# REV: A Video Engine for Object Re-identification at the City Scale

Tiantu Xu
Purdue University

Kaiwen Shen
Purdue University

Yang Fu
UIUC

Humphrey Shi
UIUC

Felix Xiaozhu Lin
University of Virginia

***Abstract*—Object re-identification (ReID) is a key application of city-scale cameras on the edge. It is challenged by the limited accuracy of vision algorithms and the large video volume. We present REV, a practical ReID engine that builds upon three new techniques. (1) REV formulates ReID as a spatiotemporal query. Instead of retrieving all the images of a target object, it looks for locations and times in which the target object appeared. (2) REV makes robust assessment of the target object occurrences by clustering unreliable object features. Each resultant cluster represents the general impression of a distinct object. (3) REV samples cameras strategically in order to maximize its spatiotemporal coverage at low compute cost. Through an evaluation on 25 hours of videos from 25 cameras, REV reached a high accuracy of 0.87 (*recall at 5*) across 70 queries. It runs at 830× of video realtime in achieving high accuracy.**

***Index Terms*—camera networks, vehicle re-identification**

## I. INTRODUCTION

**City-scale camera networks: a key edge application** As video intelligence advances and camera cost drops, city cameras expand fast [31], [12]. Camera networks are considered as an important edge application [27]; edge servers are often the ideal platforms for processing camera videos, as transmitting the videos to data centers is often expensive. Being strategically deployed near key locations such as highway entrances or road intersections, multiple cameras (2–5 per location as reported [59], [46]) offer complementary and overlapped viewpoints of scenes.

**Object ReID on city videos** A key application of city cameras is object re-identification (ReID): given an input image of an object X, search for occurrences of X in a video repository. ReID has been an important computer vision task, seeing popular use cases such as criminal investigation and traffic planning [35], [46], [47]. Many ReID algorithms are proposed recently as fueled by neural networks [71], [57], [69], [72], [14], [58], [23], [13]. Object ReID over city videos is typically "finding a needle in a haystack". The videos to be queried are long and produced by many cameras. The videos may *not* contain the input image or any images from the camera that produced the input image (called the *origin* camera). The occurrences of the target object can be rare and transient. For instance, in a popular dataset of city traffic videos [59], 99% of distinct vehicles only appear for less than 10 seconds.

**A common pipeline for ReID** is shown in Figure 1: (1) given an input image of target object X, the pipeline extracts its feature, e.g., using ResNet-152 [19] to extract a 1024-dimension vector [58], [23]; (2) from the queried videos, the pipeline detects all bounding boxes belonging to the same *class* as the target, e.g., using YOLOv3 [55]; (3) the pipeline extracts the features of all detected bounding boxes; (4) it calculates pairwise similarities between X and the bounding boxes. The similarity is often measured as feature distance [45], where a shorter distance suggests a higher similarity between X and a bounding box. Of the four stages, stage (2) and stage (3) are the most expensive. For instance, extracting features in stage (3) is three orders of magnitude slower than calculating feature distance in stage (4). The cost of stage (2) and stage (3) further grows with the amount of videos. This pipeline structure is widely used, e.g., by almost all participants in popular vehicle ReID challenges [23], [58], [43].

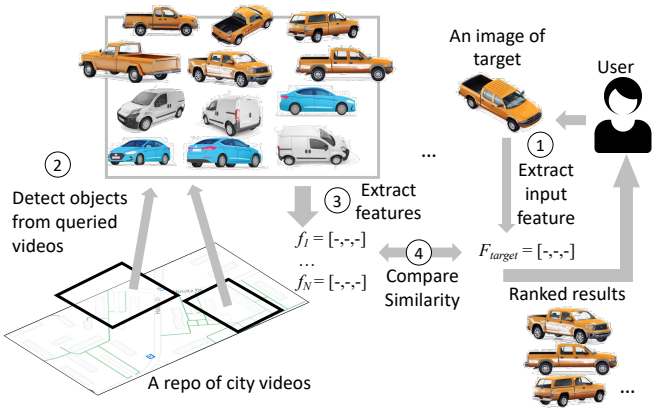



Fig. 1: **The classic pipeline for object ReID**, which formulates ReID as an image retrieval task

Proliferating ReID algorithms call for practical ReID systems. Our driving use case is vehicle ReID, where identifiably personal information such as license plates are intentionally blocked for privacy [59]. Vehicle ReID is one of the most important ReID problems [47]. The solutions are likely transferable to other object classes for ReID.

**Challenge 1: Limitations of modern ReID algorithms** By its definition, ReID focuses on differentiating objects of the *same class*, e.g., vehicles. The inherent difficulty is that in real-world videos, many objects of the same class exhibit only subtle visual differences. Meanwhile, bounding boxes of the same object that are captured by the same or different cameras, may appear quite different. As a result, ReID can be challenging even to human eyes, let alone algorithms. As

Section II will show, state-of-the-art vision algorithms may deem bounding boxes of *different* objects more similar than bounding boxes of the *same* object. Even with a cascade of expensive NNs without considering execution speed, the mean average precision (mAP) accuracies of vehicle ReID are only 0.70–0.85 [58], [23], [20], [43], [59]. In other words, a substantial fraction of bounding boxes returned by ReID is the false positive. This suggests that ReID is much harder than other popular vision tasks, e.g., modern object classifiers, which can achieve accuracy of near 99% [51]. Any practical ReID system shall manage the algorithm limitations explicitly.

**Challenge 2: Many cameras, expensive processing** Recent city deployment reported 2-5 cameras per traffic intersection [59], [46] and 60–500 intersections per square mile in urban areas [11]. Hence, a query covering a few square miles needs to process videos from a few hundred, if not a few thousand, cameras. Modern ReID has an insatiable need for resources. For example, a ∼$3,000 NVIDIA Titan V GPU, detects bounding boxes with YOLOv3 [55] at up to only 45 FPS. The same GPU extracts features with ResNet-152 at ∼90 bounding boxes per second. To process city videos from a square mile in a day, at least a few hundred GPU hours are needed. This cost quickly becomes prohibitive as camera deployment and query scope further expand. Simply using cheap vision operators does not solve the problem, as we will show in the paper.

**REV** We present REV, a practical ReID system that overcomes algorithm limitations, optimizes for resource efficiency, and provides useful ReID answers. REV centers on two insights.

• **No need to label all bounding boxes correctly:** Instead of searching for all occurrences of a target, REV focuses on what users care about: times and locations in which the target appeared. This allows REV to derive robust query answers out of unreliable features of individual bounding boxes.

• **Better to know a bit from many cameras than knowing much from a few cameras:** Prior work often exploits resource/quality tradeoffs *within* a video stream, e.g., tuning frame resolutions, rates, and cropping factors [67], [32], [21], [65], [28]. By contrast, REV focuses on tradeoffs in camera sampling. This is because on city-scale ReID: (1) cameras in different locations provide extensive spatial coverage; (2) cameras near the same location provide complementary viewpoints. As a result, processing more cameras is almost always favorable than processing more pixels from one camera.

With the insights, REV has three noteworthy designs.

**Key design 1: ReID as spatiotemporal queries** While prior research treats ReID as image retrieval – search for every bounding box of a target object X [26], [27], [23], [58], we formulate ReID as a spatiotemporal query: search for times and locations in which object X appeared. REV organizes all videos in spatiotemporal cells. Each cell comprises video clips captured by cameras near a geo-location during a time window. REV answers a query with a short list of spatiotemporal cells that are most likely to contain the target object. The user reviews the returned cells and makes the final assessment.

**Key design 2: Approximating distinct objects with clustering** All objects captured by a camera are heavily impacted by the camera's posture, including its position and orientation. They are also impacted by random visual disturbance, including occlusion and background clutter. REV minimizes the two impacts. It samples co-located cameras, looking for camera postures that match the origin camera. It clusters bounding boxes, mitigating the random visual disturbance. Each resultant cluster thus represents the collective "impression" of a distinct object by the cameras. Clustering has been a classic algorithm for data processing [30] and vision [21]. REV gives it a novel use that derives robust ReID answers out of sparse video frames and unreliable object features.

**Key design 3: Incremental search in spatiotemporal cells** To tame the processing of videos in numerous spatiotemporal cells, REV hinges on its camera sampling strategy. It avoids processing redundant video contents as much as possible while exploiting diverse camera postures as needed. To do this, REV starts executing a query by sampling a minimum number of cameras to estimate the *promises* of all cells, i.e., how likely a cell contains the target object. As the query progresses, REV spends resources on cells where the discovery of the target occurrences is most likely. Based on the processing results, REV updates cell promises and re-prioritizes the cells, which guide the remaining query execution.

We implement and evaluate REV on a dataset of 25 hours of videos from 25 cameras. On 70 queries with diverse target objects, REV delivers high accuracies: recall at 5 of 0.87 and recall at 10 of 0.91 on average. REV is fast: its output converges to high accuracy in 108.5 seconds and 105.1 seconds on average, which are 830×–856× of video realtime. Compared to competitive baselines, REV reduces query delays by up to 6.5×. Our evaluation further validates REV's optimizations that exploit the knowledge of camera deployment.

**Contributions** We made the following contributions.

• Towards a practical and resource-efficient ReID system, we advocate a new approach: focusing on finding relevant spatiotemporal cells rather than individual object occurrences.

• We present to cluster unreliable features of bounding boxes. As approximations of distinct objects, the resultant clusters effectively overcome the limitation in ReID algorithm accuracy.

• We present an incremental search algorithm for spatiotemporal cells. This algorithm exploits diverse camera viewpoints while minimizing the processing of redundant videos.

• We report REV, a first-of-its-kind ReID system that embodies the above ideas and works on large video repositories.

**Ethical considerations** All visual data used is from the public domain; no information traceable to human individuals is collected or analyzed.

## II. Motivations

### A. System model

**Queries & videos** Rev targets retrospective queries: at the query time, all videos are already stored in a central repository, which complements and is orthogonal to real-time processing [27]. We assume a large repository of videos from geo-distributed cameras. Preprocessing at ingestion, i.e., as videos are being captured, is optional, as permitted by compute resource. At ingestion time, the system knows the object classes that will be queried, e.g., cars; the system does not need to know the input images to be used for queries. Hence, preprocessing must be agnostic to specific queries. Retrospective queries can be latency-sensitive because human analysts in the loop, e.g., a crime investigation before the criminal flees. This motivation is shared by many existing video systems [21], [32], [65] .

A query includes an input image of the target object X and the scope of videos to be queried. Following the norm in ReID research [57], [72], [14], [58], [23], [13], we do not assume that the query provides additional information about the image's timestamp or which camera captured the image; we do not assume the video repository contains the input image; we do not assume any other images from the origin camera is available.

**Camera Deployment** Our base design is based on the following assumptions. The deployment spans multiple geo-locations. At each location, multiple cameras are co-located as a *geo-group*. The query system knows which cameras are co-located, i.e., belonging to the same geo-group. Of the same geo-group and during a short period of time, e.g., tens of seconds, cameras are likely (although not necessarily) to capture similar sets of objects from different viewpoints. These assumptions hold for popular city video datasets, as shown in CityFlow [59], where camera locations (the arrow tails), camera orientations (the arrow directions), and camera IDs (the numbers) are clearly annotated in their Figure 2.

Beyond the above, a query system may have deployment-specific knowledge, including quantitative camera postures and correlations across camera geo-groups, e.g., "one object appearing in geo-group A has 50% chance to reappear in geo-group B within the next 10 minutes". To exploit such knowledge, we augment our base design with a series of optimizations to be presented in Section VI.

### B. Challenge 1: Algorithm limitations

**Observation: unreliable features** Figure 2 compares the features of a target vehicle A and a confusing vehicle B. The features are extracted by ResNet-152, a state-of-the-art neural network. Given an image of vehicle A: (1) ResNet-152 deems 10% of B's bounding boxes exhibit shorter feature distances than A's bounding boxes, hence are more similar to the input image; (2) the bounding box most similar to the input image is from the confusing vehicle B but not A; (3) the features of bounding boxes of the same vehicle show a high variation, as reflected by the wide range of the feature distances. The above

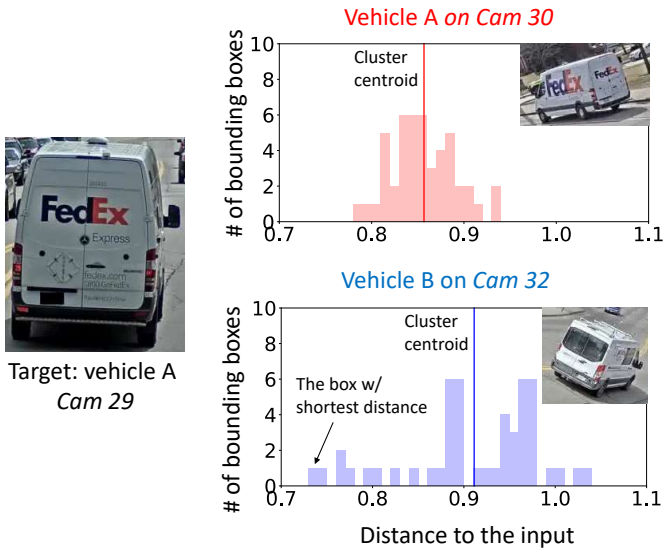



Fig. 2: **Examples of unreliable bounding boxes**. Left: an image of vehicle A, whose feature is the input. Top: a histogram of distances between the input and other features of A. Bottom: a histogram of distances between the input and features of B, a confusing vehicle. All features are $1\times1024$ vectors extracted by ResNet-152. Euclidean distances with L-2 norm [45] are used. Video clips: 4.7/4.9 seconds for vehicle A/B from CityFlow [59]

example of unreliable features is not isolated: they are the major hurdle for ReID accuracy. Our experiment will show that they are the major cause of low query accuracy (Section VII).

**Causes of unreliable features** Ideally, a ReID system should learn an object's inherent characteristics, e.g., its color, shape, skeleton, and key points, and match them to the input. Yet, a camera's actual observation is affected by the camera's posture and the transient disturbances occurred to objects as they move, e.g., changes in size and viewpoint, background clutter, and occlusion. Hence, classic ReID pipelines that aim at labeling each bounding box are heavily affected by the aforementioned factors. The pipelines cannot achieve high accuracy because it cannot eliminate the impacts.

### C. Challenge 2: Numerous cameras & videos

**Colossal data volume** A city camera generates more than 6 GBs of videos daily (720P at 1FPS). Estimated from recent reports on camera deployment in northern American cities [46], [59], [11], the number of city cameras per square mile ranges from a few hundred to a few thousand. A ReID query covering only a few square miles and one day of videos will have to consume PBs of videos.

**Expensive vision operators** Extensive work uses neural networks (NNs) for ReID, advancing the accuracy steadily [62] on public datasets [15], [70]. For instance, recent operators cascade multiple NNs, each detecting a separate set of vehicle features, e.g., orientations or roof types. The additional NNs are reported to improve accuracy (mAP) by 10% at the cost

of 7× overhead [58], [22]. We estimate that they run no more than 15 FPS on a modern GPU.

**Would cheaper operators help?** Cheaper NNs and vision primitives were used in detecting object classes as tradeoffs between accuracy and cost [32], [21], [64], [6]. Yet, they are unlikely remedies to ReID. This is because ReID tasks require to differentiate *objects of the same class*, and cheap operators simply do not offer surplus accuracy for systems to trade off. We have evaluated the option of two cheap vision primitives, RGB histogram [48] and SIFT [41]. RGB histograms are highly volatile to changes in lighting conditions and background clutter; SIFT captures various local features, which is much less reliable compared to global features extracted by modern NNs, while not significantly faster than the latter. Section VI will present more details.

### D. Why is prior work inadequate

**Computer vision research** typically treats ReID as an image retrieval task [71], [58], [22], [43]. Aiming at finding all bounding boxes of a target object, computer vision studies focus on improving accuracy without considering query speed or efficiency much. However, retrieving every bounding box would miss the opportunities in answering spatiotemporal queries with much lower delays.

**Existing ReID systems** are often designed for and evaluated on smaller camera deployment, e.g., 8 cameras over a university campus [15]. They lack mechanisms for processing city-scale videos. Many designs work around the vision algorithm limitations by relying on deployment-specific knowledge, e.g., strong spatiotemporal correlations across cameras [27], knowledge of object trajectories [8], road coordinates, and vehicle transition patterns [60]. It is unclear how well such assumptions universally hold for city-scale deployment. By contrast, our design is more generic as it only requires information from video frames and can nevertheless optimize queries with additional information as they become available (Section VII).

**Spatiotemporal databases** are designed for managing object trajectories, e.g., airplane movements and human movement, and answering queries about them [24], [10], [2], [50], [52]. They ingest *structured* data, e.g., sequences of GPS coordinates; they cannot ingest unstructured video data and recognizes object occurrences as REV does. The output of our system can be the source of a spatiotemporal database.

## III. SYSTEM OVERVIEW

REV organizes all videos in a repository in spatiotemporal cells, where a cell <$L$, $T$> comprises video clips captured by cameras near a geo-location $L$ during a time period $T$. To query for a target object X, a user submits an image of X. REV answers the query with a short list of spatiotemporal cells ranked by their likelihood of containing the target object X. Alongside each returned cell, REV presents video clips and annotates the bounding boxes likely to be of X. While executing a query, REV keeps updating the rank based on its video processing progress.

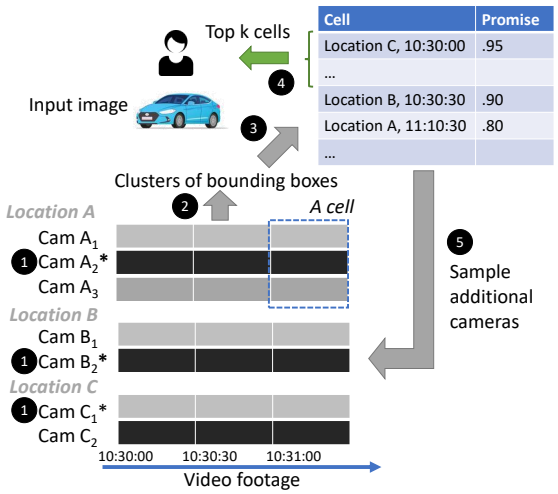



Fig. 3: **An overview of REV**. (* indicates a starter camera)

To answer spatiotemporal queries, REV does not have to find all appearances of the object, which can be costly if at all possible. Instead, REV focuses on finding a cluster of object appearances and uses the cluster to identify potential times/locations that are likely to contain the target object, which is much more robust and efficient.

We design REV as a recall-oriented system [4], seeking to find all positive cells and rank them to the top. As such, REV minimizes human efforts in analyzing videos. We do not intent REV to replace humans because vision algorithms cannot (yet) substitute human assessment on real-world videos. The same spirit is seen in other popular recall-oriented systems, e.g., search for legal documents or patents [3], [5], where final decisions from humans are vital.

**Video ingestion** At ingestion time, REV pre-processes videos captured by a small number of cameras. The processing is optional, as permitted by REV's compute resource. The pre-processing detects objects of interesting classes (e.g., vehicles) on the selected cameras and extracts their features; it is oblivious to any specific input image. At ingestion time, REV also profiles videos, which is a common practice of video systems [21], [27], [64], [63]. It periodically samples videos from each camera to train parameters for clustering bounding boxes and for sampling cameras. We will discuss these parameters in Section IV and Section V. The profiling is light, processing 30 seconds of every 1-hour video and taking less than 10 seconds on a modern GPU.

**Executing a query** As shown in Figure 3, REV searches all cells iteratively; it processes videos from additional cameras in an incremental fashion. It starts by sampling from all cells in the query scope. The initial sampling is brief, as REV only processes a small fraction of video footage from selected cameras (dubbed "*starter cameras*") ❶. From the sampled video of a cell, REV detects distinct objects out of unreliable bounding box features; it does so by clustering these features. Each resultant cluster represents a distinct object, where the cluster's centroid is an approximation of the object's

true feature ❷. REV ranks all the cells by their *promises*, which are estimated as the similarity between a cell's objects and the input image ❸. REV emits the ranked cells as the query results to users, who review the top ones ❹. For the remaining cells yet to be decided, REV processes videos from additional cameras and uses the results to update the cell rank continuously ❺. A query is terminated by users when they are satisfied with the current results, or by REV when it finishes processing all videos in the query scope.

REV will not miss any target object – if a query is executed to completion, all videos will be searched. REV is designed to reduce the latency of finding the target object, a goal valid even in offline search. REV has a number of hyper-parameters as many ML systems do. Fortunately, REV's performance is less sensitive to these hyper-parameters as we measured (Section VII-E). Furthermore, REV may continuously update its hyper-parameters as it runs.

**Limitations** REV inherits the statistical nature of its underpinning ReID algorithms, notably neural networks. While REV delivers high-confidence results, e.g., our results show that on more than 70% of queries, it ranks all the true cells among the top 5 (Section VII), it cannot provide sound guarantees. Similarly, although REV's output accuracy often quickly converges as it executes a query, there is no guarantee on the convergence rate. Our expectation is that users review the top k cells to confirm the target object's existence; they entrust REV on the remaining unreviewed cells, being comfortable with the level of confidence that REV provides. However, in case they want to be absolutely certain that no true cells are left out of the top k, they would need to inspect all the videos.

## IV. CLUSTERING BOUNDING BOXES

A core mechanism of REV is to recognize distinct objects from bounding boxes and compare the recognized objects to the input image. The mechanism serves two purposes:

- Working around the unreliable features of bounding boxes, which addresses the algorithm limitation of feature extractors.
- Tolerating low frame rates of videos, which allows REV to maximize the camera coverage in query execution.

**Clustering similar bounding boxes** Bounding box features as captured by cameras are subject to sudden or graduate disturbance. For example, as a vehicle travels towards or away from a camera, the size of its bounding box changes; the view angle may change; it may be occluded by obstacles such as a light pole; its background may be intruded by other vehicles.

Disturbance to bounding box features is difficult to model and eliminate in general. Fortunately, if we consider similar bounding boxes in consecutive video frames (even if they are sparse in time), the feature disturbances due to graduate impacts are likely to smooth out, and the outlier features due to sudden impacts can be removed from consideration.

To this end, REV clusters object features based on their similarities. The similarity, for instance, can be measured by Euclidean distances across 1024-dimension feature vectors. As a result, each cluster represents a camera's *general impression* of a distinct object during a given time window. The cluster's centroid is an approximation of the camera's true observation of the object. REV's clustering is novel in that it overcomes the accuracy limitations on individual bounding boxes for ReID. It differs from prior work [21] that uses clustering for processing efficiency, e.g., to avoid processing similar objects. Figure 2 showcases why clustering is useful. Once we cluster the respective bounding boxes of the two vehicles, the centroids distances (solid vertical lines) are much more robust indicators of object similarity, suggesting the general impression of vehicle A is much closer to the input image.

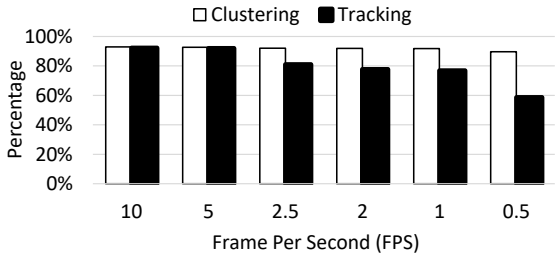



Fig. 4: **Clustering bounding boxes tolerates low frame rates.** Y-axis: the percentage of bounding boxes correctly attributed to respective objects. Object tracking implemented in OpenCV 3.4.4. Videos from CityFlow [59]

**Predicting the number of distinct objects ($k$)** REV runs k-means clustering within each spatiotemporal cell. By minimizing the sum of intra-cluster variances across all clusters, k-means thus puts most visually similar objects in the same cluster. k-means guarantees convergence to local optimum and is known robust to outliers [33]. We also tested other popular clustering methods, e.g., hierarchical clustering. We find them less favorable. For instance, they often attribute bounding boxes of the same object to separate clusters.

As a prerequisite for applying k-means, REV must specify $k$ as the number of distinct objects in each video clip. An accurate $k$ is crucial to the clustering outcome. REV predicts $k$ based on a simple intuition: of a given video scene, the distinct vehicle number is correlated to the spatial density of bounding boxes. Therefore, REV only needs twofold information to predict $k$: (1) $x_1$, the number of bounding boxes detected in the video clip; (2) $x_2$, the number of frames that contain non-zero objects. Such information is already available from object detection (the ReID stage that precedes feature extraction), as described in Section I. From $x_1$ and $x_2$, REV further derives three variables as different orders of the box/frame ratio: $x_3 = (x1/x2)^2$, $x_4 = (x1/x2)$, and $x_5 = (x2/x1)$.

We formulate a regression model: $k = \mathbf{a}\mathbf{x} + b$. The model takes as an input $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5]$ which consists of variables above; its parameters are a vector $\mathbf{a}$ and a scalar $b$. We instantiate one shared model for all cells, for which we train $\mathbf{a}$ and $b$ offline in one shot on 30-second labeled videos.

**Tolerance of low frame rates** k-means clustering is robust to low frame rates. As a comparison, we have investigated object tracking. We find that object tracking demands a much higher frame rate than clustering in order to estimate trajectory

with good accuracy. Figure 4 shows an experiment. As the frame rate drops to below 2.5 FPS, object tracking quickly loses accuracy to become barely useful. By contrast, k-means still maintains high accuracy of over 90%. Object tracking also suffers from other difficulties, e.g., differentiating nearby objects that move along similar trajectories [1].

The experiment also shows that increasing the frame rate for clustering leads to a diminishing return. When the frame rate increases from 1 FPS to 10 FPS, the clustering accuracy improves by less than 3%. This supports our principle: prioritizing camera coverage over video fidelity.

## V. Incremental search in spatiotemporal cells

The search mechanics address two questions. (1) How likely does a cell contain the target object? (2) For which cells Rev should process videos from additional cameras?

### A. Assessing cell promises

Rev quantifies the likelihood of a cell containing the target by *promise*: a cell shows high promise if *any* object in this cell is similar to the input image. Promise reflects the following rationale: due to the rarity of the target object occurrences, a cell is promising as long as any camera has captured any object that is sufficiently similar to the input image. Such a cell is more likely to contain the target than another cell with many objects *somewhat* similar to the input. We define the *single-camera promise* for a cell $\mathscr{C}$ as $p_{single}(R, \mathscr{C})$, which is the promise Rev perceives assuming it only processes one video clip from camera $R$. $p_{single}$ is reciprocal to the smallest feature distance between the input and any centroid of object clusters from the video clip, i.e., $p_{single}(R, \mathscr{C}) = 1/min(dist(X, o))$ where $X$ is the feature of target object and $o$ is the centroid of any object features cluster. Rev further maintains the overall promise for $\mathscr{C}$ as the *highest* of single-camera promises of $\mathscr{C}$.

### B. Prioritizing cells in search

A cell's promise reflects the single object most similar to the input. The metric, however, is inadequate for Rev to decide whether additional cameras are worth sampling for a cell. Instead, Rev needs to track the *accumulated* findings for the cell and the *accumulated* search efforts spent on the cell. For instance, if Rev has sampled a large fraction of cameras for a cell already but only discovering objects with low similarity to the input, Rev should consider prioritizing other cells.

Rev maintains the cells in the following categories:

- Green cells: Rev has collected enough – though not necessarily all – evidences for them, and predicts them *likely* to contain the target object. Sampling from additional cameras is unlikely to change this assessment.
- Red cells: Rev has collected enough evidences and predicts them *unlikely* to contain the target object. Sampling from additional cameras is unlikely to change this assessment.
- Gray cells: the existing evidences are insufficient. Sampling from one or a few cameras will likely turn a cell to red or green. Just like how humans would decide on a suspicious cell, Rev will request additional video from a different viewpoint.

**The search plan** All cells begin in gray. Rev starts from processing videos for the the gray cells which are undecided, to the green cells (to refine their rank of presentation to the user), and then to the red cells (in the unlikely event that any true cells fall into this category). Based on the new processing results, Rev moves cells across categories, as will be discussed below. Rev exhausts all cells in a category before moving to the next category. In each category, Rev always processes the cell that has the highest overall promise.

**Categorizing cells with camera votes** To categorize a cell $\mathscr{C}$, Rev incorporates observations from multiple cameras by voting. The voting mimics how humans would make a collective decision. Rev quantizes all single-camera promises with two thresholds, $P_{high}$ and $P_{low}$. A camera with promise $p > P_{high}$ casts a high-confidence vote with a weight of 1; a camera with promise $P_{high} < p < P_{high}$ casts a medium-confidence vote with a weight of $1/k$. $k$ controls the relative weights of high and medium confidence votes. In the current implementation, we set $k = 2$. As a result, Rev moves a cell to the green category if one camera casts a high-confidence vote or two cameras cast medium-confidences votes.

$P_{high}$ and $P_{low}$ depend the tradeoff between refining existing results and exploring for new results. A lower $P_{high}$ would eagerly put cells in green, postponing sampling of additional cameras for them until much later. By comparison, a higher $P_{high}$ would be lazier in moving a cell to green.

We set $P_{high}$ high so that Rev tends to refine the rank of promising cells. This is because we expect users to only inspect several top cells; it is thus vital to aggressively push the true cells to this small range. Based on the same rationale, we tune $P_{low}$ to a low value in order to admit more cells to the gray category. In our implementation, we set $P_{high} = 1/d_{short}$, where 99% of the bounding boxes with feature distance shorter than $d_{short}$ indeed belong to the same vehicle. We set $P_{low} = 1/d_{long}$, where 99% of the bounding boxes that belong to the same vehicle have feature distances shorter than $d_{long}$. We will evaluate their sensitivity.

### C. Putting it together: the query execution flow

**Stage 1: Initial sampling of cells** Rev starts a query by processing one starter camera for each cell in the query scope. Based on videos from the starter cameras, Rev recognizes distinct objects in each cell; for each recognized object, Rev derives their cluster centroids. In this stage, Rev prioritizes cells if there exists any heuristics on which cells are more likely to contain the target object, e.g., video clips captured during rush hours. Rev uses a low video frame rate (1 FPS) tolerable to the clustering algorithm (Section IV).

After processing the starter cameras for all cells in the query scope, Rev has assigned cells to their initial categories. The cells in each category are ranked by their promises.

*Choosing starter cameras.* The starter camera choices matter as they set the initial search direction. Ideally, the starter cameras should be the ones most likely to have captured the target from a viewpoint *similar* to the input image. In practice,

REV can exploit knowledge on camera deployment to pick starter cameras, as will be described in Section VI. If such prior knowledge is unavailable, our base design simply picks cameras that have the highest *density of distinct objects*. The heuristics is that such cameras will have higher chances of capturing the target object. To do so, REV profiles each camera's density of objects offline and picks the starter cameras ahead of the query. We evaluate REV's sensitivity to the starter camera choices in Section VII-E.

*Pre-processing at ingestion.* Stage 1 can be executed ahead of queries because its processing is independent of specific input images. Such pre-processing at ingestion is optional and elastic. The number of starter cameras REV can process depends on resources, e.g., the number of GPUs. REV processes unprocessed starter cameras right after a query starts and caches the results for future queries (Section VI).

We also consider the situation where *ample* resources are available at ingestion. Is it worth pre-processing multiple starter cameras per geo-group? Our experiments in Section VII suggest diminishing returns. This is because a small number of starter cameras properly chosen can already yield decent estimations for cell promises and hence a good initial ranking.

**Stage 2: Incremental search** Based on the initial results from starter cameras, REV may already have put some cells in gray. Common causes include: the starter cameras never captured the target vehicle; they captured the target vehicle but from viewpoints significantly different from the input; they captured a different but visually similar vehicle. REV resolves these gray cells by sampling videos from additional cameras.

REV picks the next cells as follows. (1) It first looks for unprocessed cameras for the highest-ranked gray cell. (2) If REV has already processed cameras for all gray cells, it moves to the highest-ranked green cell that still has unprocessed cameras. (3) If all cameras for green cells are processed, REV moves to red cells, hoping to find target object instances in those cells missed out previously. After picking up the next cell and processing one additional camera for it, REV updates the cell's category and re-rank all the cells. REV then picks the next cell based on the updated categories and ranks.

## VI. OPTIMIZATIONS

### A. Exploiting prior knowledge

**Picking starter cameras based on posture similarity** If the quantitative postures of deployed cameras are known to REV, e.g., as part of per camera metadata, REV can pick the starter cameras as the ones having postures most similar to the origin camera, i.e., the camera producing the query image. To do so, REV estimates the posture of origin camera in two alternative ways: it can rely on a human analyst to annotate the posture, which only requires annotating one image per query; it may automate the estimation with vision algorithms. Section VII-F will evaluate the former method.

**Prioritizing cameras with complementary postures** In our base design, when REV samples a secondary or subsequent camera for a cell, it picks a random one from the same geo-group. If REV has prior knowledge of camera postures, it can make more informed decisions. Following the selection of the starter camera, REV picks the next camera as the one that offers the most different viewpoint compared to the prior camera, i.e., the N-th camera is always the camera that has the largest viewpoint difference from the (N-1)-th camera.

| | | | |
|---|---|---|---|
| **# of all cameras** | 25 | **Total video length** | 25 hours |
| **# of geo-groups** | 7 | **# of distinct vehicles** | 70 |
| **# of total cells** | 3000 | **# of all bounding boxes** | ~1M |
| **Time duration of a cell** | 30 secs | | |

TABLE I: The video dataset used in evaluation

**Reusing states of previous queries** REV speeds up a query's execution by reusing the states from prior queries on the same videos. These queries may have different input images; they could be fully or partially executed. The states include all distinct objects recognized from the starter cameras, as well as some of the bounding boxes, distinct objects, and features from other cameras. To generate the initial ranking of cells, REV may reuse the existing distinct objects; in incremental search, REV may favor cameras for which partial results already exist. Section VII-F will evaluate the idea.

### B. Utilizing cheap vision operators

While our base design uses ResNet-152, we further investigate cheaper operators with lower accuracy in the following ways. We will evaluate these techniques in Section VII-F.

**Using cheap operators during ingestion for early ranking** When there are insufficient resources to execute the "gold" ResNet-152, REV falls back to deriving a rough rank of spatiotemporal cells to guide future incremental search in cells [63]. The choices of pre-processing are rich, e.g., processing more starter cameras with cheaper operators or processing fewer cameras with more expensive operators.

**Using cheap operators to filter object instances** REV clusters the cheap features, discard unpromising object instances, and only extract expensive features for the surviving bounding boxes. Accordingly, it adjusts $k$ to the number of surviving clusters. Note that REV cannot cluster features from different extractors, e.g., ResNet-50 and ResNet-152, because these features are different object representations.

**Cheaper operators in lieu of the gold operator** We will study REV's query speed and accuracy in this situation.

## VII. EVALUATION

We answer the following questions in evaluation:
§VII-B Can REV achieve good accuracy with low delays?
§VII-C Are the key designs useful?
§VII-D Does REV outperform prior alternative designs?
§VII-E REV's sensitivity to its parameters and query inputs?
§VII-F How effective are REV's add-on optimizations?

### A. Methodology

**Video Dataset** An ideal video dataset for benchmarking REV would: (1) comprise long videos produced by many

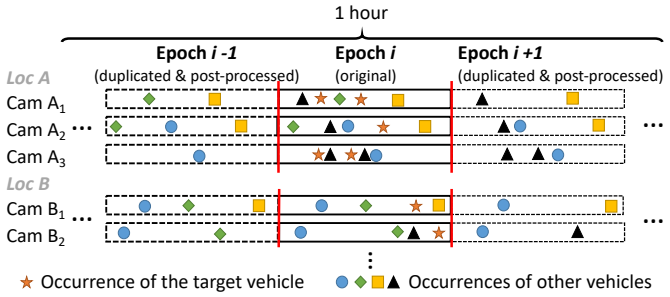



Fig. 5: **Augmenting real-world city videos [59] as our test dataset**: duplicating the original epoch; erasing random vehicles from each epoch; erasing the target vehicle from all but the original epoch.

cameras; (2) come from real-world deployment and capture spatiotemporal patterns of vehicles; (3) provide vehicle labels as the ground truth for accuracy evaluation. We prefer real-world videos [59] over simulators such as VisualRoad [17]. This is because a simulator generates long animations based on user-defined vehicle motions and camera postures, which do not necessarily reflect real-world ReID challenges, e.g., object rarity, camera diversity, and disturbance.

We use CityFlow [59], a public video dataset that best suits our needs. By NVIDIA for the AI City Challenge 2019, the dataset consists of 5 scenarios, from which we select the largest one (scenario 4). The video footage spans 30 minutes in total, as captured by 25 cameras at 7 traffic intersections (hence 7 geo-groups) of a northern American city. The footage captures 17,302 vehicle bounding boxes belonging to 70 distinct vehicles. We downsample the videos to 1 FPS, a low frame rate adopted in prior video systems [21], [32], [65].

We overcome a shortcoming of the CityFlow dataset [59]: each of the videos lasts around 30 seconds on a camera; by contrast, a real-world ReID query often spans video footage of hours or days. Using short videos trivializes the query execution: the number of spatiotemporal cells is small, and the target object is not rare. We augment the dataset, extending the video length while introducing minimum bias.

Our augmentation is shown in Figure 5. First, we duplicate the original video clips to be many *epochs*, each comprising video clips from all cameras. By doing so, we preserve the spatiotemporal patterns of vehicle instances within each epoch. Second, we remove a random fraction (0–1) of vehicles from each epoch, erasing their bounding boxes from all the video clips in that epoch. This diversifies the augmented videos over time, preventing them from becoming repeated loops of the original clips. Third, we ensure that target objects are rare and difficult to find. For a query with an input image of target X, we erase all X's bounding boxes from duplicated epochs while only keeping ones in the original epoch. We further exclude the origin camera from the query scope. Our data augmentation makes ReID *more challenging*: it extends the videos, diversifies vehicle occurrences, and preserves true vehicle rarity. Meanwhile, it preserves the vehicles' spatiotemporal patterns. The final videos used in the evaluation are summarized in Table I. They span 25 hours, 1 hour per camera. Together, the videos consist of 3000 cells of 30 seconds each and more than 1 million bounding boxes. Given a query, only 243 bounding boxes on average (0.02% of all) belong to the target vehicle, and 1.6 cells on average (0.5% of all) contain the target.

**Query setup** We test REV on 70 queries, each for one distinct vehicle in the video dataset. A query contains one vehicle image randomly selected from all bounding boxes of the vehicle in the dataset. We then exclude the origin camera from the query scope. As described in Section III, a query does not specify the timestamp or the origin camera of the input image, as opposed to prior work [27].

**Environment** REV runs on 12-core Xeon E5-2620v3 with an Titan V GPU. REV detects vehicle bounding boxes with YOLOv3 [55] and extracts their features with ResNet-152 [19]. We train ResNet-152 on images from 329 different vehicles from 34,760 images from the datasets of CityFlow [59] and Cars [34], with all vehicles used in the evaluation excluded.

**Accuracy Metric** We measure a query's accuracy with *recall at k*, i.e., the fraction of all true cells included in the top k output cells. True cells are the cells containing the target object. *Recall at k* is commonly used in recall-oriented systems, which focus on retrieving rare objects [3], [5]. By setting k as low as 5, the resultant metric (*recall@5*) measures the usefulness of query results – how likely a user finds true cells by only reviewing the top 5 cells returned by REV. A high value of *recall at 5* means that REV successfully finds most if not all true cells, since the true cells of most queries (> 98% in our dataset) are fewer than 5. We also report *recall at 10*, a more relaxed metric considering ranking true cells among the top 10 as acceptable.

Note that precision, i.e., the percentage of true cells in the top k cells, is not a suitable metric to REV. First, since users only focus on a small set of top results returned by REV, precision does not reflect the results quality or user efforts. Second, the number of true cells varies across queries, making precision less meaningful. Assume there is one true cell. Even if REV ranks it as the top 1, precision@5 will be as low as 0.2. We are unaware of precision being used for other recall-oriented queries, e.g., web or patent search.

**Speed metric** As REV keeps refining the rank of cells, we report the delays for its output accuracy (recall at 5) to reach accuracy goals: 0.25, 0.50, 0.75, and 0.99. We use *eventual* accuracy to refer to the output accuracy after REV processes all videos in the query scope. We measure delays only for queries on which REV's eventual accuracies meet or exceed these accuracy goals. REV cannot meet high accuracy goals, e.g., 0.99, on a small fraction of queries.

**Counting distinct vehicles** To predict the number of distinct vehicles in each spatiotemporal cell, we trained the regression model as in Section IV. On average, the predicted number of distinct vehicles is only 6.5% different than the true number.

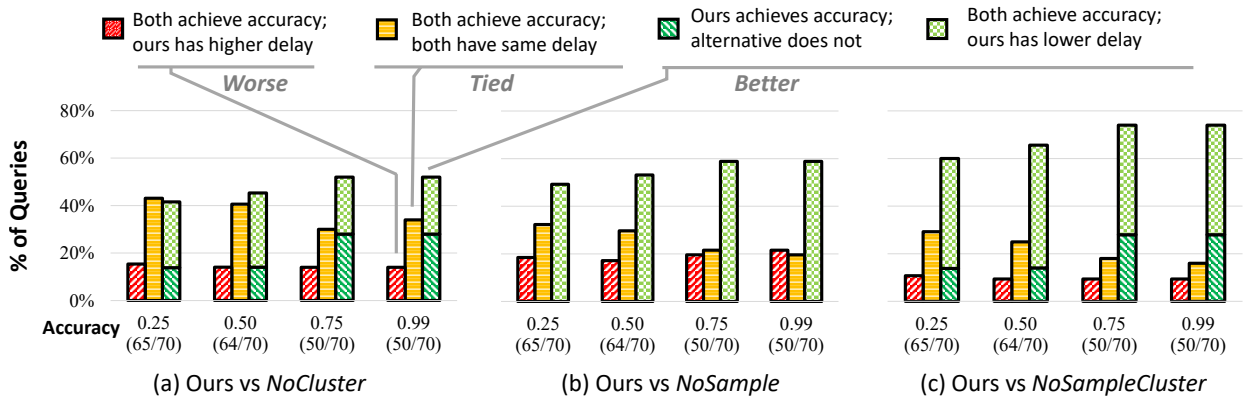



Fig. 6: **Query-by-query comparison between REV and the alternatives**, broken down by per-query comparison outcomes. Numbers (X/Y) on bottom show accuracy goals: X = the number of queries on which REV reaches the accuracy goal; Y = total query count.

### *B. End-to-end performance*

**Accuracy** On most queries, REV achieves high eventual accuracies. All 70 queries achieve an average recall@5 of 0.87. Among them, 64 queries (91%) meet or exceed an accuracy of 0.50; 50 queries (70%) meet or exceed an accuracy of 0.99. On the metric of recall@10: all 70 queries achieve an average accuracy of 0.91; among them, 69 queries (99%) meet or exceed an accuracy of 0.50; 58 queries (83%) meet or exceed an accuracy of 0.99. Such accuracy achieved through clustering is higher than what can be achieved on individual bounding boxes, explained in Section IV. This validates that clustering makes robust decisions based on unreliable features.

We manually inspect the six queries where the accuracy is lower than 0.5, attributing the cause to the limitation of feature extractors. For instance, when vehicle 262 is used as the input, it is challenging even for humans to associate the input image with all the 30 true bounding boxes. Not surprisingly, such inputs confused the feature extractors.

**Delays** REV meets accuracy goals with moderate delays. In querying 25 hours of videos on our single-GPU machine, REV takes 59.5 seconds on average (stddev: 156.2, 90th percentile: 457.5) to reach an accuracy of 0.50, and 108.5 seconds on average (stddev: 194.9, 90th percentile: 488.0) to reach 0.99. Roughly, this speed is 830× of video realtime, i.e., 4.3 seconds to perform ReID on each hour of videos.

### *C. Significance of key designs*

We test several versions of REV with key designs off:

- *NoCluster*: Clustering is off. This version ranks a cell based on the minimum pairwise distance between the input image and bounding boxes in the cell. With the rank, this version searches in cells by sampling from cameras as REV does.
- *NoSample*: Camera sampling is turned off. The version randomly picks starter cameras for each camera group. It clusters bounding boxes and ranks cells accordingly, just as REV does. Unlike REV's incremental processing of cameras and updating of the cell rank, this version processes all cameras for a cell before updating the rank.
- *NoSampleCluster*: Both clustering and sampling are off. This version ranks a cell based on the minimum distance between the input image and the bounding boxes; it processes all cameras for a cell before updating the cell rank.

Figure 6 summarizes REV's strength against the alternative versions. On most queries, REV outperforms the alternatives, either reaching higher accuracy or the same accuracy in lower delays. Only on a small fraction of queries, REV shows longer delays; REV never fails to reach the accuracy attainable by alternatives. Note that Figure 6 does not include the latencies for queries *failed* to reach the target accuracy: we consider these queries as incomplete, thus less meaningful to include.

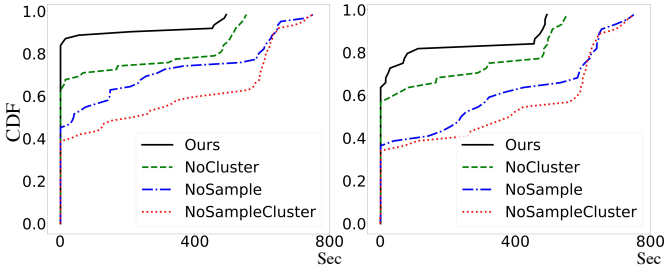
**Clustering improves query accuracy** Compared to the alternatives without clustering, REV's per-query accuracy gain higher by 0.13 on average (stddev: 0.28). Such an accuracy gain is considered significant in prior work [62]. REV's accuracy is higher on 14 out of all 70 queries; on the remaining queries, REV's accuracy ties with the alternatives (mostly with short delays, see below) and is never lower. Clustering is vital to accuracy in two ways: (1) it is robust against strong feature noise, which is the key to achieve high accuracy goals such as 0.99; (2) it makes the initial cell ranking more accurate.

**Camera sampling reduces query delays** REV is much faster in reaching accuracy goals. Figure 7 shows the delay CDFs. With accuracy goals of 0.50 and 0.99, REV's delays are 2.9× and 1.7× shorter than *NoCluster* on average, 4.5× and 3.3× shorter than *NoSample* on average, and 6.5× and 3.9× shorter than *NoSampleCluster* on average. *NoSampleCluster* and *NoSample* suffer from poor choices of starter cameras, which in turn result in the poor initial rank of cells.

### *D. Comparison to prior works*

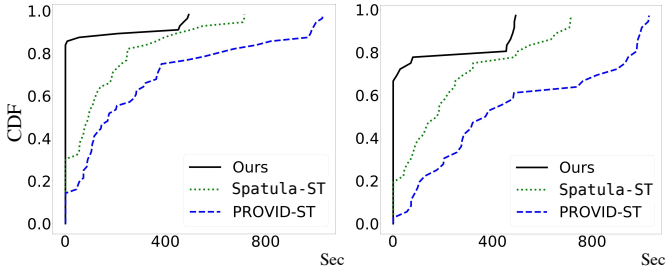
We next compare REV to two prior works: Spatula [27] and PROVID [38]. We modify them to support spatiotemporal queries. These designs lack our techniques of clustering and sampling; they randomly pick the starter cameras.

- **Spatula-ST: exploiting object spatiotemporal correlations**. Unlike our design which samples cameras to rank cells,




(a) Delay to reach accuracy 0.5 (56/70) (b) Delay to reach accuracy 0.99 (36/70)

Fig. 7: The CDF of query delays by REV and the alternatives, showing REV runs much faster. (X/Y): X = the number of queries on which all the versions reach the accuracy goal; Y = total query count.




(a) Delay to reach accuracy 0.5 (56/70) (b) Delay to reach accuracy 0.99 (36/70)

Fig. 8: The CDF of query delays by REV, Spatula-ST, and PROVID-ST, showing REV runs much faster. (X/Y): X = the number of queries on which all the versions reach the accuracy goal; Y = total query count.

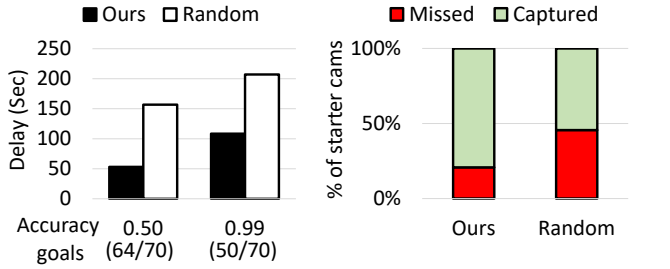
Spatula-ST utilizes Spatula's [27] cross-camera correlation model for ranking cells. To do so, Spatula-ST profiles (1) the spatial correlation of object occurrences, i.e., the portion of overlapped objects across different cameras; (2) the temporal correlation, i.e., the time difference in which an object is likely to reappear in other cameras. At ingestion time, Spatula-ST randomly picks one starter camera from each geo-group and derives an initial rank of all cells. At query time, Spatula-ST considers the current top cells as the most promising cells; it prioritizes the cells from the most correlated cameras and time ranges, processing them to update the rank of cells.

• **PROVID-ST: Re-rank by spatiotemporal constraints**: PROVID-ST ranks bounding boxes by taking into account the likelihood of a vehicle reappearing in another location and time [38]. Following the work, PROVID-ST calculates spatiotemporal proximity as follows:

$$s = \frac{|D_i - D_j|}{D_{max}} \times \frac{|T_i - T_j|}{T_{max}} \quad (1)$$

$|D_i - D_j|$ stands for the geo-distance between cell $i$ and $j$, which is normalized by $D_{max}$, the longest distance among all camera groups; $|T_i - T_j|$ stands for the time difference between cell i and j, which is normalized by $T_{max}$, the entire query time frame. A smaller $s$ indicates closer spatiotemporal proximity. Therefore, PROVID-ST first calculates a cell's promise $p$ as our design does and further the promise with the cell's spatiotemporal proximity to all the green cells, i.e., $p\prime = p/s$.

**Comparison results** REV outperforms Spatula-ST and PROVID-ST. Spatula-ST's average accuracy is 0.75, 13.8% lower than REV. PROVID-ST's average accuracy is 0.66, 24.1% lower than REV. The major causes are the lack of clustering and the limitation of their spatiotemporal models, e.g., PROVID misses vehicles traveling a long distance over a long time. As shown in Figure 8, REV executes faster than Spatula-ST and PROVID-ST on most queries. With accuracy goals of 0.50 and 0.99, REV's average delays are 3.2× and 2.3× shorter than Spatula-ST, and 6.4× and 4.8× shorter than PROVID-ST, respectively.



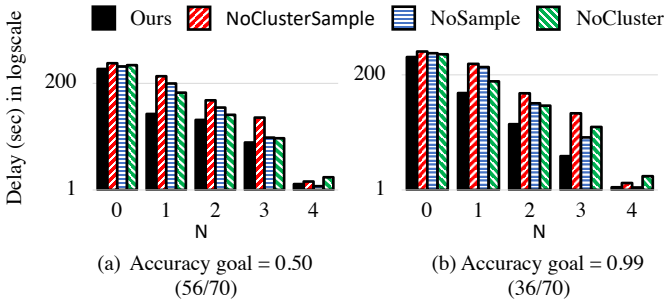

(a) Delay in reaching accuracy goal, averaged over queries (b) Starter cameras having captured / missed the target object

Fig. 9: **A comparison between REV's choices of starter cameras and random choices, showing the significance of the choices**. (X/Y): X = the number of queries reaching the accuracy goal; Y = total query count.

### *E. Sensitivity to parameters and inputs*

**Choices of starter cameras matter** While not affecting a query's eventual accuracy, the choice of starter cameras has a high impact on query delays. This is because the choice affects the initial rank of cells. As shown in Figure 9(a), with starter cameras randomly picked, the average query delays grow by 1.5× and 3.9× to meet accuracy goals of 0.50 and 0.99, respectively. Figure 9(b) shows the cause: a substantial fraction of random starter cameras miss the target vehicle they should have captured (i.e., the target captured by other cameras in the same geo-group), missing opportunity in optimizing the initial ranking of cells. With good choices of starter cameras, as shown in Figure 7, about 80% and 60% of queries can directly reach 0.50 and 0.99 without additional cameras.

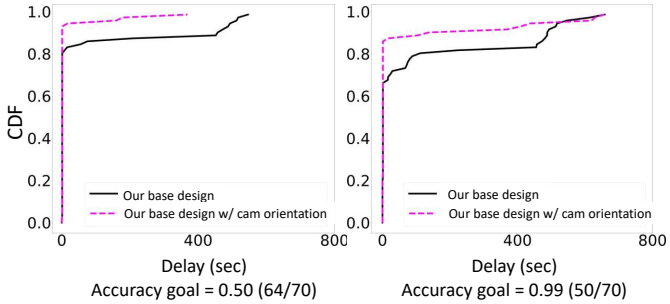
**Moderate sensitivity to input images** REV shows resilience to different input images from our dataset. First, we replace the randomly selected input images with another random batch selected from the dataset. As a result, REV sees an average of 0.04 difference in accuracy (stddev: 0.24); it sees delay differences of 7.7 seconds (11.7%) and 13.4 seconds (12.5%) for 0.50 and 0.99, respectively. Second, we test "easier" input images by including the camera that produced the input image in the query scope, while still excluding the input image from




(a) Accuracy goal = 0.50 (56/70)

(b) Accuracy goal = 0.99 (36/70)

Fig. 10: **Query delays when pre-processing $N$ cameras per geo-group at ingestion time,** showing pre-processing more than one camera per group results in diminishing return. Y-axis in logscale. (X/Y): X = # of queries on which all versions reach accuracy goal; Y = total query count. $N$ exceeding the total cameras of a geo-group: all the cameras are pre-processed




Accuracy goal = 0.50 (64/70)

Accuracy goal = 0.99 (50/70)

Fig. 11: **CDFs of query delays when REV picks starter cameras based on their orientations, showing the benefit**. (X/Y): X = # of queries reaching accuracy goal; Y = total query count.

the scope. As such, the query scope now has a camera with a viewpoint identical to the input image. REV a sees moderate benefit: 0.05 higher accuracy on average (std: 0.15), 23.0% and 12.1% reduction in delays, and 1% less processed videos on average. We attribute the results to our dataset characteristics: (1) Camera redundancy: given any input image, there are likely cameras offering similar viewpoints. (2) Decent image quality: were the input images in poorer quality, e.g., with large occlusion or low resolution, they may confuse the neural networks used in REV, resulting in lower accuracy overall.

**Low sensitivity to thresholds** We learn $P_{high}$ and $P_{low}$ via offline profiling of original video clips. As the thresholds determine when to pause sampling cameras, their values may affect query delays but not the eventual accuracy. With methods in (§V), we determine default values $P_{high}$=$1/d_{short}$=$1/0.73$ and $P_{low} = 1/d_{long} = 1/0.91$. We test REV by deviating from default values: $d_{short} \pm 0.1$ and $d_{long} \pm 0.1$. Across all 70 queries, the query delays only vary by less than 10% on average. The new thresholds increase delays on more than 90% of the queries (average increase: 2.4 seconds); and reduce delays on the remaining (average reduction: 7.2 seconds). Based on the minor variation, we conclude that the default parameters are adequate; the benefit from fine-tuning thresholds for individual queries is marginal.

### *F. Impact of optimizations*

**Processing at ingestion** Figure 10 shows the average query delay as a function of *N*, the number of starter cameras per geo-group pre-processed at ingestion time. We omit $N > 4$, where REV can extract all object features at ingestion time, leaving only feature matching to query time. Since feature matching is more than 1000× faster than feature extraction, a query's execution overhead becomes trivial.

The results support lightweight pre-processing at ingestion. (1) Pre-processing starter cameras reduces query delays substantially. Comparing to no pre-processing at all, pre-processing one starter camera per group reduces query delays by around 4×. (2) Pre-processing more than 1 starter camera per geo-group yields diminishing returns, no more than 25% delay reduction. (3) With the same pre-processing at ingestion time, REV delivers much lower delays than the alternatives.

**Picking starter cameras based on orientations** (Section VI-A) We estimate deployed camera orientations from the map in Figure 11. We use human-labeled viewpoints for input images of queries, minimizing inaccurate viewpoints. Overall, this optimization tends to benefit queries that used to be slow. On the queries used to have ≥70% percentile delays, REV sees on average 5.8× and 2× lower delays in reaching an accuracy of 0.50 and 0.99, respectively. For the remaining 70% queries, the delay reduction is negligible as most queries converge based on starter cameras. The reason is that significantly better viewpoints from manually picked starter cameras have little impact on current well-performed queries, but on those queries that used to suffers from bad indexes. Besides, by replacing starter cameras that used to have decent viewpoints, the initial rank of spatiotemporal cells typically does not have sharp changes, so the query delay will not differ (those <70% percentiles).

**Sampling cameras from complementary orientations** (Section VI-A) With camera orientations of the dataset, REV sees a 4.6% and 2.0% reduction in delays to reach an accuracy of 0.50 and 0.99, respectively. We identify two reasons for the minor benefit: co-located cameras are providing complementary viewpoints, and picking any of them is likely to help the search to similar degrees; the cameras with orientations opposed to the starter cameras can be inferior choices, e.g., capturing much fewer bounding boxes (and hence less likely the target object) than average cameras.

**Reusing states of previous queries** (Section VI-A) REV can effectively speed up a new query by reusing the intermediate state of previous queries. To show this, we test ten query pairs <$Q_{old}, Q_{new}$> on two accuracy goals of 0.50 and 0.99. The input images are randomly picked, and are different within each pair. Within each pair: we run $Q_{old}$ and terminates it once reaching the accuracy goal, and run $Q_{new}$ with the query state left from $Q_{old}$. Between pairs, we cleanse any query state. With $Q_{old}$ reaching an accuracy of 0.50 (i.e., a "brief" query), the delays for $Q_{new}$ to reach an accuracy of 0.50 and 0.99 are reduced by 86.2% and 76.8%, respectively. With $Q_{old}$ reaching

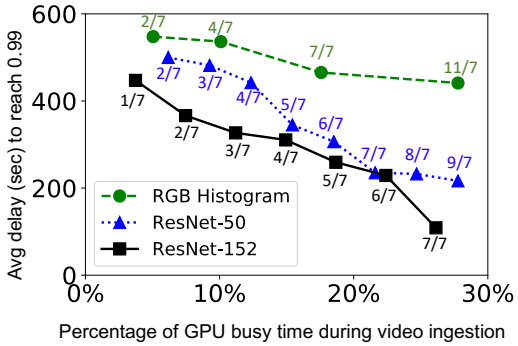



Fig. 12: The average query delay to reach an accuracy of 0.99 (Y-axis) with different resource budgets (percentage of GPU busy time as videos ingest) for ingestion-time processing (X-axis). The figure shows that using the most expensive operator at the ingestion is always beneficial. Annotation (A/B): A = # of starter cameras processed; B = the total # of geo-groups. A > B means more than 1 starter camera in some geo-groups.

an accuracy of 0.99 (i.e., a "thorough" query), the delays for $Q_{new}$ are reduced by 86.2% and 78.1%, respectively.

**Using cheap vision operators** (Section VI-B) We test two representative cheap vision operators: RGB histogram, a color filter; ResNet-50, a feature extractor less accurate than ResNet-152. Note that we avoid "local" operators such as SIFT [41] that extract regional features because local features are typically used for pairwise comparison of images and do not suit clustering. We evaluate the following designs.

- *Replacing the golden operator (ResNet-152).* The replacement is not beneficial. By replacing ResNet-152 with RGB histogram, REV's accuracy drops to nearly 0. By replacing ResNet-152 with ResNet-50, the accuracy drops by 3.2%, and the delays to reach an accuracy of 0.99 increased by 10%. The reason is that compared to ResNet-152, ResNet-50's cost reduction cannot compensate for its inferior ranking of cells which misguides the query execution.

- *For early ranking*. Using cheaper operators for ranking cells is ineffective. As shown in Figure 12, while more ingestion resources result in lower query delays in general, running the golden operator always results in lower query delays. This is because ResNet-152, despite running slower, ranks cells more accurately, which avoids processing cells with low promises.

- *For early filtering.* Using cheaper operators as early filters results in lower accuracy and longer delays. We make REV pre-cluster the features of RGB histogram, filter clusters that show low promise, and run ResNet-152 on the bounding box of the surviving clusters. The average query accuracy drops by 21%, and the delays to reach accuracy goals of 0.50 and 0.99 increase by 3.4× and 2.7×. This is because color histograms result in a poor rank of cells which misguides REV's query execution. The same experiment with ResNet-50 as the filter results in 12% decrease in accuracy and 2.7× and 2.5× longer delays to reach accuracy goals of 0.50 and 0.99. Note that REV cannot reuse the Resnet-50 features in computing the ResNet-152 features because they belong to different feature spaces.

## VIII. Related Work

**Optimizing video analytics** Besides Spatula [27], ViTrack [8], and VeTrac [60] discussed earlier, to reduce multi-camera inference cost, Caesar [39] encodes object activity correlation across cameras; Optasia [42] shares common work modules and parallelizes query plans; Jiang *et al.* [29] initiate an abstraction of camera clusters to enable resource and data sharing among cameras. There have been works optimizing video analytics with operator cascades [32], [16], [21], [56], and format tuning to trade accuracy for cost-efficiency [65], [25], [28], [67], [49]. Focus [21] saves cost by pre-processing videos with cheap NNs at ingestion. Notably, it clusters object features to avoid redundant comparisons with target objects. REV uses clustering in a different way: to smooth out transient disturbances for higher ReID accuracy. Extensive works exploit collaborations between cloud and edge [66], [53], [6], [37], [54]; cloud/edge and mobile devices [7], [9]; cloud and cameras [63]; edge and cameras [68], [36]; and edge and drones [61]. Elf [64] imposes energy planning for counting queries on resource-frugal cameras. None of above was designed for ReID over city-scale cameras.

**Information Retrieval** Recall-oriented retrievals, e.g., legal or patent search, is a group of tasks to find all relevant documents, and a bad rank typically incurs significant more efforts from domain experts [3], [5], [44]. As objects are rare in ReID tasks and typically requires domain knowledge in criminal investigation or smart traffic planning, we position REV to solve recall-oriented tasks, i.e., requiring all true cell to be retrieved, and adopt the metric of recall for evaluation.

**Vehicle Re-ID augmentations** Besides general features, recent works mine minor features like car make/model [58], number of doors/seats [40], and brands/tags in windows [18], requiring extra large amount of labeled high-resolution videos, and is not practical in urban surveillance systems.

## IX. Conclusion

We built REV, a video engine for object ReID across city-scale cameras. First, REV answers spatiotemporal queries on location and time of target vehicle occurrences. Second, REV approximates distinct objects by clustering unreliable object features emitted by ReID algorithms before matching with the input image. Third, to search in colossal video data, REV samples cameras to maximize the spatiotemporal coverage and incrementally processes additional cameras on demand. On 25 hours of city videos spanning 25 cameras, REV on average reached an accuracy of 0.87 and ran at 830× video realtime in achieving high accuracy.

## Acknowledgments

The authors were supported in part by NSF awards #2128725, #1919197, #2106893, and Virginia's Commonwealth Cyber Initiative. The authors thank the anonymous reviewers for their insightful feedback.

## REFERENCES


[1] Deep learning in video multi-object tracking: A survey. Neurocomputing, 381:61 – 88, 2020.

[2] Tamas Abraham and J. Roddick. Survey of spatio-temporal databases. GeoInformatica, 3:61–99, 1999.

[3] Avi Arampatzis, Jaap Kamps, and Stephen Robertson. Where to stop reading a ranked list? threshold optimization using truncated score distributions. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, page 524–531, New York, NY, USA, 2009. Association for Computing Machinery.

[4] Bissan Audeh, Philippe Beaune, and Michel Beigbeder. Recall-oriented evaluation for information retrieval systems. In Mihai Lupu, Evangelos Kanoulas, and Fernando Loizides, editors, Multidisciplinary Information Retrieval, pages 29–32, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[5] Dara Bahri, Yi Tay, Che Zheng, Donald Metzler, and Andrew Tomkins. Choppy: Cut transformer for ranked list truncation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, page 1513–1516, New York, NY, USA, 2020. Association for Computing Machinery.

[6] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G. Andersen, Michael Kaminsky, and Subramanya R. Dulloor. Scaling video analytics on constrained edge nodes. In Proceedings of the 2nd SysML Conference, 2019.

[7] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys '15, page 155–168, New York, NY, USA, 2015. Association for Computing Machinery.

[8] L. Cheng and J. Wang. Vitrack: Efficient tracking on the edge for commodity video surveillance systems. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pages 1052–1060, 2018.

[9] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. Cachier: Edge-caching for recognition applications. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 276–286, 2017.

[10] Martin Erwig, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. Geoinformatica, 3(3):269–296, September 1999.

[11] Congress for the New Urbanism. Street networks 101. https://www.cnu.org/our-projects/street-networks/street-networks-101, 2020.

[12] Forbes. The internet of things is creating 1984's national camera surveillance network. https://www.forbes.com/sites/kalevleetaru/2019/07/20/the-internet-of-things-is-creating-1984s-national-camera-surveillance-network/#9d301f523319, 2020.

[13] Yang Fu, Yunchao Wei, Guanshuo Wang, Yuqian Zhou, Honghui Shi, and Thomas S. Huang. Self-similarity grouping: A simple unsupervised cross domain adaptation approach for person re-identification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

[14] Yang Fu, Yunchao Wei, Yuqian Zhou, Honghui Shi, Gao Huang, Xinchao Wang, Zhiqiang Yao, and Thomas Huang. Horizontal pyramid matching for person re-identification. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01):8295–8302, Jul. 2019.

[15] M. Gou, S. Karanam, W. Liu, O. Camps, and R. J. Radke. Dukemtmc4reid: A large-scale multi-camera person re-identification dataset. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 1425–1434, July 2017.

[16] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '16, pages 123–136, New York, NY, USA, 2016. ACM.

[17] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. Visual road: A video data management benchmark. In SIGMOD, pages 972–987, 2019.

[18] Bing He, Jia Li, Yifan Zhao, and Yonghong Tian. Part-regularized near-duplicate vehicle re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[20] Zhiqun He, Yu Lei, Shuai Bai, and Wei Wu. Multi-camera vehicle tracking with powerful visual features and spatial-temporal cue. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 203–212, 2019.

[21] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B. Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, 2018. USENIX Association.

[22] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In CVPR, pages 7366–7375. IEEE Computer Society, 2018.

[23] Tsung-Wei Huang, Jiarui Cai, Hao Yang, Hung-Min Hsu, and Jenq-Neng Hwang. Multi-view vehicle re-identification using temporal attention model and metadata re-ranking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.

[24] James N. Hughes, Andrew Annex, Christopher N. Eichelberger, Anthony Fox, Andrew Hulbert, and Michael Ronquest. GeoMesa: a distributed architecture for spatio-temporal fusion. In Matthew F. Pellechia, Kannappan Palaniappan, Peter J. Doucette, Shiloh L. Dockstader, Gunasekaran Seetharaman, and Paul B. Deignan, editors, Geospatial Informatics, Fusion, and Motion Video Analytics V, volume 9473, pages 128 – 140. International Society for Optics and Photonics, SPIE, 2015.

[25] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodík, Leana Golubchik, Minlan Yu, Victor Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. October 2018.

[26] Samvit Jain, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, and Joseph E Gonzalez. Scaling video analytics systems to large camera deployments. arXiv preprint arXiv:1809.02318, 2018.

[27] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Victor Bahl, and Joseph Gonzalez. Spatula: Efficient cross-camera video analytics on large camera networks. In ACM/IEEE Symposium on Edge Computing (SEC 2020), November 2020.

[28] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18, pages 253–266, New York, NY, USA, 2018. ACM.

[29] Junchen Jiang, Yuhao Zhou, Ganesh Ananthanarayanan, Yuanchao Shu, and Andrew 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, HotEdgeVideo'19, page 1–7, New York, NY, USA, 2019. Association for Computing Machinery.

[30] Xin Jin and Jiawei Han. K-Means Clustering, pages 563–564. Springer US, Boston, MA, 2010.

[31] The Wall Street Journal. A world with a billion cameras watching you is just around the corner. https://www.wsj.com/articles/a-billion-surveillance-cameras-forecast-to-be-watching-within-two-years-11575565402, 2019.

[32] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. Noscope: Optimizing neural network queries over video at scale. Proc. VLDB Endow., 10(11):1586–1597, August 2017.

[33] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. 24(7), 2002.

[34] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. ICCVW '13, page 554–561, USA, 2013. IEEE Computer Society.

[35] Xiying Li and Zhihao Zhou. Object Re-Identification Based on Deep Learning. 06 2019.

[36] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi 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, SIGCOMM '20, page 359–376, New York, NY, USA, 2020. Association for Computing Machinery.

[37] Peng Liu, Bozhao Qi, and Suman Banerjee. Edgeeye: An edge service framework for real-time intelligent video analytics. In Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking, EdgeSys'18, pages 1–6, New York, NY, USA, 2018. ACM.

[38] X. Liu, W. Liu, T. Mei, and H. Ma. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. IEEE Transactions on Multimedia, 20(3):645–658, 2018.

[39] Xiaochen Liu, Pradipta Ghosh, Oytun Ulutan, B. S. Manjunath, Kevin Chan, and Ramesh Govindan. Caesar: Cross-camera complex activity recognition. In Proceedings of the 17th Conference on Embedded Networked Sensor Systems, SenSys '19, page 232–244, New York, NY, USA, 2019. Association for Computing Machinery.

[40] Xinchen Liu, Wu Liu, Tao Mei, and Huadong Ma. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 869–884, Cham, 2016. Springer International Publishing.

[41] David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004.

[42] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. Optasia: A relational platform for efficient large-scale video analytics. In Proceedings of the Seventh ACM Symposium on Cloud Computing, SoCC '16, page 57–70, New York, NY, USA, 2016. Association for Computing Machinery.

[43] Kai Lv, Heming Du, Yunzhong Hou, Weijian Deng, Hao Sheng, Jianbin Jiao, and Liang Zheng. Vehicle re-identification with location and time stamps. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.

[44] Walid Magdy and Gareth J.F. Jones. Pres: A score metric for evaluating recall-oriented information retrieval applications. In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, page 611–618, New York, NY, USA, 2010. Association for Computing Machinery.

[45] Wolfram MathWorld. $l^2$-norm. https://mathworld.wolfram.com/L2-Norm.html, 2020.

[46] Microsoft. Video analytics towards vision zero, 2019.

[47] Milind Naphade, Rama Chellappa, David Anastasiu, Anuj Sharma, Ming-Ching Chang, Xiaodong Yang, Shuo Wang, Zheng Tang, and Liang Zheng. Ai city challenge, 2020.

[48] OpenCV. Histogram calculation. docs.opencv.org.

[49] Chrisma Pakha, Aakanksha Chowdhery, and Junchen Jiang. Reinventing video streaming for distributed vision analytics. In 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18), Boston, MA, 2018. USENIX Association.

[50] Neelabh Pant, Mohammadhani Fouladgar, Ramez Elmasri, and Kulsawasd Jitkajornwanich. A survey of spatio-temporal database research. In Ngoc Thanh Nguyen, Duong Hung Hoang, Tzung-Pei Hong, Hoang Pham, and Bogdan Trawiński, editors, Intelligent Information and Database Systems, pages 115–126, Cham, 2018. Springer International Publishing.

[51] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V. Le. Meta pseudo labels, 2021.

[52] Kriengkrai Porkaew, Iosif Lazaridis, and Sharad Mehrotra. Querying mobile objects in spatio-temporal databases. In Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras, editors, Advances in Spatial and Temporal Databases, pages 59–78, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[53] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. Deepdecision: A mobile deep learning framework for edge video analytics. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pages 1421–1429, April 2018.

[54] Arun Ravindran and Anjus George. An edge datastore architecture for latency-critical distributed machine vision applications. In USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18), Boston, MA, July 2018. USENIX Association.

[55] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. CoRR, abs/1804.02767, 2018.

[56] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[57] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[58] Xiao Tan, Zhigang Wang, Minyue Jiang, Xipeng Yang, Jian Wang, Yuan Gao, Xiangbo Su, Xiaoqing Ye, Yuchen Yuan, Dongliang He, Shilei Wen, and Errui Ding. Multi-camera vehicle tracking and re-identification based on visual and spatial-temporal features. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019.

[59] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[60] Panrong Tong, Mingqian Li Li, Mo Li, Jianqiang Huang, and Xiansheng Hua. Large-scale vehicle trajectory reconstruction with camera sensing network. In Proceedings of the 27th Annual International Conference on Mobile Computing and Networking, MobiCom '21, 2021.

[61] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In 2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018, pages 159–173, 2018.

[62] Paper with code. Person re-identification on dukemtmc-reid, 2020.

[63] Mengwei Xu, Tiantu Xu, Yunxin Liu, Xuanzhe Liu, Gang Huang, and Felix Xiaozhu Lin. Supporting video queries on zero-streaming cameras. CoRR, abs/1904.12342, 2019.

[64] Mengwei Xu, Xiwen Zhang, Yunxin Liu, Gang Huang, Xuanzhe Liu, and Felix Xiaozhu Lin. Approximate query service on autonomous iot cameras. In Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, MobiSys '20, page 191–205, New York, NY, USA, 2020. Association for Computing Machinery.

[65] Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. Vstore: A data store for analytics on large videos. In Proceedings of the Fourteenth EuroSys Conference 2019, EuroSys '19, pages 16:1–16:17, New York, NY, USA, 2019. ACM.

[66] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. Lavea: Latency-aware video analytics on edge computing platform. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pages 2573–2574, June 2017.

[67] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pages 377–392, Boston, MA, 2017. USENIX Association.

[68] Tan Zhang, Aakanksha Chowdhery, Paramvir (Victor) Bahl, Kyle Jamieson, and Suman Banerjee. The design and implementation of a wireless video surveillance system. In Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, MobiCom '15, pages 426–438, New York, NY, USA, 2015. ACM.

[69] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1116–1124, 2015.

[70] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1116–1124, 2015.

[71] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. Person re-identification: Past, present and future. CoRR, abs/1610.02984, 2016.

[72] Z. Zhong, L. Zheng, D. Cao, and S. Li. Re-ranking person re-identification with k-reciprocal encoding. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3652–3661, 2017.