, CA, and Singapore, and is indicative of city driving scenarios. Each sample is annotated with the location of objects such

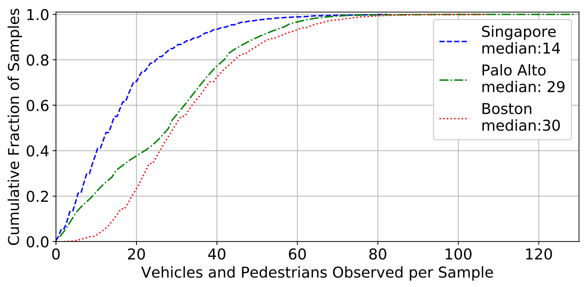


Figure 3: Distribution of observed vehicles and pedestrians.

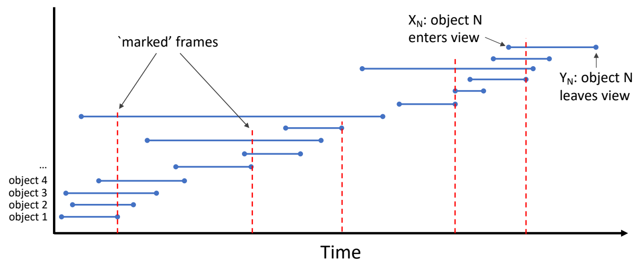


Figure 4: Visualizing objects entering and leaving an AV's view. Dashed lines represent frames chosen by the *min-frame* algorithm.

as pedestrians, vehicles, and various driving-related obstacles and signage, and object instances are tracked across samples.

Table 1 shows the observation statistics for dynamic objects, specifically pedestrians and vehicles. Each image contains an average of 4-5 objects, and the rate of objects that come into view is roughly 2-4 per second. If a single unique image for each observed pedestrian and vehicle is kept, then the total reduction in data volume, assuming 6 cameras recording at 30 fps, is between 47-95× before any inter-frame image compression is applied.

The distribution of dynamic object density across all samples is shown in Figure 3; at least 50% of samples have 14 or more instance observations across 6 cameras, the exact number varying with location. Capturing multiple objects within a single sample can reduce the number of samples needed to capture all objects.

3.3.2 Algorithms for Frame Selection

It is helpful to first consider a simpler form of the problem. Sample selection can be framed as an interval coverage problem: given a set of one-dimensional intervals, representing the times at which object enters and leaves an AV's view, we seek a minimal set of points (samples) that covers all intervals. Figure 4 visualizes this in one dimension. A minimal set can be obtained using Algorithm 1, where intervals are 'marked' whenever a point in the interval is added to the set; points are considered in ascending order, and are added when a hitherto unmarked interval ends. The proof of optimality for this algorithm is provided in the appendix.

The last row of Table 1 records the number of images needed to capture all unique vehicle and pedestrian instances in each dataset using the *min-frame* algorithm. The sampling rate, compared to an initial rate of 180 images per second (30fps, 6 cameras) drops to 0.7 for Singapore, 1 for Boston, and 2.7 for Palo Alto, a 67-257× reduction. Coupled with a 10:1 intra-frame compression ratio, the volume is reduced by 670-2570× to 1.9-7.5 GB/h, or \$69-\$274 in storage and transmission costs per year for the example given in Section 3.1.

Algorithm 1: MIN-FRAME

```
Result: Set of points $S$ that covers all intervals
$N$ intervals $[x_i, y_i]$, sorted by finish time $y$
$S, M \leftarrow \{\}$
for $i \leftarrow 1$ to $N$ do
    if $i \notin M$ then
        $S \leftarrow S \cup y_i$
        $M \leftarrow M \cup \{j : x_j \leq y_i\}$
    end
end
```

However, this approach is a simplification of the setting. In reality, the quality of a sample with respect to an object changes over its interval of observation. For example, a sample that captures several objects at a short distance without any obstructions is more useful for downstream processing than a sample where those same objects are distant and/or obstructed. Balancing sample quality with sample volume is an interesting optimization problem that warrants further study.

Scenarios where multiple AVs simultaneously observe the same object can lead to redundancies in the union of their collected data. If the data is being uploaded to a central database, then redundant transmissions can be avoided through a simple protocol that, before sending expensive image frames, checks for existing observations with matching features and location. We are not aware of existing datasets featuring this scenario, and are currently developing one using Carla [28], an AV driving simulator. This effort is similar in spirit to the Visual Road benchmark [32] for static camera data that is built using the same simulator.

3.4 Efficient Data Formats

Selective image compression can be used to reduce the size of image segments that hold less information [10, 20]. For example, the image in Figure 1 can be compressed with lossy algorithms in sections outside of object bounding boxes, such as the road surface, trees, and sky. However, high-resolution images are important for object detection and classification accuracy [57], and backgrounds can provide important context for some queries (eg. on the presence of dangerous black ice on road surfaces.) Hence, it's preferable to explore data formats where these features are preserved.

Embedding networks provide opportunities to extract features from high-resolution data, and offer an alternative, lower-volume data representation. An embedding network can be used to convert an image into a condensed set of image features, which can be processed during query time by a model trained on those features. The challenge in this approach lies in creating an embedding that captures the large state space of AV data. Recent work in this space includes spatio-temporal scene tagging [51] using a "universal" embedding network trained alongside a large set of attribute embeddings. This work presumes an exhaustive list of scene attributes that might be queried. Extending universal embeddings with additional attributes that weren't initially considered in order to support queries on ad-hoc attributes is an interesting research direction.

3.5 Summary of Data Volume Challenge

In summary, AVs produce data in volumes that can't be stored or transmitted at scale. and we discussed three ways in which this data can be reduced while minimizing information loss. First, data can be filtered to match specific classes of queries, especially when queries are assumed to be pre-specified. Second, interesting objects and events are not evenly distributed across frames, and algorithms for balancing frame quality and data volume are needed. Third, alternative formats, such as embeddings, could store image features rather than pixels, and the creation of a universal and extensible embedding to capture the large AV data state space is a promising direction.

4 Challenge: Data Bias

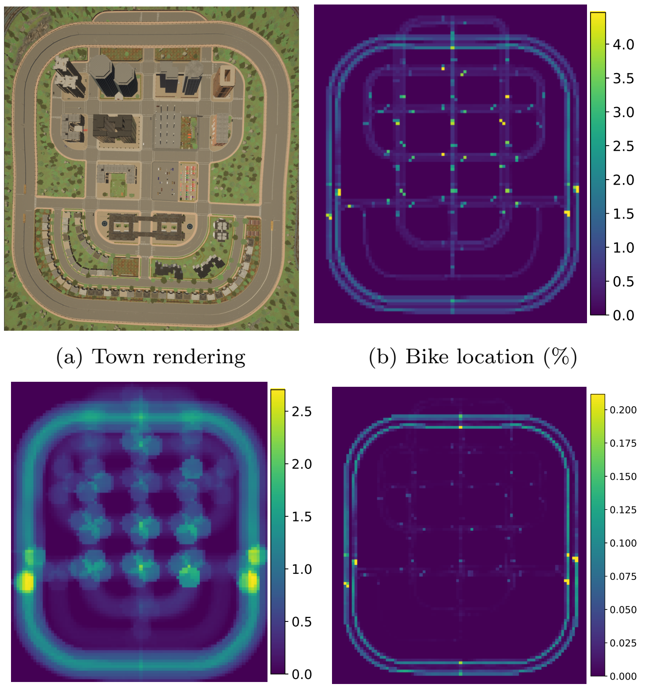
AVs collect data opportunistically as a byproduct of their transportation task. Due to the spatial and temporal patterns that naturally occur in traffic, the data has both spatial and temporal bias. This bias is a problem for queries about stochastic phenomena. For example, consider Q3 from Section 2 on the frequency of observing cyclists in a particular intersection during the daytime. AVs are more likely to drive through the intersection during commuting hours, which may be correlated with the presence of cyclists. Naively averaging the number of cyclists seen across the collected frames would give an incorrect result when the frames are saved only if they contain something of interest.

In this section, we discuss how to account for biased data using importance sampling, how to estimate the sampling distribution, and the caveats of applying importance sampling to AV data.

4.1 Importance Sampling

Importance sampling is a general technique for estimating the properties of a distribution (eg. bicycles in the intersection) using samples generated from a different distribution (eg. AV traffic patterns). At a high level, importance sampling is the use of the sampling distribution to adjust an observed statistic. For example, to execute Q3, the number of bikes observed in a particular location should be normalized by the frequency with which the location was observed. As both observations and bike frequency depend on the time of day, adjustment should be done over short time intervals and then aggregated to produce the final result. However, importance sampling requires knowledge of the sampling distribution; for Q3, this equates to tracking AV observations of the intersection. Restated, it is not enough to track only bike observations; all observations of intersection must be known. This is problematic for data from mobile sweep sensors in an environment with many line-of-sight obstacles and where the vast majority of data is discarded.

To demonstrate the error of an incorrect sampling distribution in a realistic setting, we ran an experiment in Carla where an AV drives around a small town and tracks dynamic objects (cars, bicycles, and pedestrians) encountered within a 20-meter radius over 48 hours. The AV keeps video frames if they contained a dynamic object. Some of the data generated from this experiment is summarized in Figure 5. We then queried every 5×5-meter cell containing a roadway, scaling the frequency of bike observations (Figure 5d) by the inverse of the frequency with which each location is captured in a saved camera frame. The results grossly overestimated the presence of bicycles in the majority of locations. This is expected, as the sampling distribution reflects observations only in the presence of dynamic objects, which naturally skews the query result to over-estimating the likelihood of object presence.



(a) Town rendering (b) Bike location (%)

(c) AV location observations (%) (d) AV bike observations (%)

Figure 5: Carla simulation with one AV and 17 bikes over 48 hours. 5b shows the percentage of time a bike is within each 5×5m cell, 5c shows the percentage of time each location is observed by an AV, and 5d shows how often a bike is observed by an AV.

4.2 Estimating the Sampling Distribution

To serve arbitrary location queries, each AV must upload an *observation record* describing the time and location of all observations. We propose a segmentation abstraction for tracking observations, where location is discretized for tractability. Observed segments are identified through GPS, LIDAR, and supporting localization sensors. Figure 7 illustrates this concept in 2D; a LIDAR sweep, shown as a point cloud, is superimposed onto a color-coded grid that segments the physical world and indicates which locations are observed. Using a 3D discretization allows for physical locations to be queryable in three dimensions, although many use-cases can be satisfied with 2D. Each segment corresponds to an observation record consisting of three items: an ID that uniquely identifies the segment, and two timestamps marking when the segment was in view. Returning to Q3, the sampling distribution is characterized by the records of segments composing the target intersection.

This approach is not perfect, as segments may be observed without LIDAR detecting them. For example, many concrete freeway dividers block sight of the roadway, but vehicles are still visible. In this case, the sampling distribution of the road is skewed to observations involving vehicles. Furthermore,

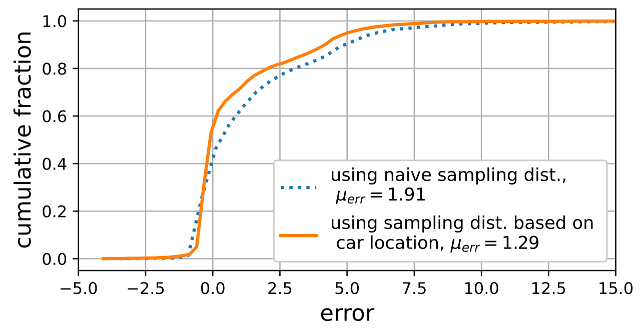
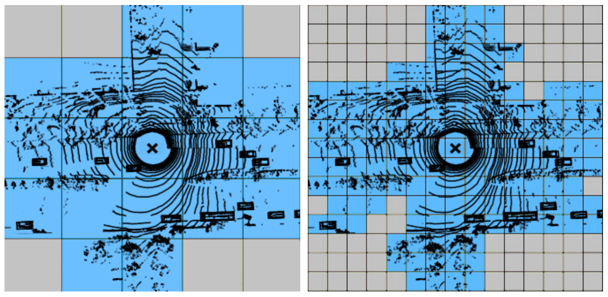


Figure 6: Distribution of the magnitude of error in estimating the percentage of time each road segment contains a bike. Using an accurate sampling distribution reduces the error, although it does not fully eliminate it.



(a) 15m × 15m segments (b) 5m × 5m segments

Figure 7: Aerial view of a LIDAR point cloud, from [40]. "X" marks the location of the LIDAR sensor, and rectangles mark the location of observed vehicles. Smaller segments offer higher specificity of which areas are observed (blue), providing a more accurate sampling distribution estimate for each location.

the right granularity of the segmentation is not obvious and presents a trade-off between query accuracy and data management. Smaller segments allow the system to query very specific locations at the cost of having to manage many records, while large segments lose specificity but reduce the overhead of managing records. For example, in a city the size of San Francisco (120 km^2) with 25% visibility from a roadway, a 5m×5m segmentation as in Figure 7b creates 144 million segments. Segment sizes do not need to be identical across all physical locations; there may be a greater benefit to smaller segments in information-dense urban landscapes, and larger ones along rural roads and expressways. A technique for context-aware segmentation that considers both overhead and sampling estimation error is needed here.

4.3 Where Importance Sampling Can Fail

In AV data, there is no guarantee that observed phenomena are independent of the sampling distribution. Consider an AV that is always chased by a dog as it drives by a particular house; the data would suggest that a dog can always be found on the stretch of road near the house, though in reality this occurs only in the presence of AVs. Covariance between sampling distribution and observations can skew query results.

The example with the dog is not as pathological as one might expect. A moderate form of this phenomenon occurs through traffic lights, which synchronize traffic patterns. The longer an AV is stopped, the more likely it is to observe other cars, cyclists, and pedestrians that join it at the intersection, and that would otherwise have been missed if the traffic light was green. It then continues to drive among the group of vehicles that built up while the light was red, resulting in objects being present in the data with a higher frequency than they occur. We have observed the relationship between sampling and object presence to be common in our initial experiments in Carla. When we use importance sampling with the better sampling distribution obtained using the technique described earlier, the errors are lower but still heavily skew towards over-estimating Q3 (Figure 6. This can be partially attributed to the synchronization of AV and bike traffic. How to adjust for this behavior is an open question.

4.4 Summary of Bias Challenges

In summary, we discuss two challenges to dealing with AV data bias. First, tracking observed locations to estimate the sampling distribution allows us to use importance sampling, with opportunities to reduce the overhead in managing this data over large areas and at high resolution. Second, the distributions of samples and of observed phenomenon may not be independent and can skew query results.

5 Challenge: Privacy

Privacy is of primary importance for AV query systems. AV data can be used to infringe on personal privacy rights of the observed as well as the observers (AV owners). Most people experience "overwhelming discomfort" at the idea AV data being used to identify and track individuals [22]; on the other hand, most reported feeling comfortable sharing anonymized data collected by their car [26] [46], especially if benefiting from the features it enables. We discuss privacy implications using Google's StreetView as a case study, and how privacy concerns may differ depending on the use-case.

5.1 StreetView Case-Study

StreetView, which launched in 2007, challenged pre-existing notions of privacy. A prevalent concern is public access to photos of individuals who may be engaging in activities that they would not wish to be published online [49]. As a result, some countries are sparsely covered by StreetView outside of major cities [4], while others prohibit it entirely over security concerns [1].

AV query systems exacerbate the issues faced by StreetView. For example, StreetView blurs faces and license plates in their images to preserve privacy. In an AV query system, there may be multiple images of an individual over time. It takes as little as four spatiotemporal data-points to distinguish a particular individual from over a million others with high accuracy [27]; blurring the face or license plate does little to preserve the privacy of the individual, who could be tracked across multiple observations by their clothing or other features. Advanced forms of data redaction, such as silhouette redaction via image segmentation [45], are needed.

The owner of an AV that uploads data is also susceptible to privacy violations. While this could be solved by removing any associations of the observing AV from their data, it poses a problem for systems which compensate AVs in proportion to the value realized through their data. In this case, the problem could be ameliorated by periodically refreshing an AV's identifying tag, as done in contact tracing apps [25]. However, images can sometimes be attributed to a specific camera by analyzing digital camera noise "fingerprints" [29]. Approaches to mitigate these issues, for example by introducing obfuscating noise, need to be explored.

5.2 Privacy by Use-Case

Different use-cases of AV-query systems have varied notions of privacy. In some cases, a car owner may wish to opt into sharing sensitive data. For example, consider a child abduction alert that is broadcast to every AV with information on the make, model, and license plate number of the suspected kidnapper's vehicle. The user could be prompted to temporarily allow for the detection of the specific car, and flag its location to the police. This capability would require judicious use and oversight, and should be designed to minimize privacy invasions. For example, rather turning off redaction for all image uploads, the car could download and run a light-weight model designed to detect the target vehicle.

6 Implications for System Design

To illustrate the design space for an AV query system, we present a system architecture for physical components and data flow in Figure 2. Users represent individuals sending queries through a web portal, or applications that send queries to the servers through an API. AVs and cloud servers are entities within the system that are equipped with computing hardware and storage, and data flows between them. Submitted queries are answered using data from AV sensor, including cameras, LIDAR, RADAR, and sensors that help with localization. AVs can be made aware of queries, and send their observations to a data warehouse.

The choice of query modes has many implications on system design. Real-time queries require timely data uploads and processing, and historical queries require vast amounts of storage. Supporting only pre-specified queries minimizes data volume, while supporting arbitrary queries on historical data requires vast amounts of data. In this section, we cover system design variables that are mostly unaffected by query mode, and propose three designs that navigate the design space while supporting different query modes.

6.1 Design space

Here, we discuss the design space with respect to where queries are processed, system scaling, and active data collection.

Edge vs. central processing. Queries could be processed on a mix of central server and AV hardware. Driving is a compute-intensive task, and hence any secondary applications should be designed with minimal reliance on the AV's processing power while driving. However, when parked, it's computational resources can be leveraged to process queries. Privacy, a major factor in system design, and is best supported when minimal sensor data leaves the AV, and sufficient resources are available for privacy-preserving anonymization algorithms. Predicate pushdown (i.e, query processing at the edge), also reduces the amount of raw data that needs to be uploaded [37]. However, when edge resources are not available, the data required for delay-sensitive real-time queries should be uploaded for central processing.

Hierarchical system scaling. To effectively support widely geographically-distributed queries at scale, the system for storing data and serving queries should be hierarchical. Servers placed in cities can answer narrow-scale location queries to limit data movement, and partially process wide-scale location queries that are submitted to by higher-level servers. Figure 2 shows a hierarchy with two levels, city-scale servers and nation/state-scale servers; additional levels can be added to support wider-scale queries.

Active data collection. AV's could collect data for answering queries passively, as a byproduct of driving. However, AV's could be encouraged to drive past locations featured in existing un-served queries, or which haven't had recent observations uploaded. Query-aware navigation has a place in a framework that compensates AV owners proportionally for data that is used to answer queries.

6.2 Design Trade-Offs

Selecting the optimal design for each query mode requires a cost model that considers many factors in addition to what we discussed in Section 6.1 that are presently unknown, such as query workload, availability of AV on-board processing, and query service monetization. Here, we sketch three potential AV query system designs, each supporting a different set of query modes.

A: Pre-specified + real-time query design
Queries are broadcast to AVs in the region of interest. The server specifies the required data, and provides the computer vision models for processing the data at the edge. For real-time queries, the data is uploaded to a regional server for timely processing, while other queries are processed opportunistically at the edge and partial results are uploaded to the server for aggregation. AV owners are compensated by clients in proportion to their contribution to the query result, and the AV navigation system takes into account the expected revenue from various travel routes. This design especially favors electric AVs that spend a long time connected to a power source. In this design, AVs comprise a virtual distributed computing and sensing platform for answering real-time queries.

B: Arbitrary historical query design
Data is opportunistically uploaded to a regional warehouse, where it is used to answer historical queries. Some basic processing such as redaction and volume reduction happens at the edge. The AV is made aware of pre-specified queries and ensures that any relevant observations are uploaded, but makes no guarantees on the latency. A hierarchical server setup is used to support queries distributed over wide geographic areas.

C: Arbitrary historical + real-time query design
Real-time queries are pushed to AVs, which upload relevant data upon collection. AVs compress and opportunistically upload data that was not requested in real-time to a regional data warehouse for answering historical queries. A hierarchical server setup is used.

7 Related Work

AV query systems stand to benefit from a broad set of related literature spanning video analytics, sensor networking, and databases for the physical world.

Video Analytics
Recent video analytics frameworks and database management systems, such as BlazeIt [38], Scanner [47], Focus [34], VisualWorldDB [31], and ExSample [43], have made significant progress in query latency over large video datasets. For

example, ExSample reduces the number of processed frames with a probabilistic approach to adapting frame sampling for a given query, and Focus builds an approximate index of frame contents at ingest time to improve video query latency. These systems are designed to operate on video streams; in contrast, the reduced format of uploaded AV data for a given location would more so resemble a collection of images from sporadic angles and times, and violates temporal and spatial data continuity assumptions used by these systems. Nevertheless, similar optimizations can be applied to image collections. In particular, the Vroom system prototype [44] proposes several strategies for ad-hoc queries over AV data streams, including spatial/chronological storage clustering and memoization, that can support historical queries run over the centralized aV data warehouses.

Sensor Network Platforms and Crowdsensing

Querying the world through visual data is possible through satellite imagery [13], aerial drone video (eg. [21], and static camera networks [58, 35]. Queries on satellite imagery are restricted by cloud obstruction and image clarity, in which objects of interest are represented by a handful of pixels, while drones lack the convenience and coverage provided by cars. Surveillance cameras can offer higher clarity, but have low coverage and hardware constraints that limits the edge processing available to support a scalable query system [58, 35].

Cars have a history as a platform for collecting data on the state of the physical world at scale. CarTel [36] is perhaps the most notable system designed to collect and analyze data from sensor-mounted vehicles. We share the broad vision of CarTel: cars could be used to sense the physical world at a larger scale than ever before. However, incorporating the past decade's technological advances in sensory and algorithmic capabilities to detect and identify physical objects presents a different set of opportunities and challenges.

More recently, research on vehicular fog computing and crowdsensing [33, 48, 54, 18] has looked at using vehicles for sensory tasks such as road debris detection, or as passive data mules for low-power remote sensors. Existing work looks at the problems of private data transfer [54, 53], vehicle recruitment [56], reliability [23], and incentives for participation [55]; this work is complementary to the development of AV query systems.

Physical World Databases

AV literature considers maintaining detailed dynamic maps of the physical world (see Reference [12] for an overview). Live crowdsourced mapping has been the focus of several AV technology companies [11, 9]. The maps focus on data specifically related to AV driving, such as vehicle location and road conditions, and do not consider users submitting queries within their ecosystem.

Existing approaches to street-level visual databases, such as OpenStreetCam and Google's StreetView [19], have some limited use-cases, such as demographics estimation [30], localization [52], and virtual tours. These databases are infrequently updated and lack image content annotations that would support efficient querying. For indoor spaces, the concept of a "Marauders map", featuring the location and identity of people and IoT devices such as cameras and smart appliances, has similar privacy challenges [50].

8 Conclusion and Future Work

AVs have the sensory and processing capability to make detailed observations on the world they drive through. We believe that these capabilities should be leveraged across large fleets of AVs to form a mobile sensor network, whose data would offer the most complete and current view of the physical world. To facilitate the querying of this data, we propose a new class of data management systems: AV-based query systems. We characterize a number of challenges around data volume, bias, and privacy, and offered basic solutions to some of these.

Looking forward, we intend to develop benchmark datasets to enable innovation and development of AV query systems using the Carla, an AV-driving simulator, to create realistic driving scenarios not captured by existing datasets. such as multiple AVs driving simultaneously.

Acknowledgments

We thank Deepak Narayanan, Cody Coleman, Sahaana Suri, Sudeep Pillai, and Srinivasan Keshav, as well as the CIDR reviewers, for their insightful feedback. Fiodar Kazhamiaka is supported by an NSERC Postdoctoral Fellowship. This research was supported in part by affiliate members and other supporters of the Stanford DAWN project—Ant Financial, Facebook, Google, Infosys, NEC, and VMware—as well as Toyota Research Institute, Northrop Grumman, Cisco, SAP, and the NSF under CAREER grant CNS-1651570. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

9 References

[1] Google's street view turned down by india. https://indianexpress.com/article/technology/tech-news-technology/googles-street-view-turned-down-by-india-2843618/, 2016.

[2] Mit ecommerce spurs innnovations in last-mile logistics. http://news.mit.edu/2018/mit-e-commerce-spurs-innovations-last-mile-logistics-0904, 2018.

[3] Arizona road debris cause 1000 crashes yearly. https://www.abc15.com/news/operation-safe-roads/secure-your-load-az-road-debris-causes-1-000-crashes-yearly, 2019.

[4] Why germany is a blank spot on google's street view. https://bigthink.com/strange-maps/germany-street-view, 2019.

[5] Amazon s3 pricing. https://aws.amazon.com/s3/pricing/, 2020.

[6] Arizona traffic cameras. https://www.az511.gov/cctv, 2020.

[7] Aws glacier storage pricing. https://aws.amazon.com/glacier/pricing/, 2020.

[8] Cadillac super cruise. https://www.cadillac.com/world-of-cadillac/innovation/super-cruise, 2020.

[9] Deepmap. https://www.deepmap.ai, 2020.

[10] Fujitsu streamlines ai video recognition with high-quality compression technology. https://www.fujitsu.com/global/about/resources/

news/press-releases/2020/0305-01.html, 2020.

[11] Mobileye. https://www.mobileye.com, 2020.

[12] Operational behavior of a high definition map application. https://aecc.org/resources/publications/, 2020.

[13] Planet labs inc. https://www.planet.com/, 2020.

[14] Satellite imagery case studies. https://geocento.com/satellite-imagery-case-studies/, 2020.

[15] Surf cams. https://www.surfline.com, 2020.

[16] Tesla autopilot. https://www.tesla.com/autopilot, 2020.

[17] Yolov5. https://github.com/ultralytics/yolov5, 2020.

[18] D. Agarwal, S. Iyengar, M. Swaminathan, E. Sharma, A. Raj, and A. Hatwar. Modulo: Drive-by sensing at city-scale on the cheap. In *Proceedings of the 3rd ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 187–197, 2020.

[19] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010.

[20] X. Bao, T. Narayan, A. A. Sani, W. Richter, R. R. Choudhury, L. Zhong, and M. Satyanarayanan. The case for context-aware compression. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 77–82, 2011.

[21] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *Proceedings of the IEEE CVPR*, pages 4720–4728, 2018.

[22] C. Bloom, J. Tan, J. Ramjohn, and L. Bauer. Self-driving cars and data collection: Privacy perceptions of networked autonomous vehicles. In *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*, pages 357–375, 2017.

[23] A. Boukerche, B. Kantarci, and C. Kaptan. Towards ensuring the reliability and dependability of vehicular crowd-sensing data in gps-less location tracking. *Pervasive and Mobile Computing*, page 101248, 2020.

[24] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF CVPR*, pages 11621–11631, 2020.

[25] H. Cho, D. Ippolito, and Y. W. Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. *arXiv preprint arXiv:2003.11511*, 2020.

[26] F. I. de l'Automobile. What europeans think about connected cars, 2017.

[27] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3:1376, 2013.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[29] D. Freire-Obregón, F. Narducci, S. Barra, and M. Castrillón-Santana. Deep learning for source camera identification on mobile devices. *Pattern Recognition Letters*, 126:86–91, 2019.

[30] T. Gebru, J. Krause, Y. Wang, D. Chen, J. Deng, E. L. Aiden, and L. Fei-Fei. Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the united states. *Proceedings of the National Academy of Sciences*, 114(50):13108–13113, 2017.

[31] B. Haynes, M. Daum, A. Mazumdar, M. Balazinska, A. Cheung, and L. Ceze. Visualworlddb: A dbms for the visual world. In *Proceedings of CIDR*, 2020.

[32] B. Haynes, A. Mazumdar, M. Balazinska, L. Ceze, and A. Cheung. Visual road: A video data management benchmark. In *Proceedings of the 2019 International Conference on Management of Data*, pages 972–987, 2019.

[33] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Transactions on Vehicular Technology*, 65(6):3860–3873, 2016.

[34] 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)*, pages 269–286, 2018.

[35] W. Hu, B. Amos, Z. Chen, K. Ha, W. Richter, P. Pillai, B. Gilbert, J. Harkes, and M. Satyanarayanan. The case for offload shaping. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 51–56, 2015.

[36] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138, 2006.

[37] L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th MobiSys*, pages 82–95, 2017.

[38] D. Kang, P. Bailis, and M. Zaharia. Blazeit: optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv preprint arXiv:1805.01046*, 2018.

[39] Z. Katona, M. Painter, P. N. Patatoukas, and J. Zeng. On the capital market consequences of alternative data: Evidence from outer space. In *9th Miami Behavioral Finance Conference*, 2018.

[40] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 perception dataset 2020. https://level5.lyft.com/dataset/, 2019.

[41] S. Li, L. Da Xu, and S. Zhao. 5g internet of things: A survey. *Journal of Industrial Information Integration*, 10:1–9, 2018.

[42] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ IROS*, pages 922–928, 2015.

[43] O. Moll, F. Bastani, S. Madden, M. Stonebraker,

V. Gadepally, and T. Kraska. Exsample: Efficient searches on video repositories through adaptive sampling. *arXiv preprint arXiv:2005.09141*, 2020.

[44] O. Moll, A. Zalewski, S. Pillai, S. Madden, M. Stonebraker, and V. Gadepally. Exploring big volume sensor data with vroom. *Proceedings of the VLDB Endowment*, 10(12):1973–1976, 2017.

[45] T. Orekondy, M. Fritz, and B. Schiele. Connecting pixels to privacy and utility: Automatic redaction of private information in images. In *Proceedings of the IEEE CVPR*, pages 8466–8475, 2018.

[46] Otonomo and E. Research. What american drivers think about connected car data and privacy. https://info.otonomo.io/connected-car-survey-edison-lp, 2019.

[47] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian. Scanner: Efficient video analysis at scale. *ACM Transactions on Graphics*, 37(4):1–13, 2018.

[48] L. Pu, X. Chen, G. Mao, Q. Xie, and J. Xu. Chimera: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications. *IEEE Internet of Things Journal*, 6(1):84–99, 2018.

[49] L. H. Rakower. Blurred line: zooming in on google street view and the global right to privacy. *Brook. J. Int'l L.*, 37:317, 2011.

[50] E. M. Schooler, D. Zage, H. Moustafa, and J. Sedayao. A marauder's map for the iot edge. In *2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC)*, pages 236–245. IEEE, 2019.

[51] S. Segal, E. Kee, W. Luo, A. Sadat, E. Yumer, and R. Urtasun. Universal embeddings for spatio-temporal tagging of self-driving logs. *Conference on Robot Learning*, 2020.

[52] A. Taneja, L. Ballan, and M. Pollefeys. Never get lost again: Vision based navigation using streetview images. In *Asian Conference on Computer Vision*, pages 99–114, 2014.

[53] B. Wang, Z. Chang, Z. Zhou, and T. Ristaniemi. Reliable and privacy-preserving task recomposition for crowdsensing in vehicular fog computing. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE, 2018.

[54] J. Wei, X. Wang, N. Li, G. Yang, and Y. Mu. A privacy-preserving fog computing framework for vehicular crowdsensing networks. *IEEE Access*, 6:43776–43784, 2018.

[55] S. Xu, X. Chen, X. Pi, C. Joe-Wong, P. Zhang, and H. Y. Noh. ilocus: Incentivizing vehicle mobility to optimize sensing distribution in crowd sensing. *IEEE Transactions on Mobile Computing*, 2019.

[56] T.-Y. Yu, X. Zhu, and H. Chen. Gosense: Efficient vehicle selection for user defined vehicular crowdsensing. In *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2017.

[57] Y. Zeng, P. Zhang, J. Zhang, Z. Lin, and H. Lu. Towards high-resolution salient object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7234–7243, 2019.

[58] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 426–438, 2015.

APPENDIX

A Min-frame Algorithm Optimality

We offer a proof of optimality for Algorithm 1. In particular, given a set of intervals $\{[x_i,\, y_i]\}$, the set of points S constructed by the algorithm meets the condition that each interval contains at least one of the points in S and is of minimal size. We use the fact that by sorting the intervals in ascending order of y, the problem can be viewed as a series of sub-problems in which we wish to select a point that covers the maximum number of intervals while also covering the first, and then similarly for the first of the intervals that are not covered by the first point, and so on.

Proof. The first point chosen by the algorithm is y_1, since the intervals are numbered in ascending order of y, and the first interval must be covered. y_1 covers the maximum number of intervals that intersect with the first interval. Then, either y_1 covers all intervals and the algorithm terminates with the minimal $|S| = 1$, or there exists $[x_j, y_j]$ where $x_j > y_1$, implying that the minimal $|S| > 1$, as intervals 1 and j cannot be covered by a single point. In the latter case, the intervals $[x_i,\, y_i]$: $x_i \leq y_i$ are covered and removed from consideration. By applying induction on $[x_j,\, y_j]$ as the first of remaining unmarked intervals $\{[x_i,y_i]$: $x_i > y_1\}$, and the algorithm selecting y_j as the next point, we see that the algorithm terminates with a minimal $|S|$.