This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2024.3459409

Argus: Enabling Cross-Camera Collaboration for Video Analytics on Distributed Smart Cameras

Juheon Yi, Utku Günay Acer, Fahim Kawsar, and Chulhong Min

Abstract—Overlapping cameras offer exciting opportunities to view a scene from different angles, allowing for more advanced, comprehensive and robust analysis. However, existing video analytics systems for multi-camera streams are mostly limited to (i) per-camera processing and aggregation and (ii) workload-agnostic centralized processing architectures. In this paper, we present Argus, a distributed video analytics system with *cross-camera collaboration* on smart cameras. We identify multi-camera, multi-target tracking as the primary task of multi-camera video analytics and develop a novel technique that avoids redundant, processing-heavy identification tasks by leveraging object-wise spatio-temporal association in the overlapping fields of view across multiple cameras. We further develop a set of techniques to perform these operations across distributed cameras without cloud support at low latency by (i) dynamically ordering the camera and object inspection sequence and (ii) flexibly distributing the workload across smart cameras, taking into account network transmission and heterogeneous computational capacities. Evaluation of three real-world overlapping camera datasets with two Nvidia Jetson devices shows that Argus reduces the number of object identifications and end-to-end latency by up to 7.13× and 2.19× (4.86× and 1.60× compared to the state-of-the-art), while achieving comparable tracking quality.

Index Terms—Cross-camera collaboration, Smart cameras, Video analytics

1 Introduction

It is increasingly common for physical locations to be surrounded and monitored by multiple cameras with overlapping fields of view (hereinafter 'overlapping cameras'), e.g., intersections, shopping malls, public transport, construction sites and airports, as shown in Fig. 1. Such multiple overlapping cameras offer exciting opportunities to observe a scene from different angles, enabling enriched, comprehensive and robust analysis. For example, our analysis of the CityFlowV2 dataset [4] (5 cameras deployed to monitor vehicles on the road intersection) shows that each camera separately detects only 3.7 vehicles per frame on average, while five cameras detect a total of 12.0 vehicles altogether. Since a target vehicle can be captured by multiple cameras from different distances and angles, we can also observe objects of interest with a holistic view. Such view diversity can make the analytics more enriched and robust, e.g., a vehicle's license plate may be occluded in one camera's view due to its position or occlusion, but not in the other cameras.

Most visual analytics systems are deployed in cloud environments. On the other hand, *on-camera video analytics* offer various attractive benefits such as immediate response, increased reliability and privacy protection. We envision that on-board AI accelerators [5], [6], [7] (e.g., Nvidia Jetson [8], [9], Google Coral TPU [10] and Analog MAX78000 [11]) and embedded AI models [12], [13] will accelerate this trend. However, the current practice of multi-camera stream processing is limited to being deployed on cameras without relying on cloud servers in two ways. (i) *Per-camera processing and aggregation.* Previous work has mostly focused on processing the video analytics pipeline on each camera individually and aggregating the results at the final stage [14], [15], [16], thereby suffering from significant processing redundancy and latency. (ii) *Workload-agnostic centralized processing.* Some systems have been proposed to handle multi-video streams, but they mostly assume that the videos are streamed to the central node, e.g., cloud servers, and focus on optimization and coordination of the serving engine (e.g., GPU scheduling and batch processing [17], [18], [19]).

Fig. 1: Places with overlapping cameras: intersection, shopping mall, transport, construction site.

In this paper, we present Argus, a distributed video analytics system designed for *cross-camera collaboration* with overlapping cameras. Here, the term 'cross-camera collaboration' not only encompasses the fusion of multi-view images for video analytics, but also refers to the cooperative utilization of distributed resources to ensure video analytics with high accuracy and low latency on distributed smart cameras, eliminating the need for a cloud server. To this end, we identify that *multi-camera, multi-target tracking* serves as a fundamental task for multi-camera video analytics. This process involves determining the location and capturing image crops of target objects (presented as query images)

- *Juheon Yi is with Nokia Bell Labs (e-mail: juheon.yi@nokia-bell-labs.com).*
- *Utku Günay Acer is with Nokia Bell Labs (e-mail: utku_gunay.acer@nokia-bell-labs.com).*
- *Fahim Kawsar is with Nokia Bell Labs (e-mail: fahim.kawsar@nokia-bell-labs.com).*
- *Chulhong Min, the corresponding author, is with Nokia Bell Labs (e-mail: chulhong.min@nokia-bell-labs.com).*

TABLE 1: Comparison of cross-camera collaboration approach in Argus with REV [1], Spatula [2], and CrossRoI [3].

	REV [1]	**Spatula** [2]	**CrossRoI** [3]	**Argus (Ours)**
Target environment	Overlapping cameras	Non-overlapping cameras	Overlapping cameras	Overlapping cameras
Optimization goal	On-server computation costs	Communication and on-server computation costs	Communication and on-server computation costs	End-to-end latency on cameras
Collaboration granularity	Cells (group of cameras)	Cameras	Areas (RoIs)	Objects
Applying association	Dynamic (depending on the target's existence)	Dynamic (depending on the target's existence)	Static (once when the cameras are deployed)	Dynamic (depending on the target's location)
Approach	Incrementally search cells that have lowest identification confidence	Identify the subset of cameras that capture target objects	Find the smallest RoI that contains the target objects	Minimise the # of identification operations across cameras
Video processing	Centralized	Centralized	Centralized	Distributed

on deployed cameras over time. We find that the computational bottleneck for camera collaboration arises due to the frequent execution of identification model inference across different cameras. To address this challenge, we develop a fine-grained, object-wise spatio-temporal association technique. This novel approach strategically avoids redundant identification tasks on both spatial (across multiple cameras) and temporal (within each camera over time) axes. This not only streamlines the process but also enhances the efficiency.

To enable effective multi-camera, multi-target tracking across overlapping cameras, we develop an object-wise association-aware identification technique. Specifically, Argus continuously tracks records of the association of objects (their bounding boxes) with the same identity across both multiple cameras (§4.1) and time (§4.2). Then, it identifies the object by matching the location association instead of running the identification model inference and matching the appearance feature. The concept of spatio-temporal association has been proposed in several previous works to reduce the repetitive appearance or query irrelevant areas [2], [3], [20]. However, they apply to association at a *coarse-grained* level, e.g., groups of cameras [1], cameras [2], [20] or regions of interest (RoIs) [3]. Thus, the expected gain is small for our target environment, which is multi-camera, multi-target tracking on overlapping cameras. For example, the resource saving from camera-wise association and filtering [2], [20] is expected to be marginal for densely deployed overlapping cameras. RoI-wise association and filtering [3] also degrade tracking accuracy, as the target object is not detected on a subset of cameras (refer to Table 1 and §7 for more details). In §2.2 and §6.2, we also provide an in-depth analysis and a comparative study with these prior arts, respectively. Furthermore, we carefully incorporate techniques to handle corner cases in the association process (e.g., newly appearing objects, occasional failure of the identification model and its error propagation) and improve the robustness of the spatio/temporal association process (§5.3).

Next, we develop a set of strategies that perform spatio-temporal association over distributed smart cameras at low latency. To maximize the benefits of association-aware identification, it needs to process cameras one by one in a sequential manner so that the number of identification model inferences is minimized; identification model inference needs to be performed when the identity of the pivot object is not yet known. This would lead to an increase in end-to-end latency, even with the fewer number of identification model inferences. Also, since cameras have different workloads (i.e., the number of detected objects) and heterogeneous processing capabilities, careless scheduling and distribution might not maximize the overall performance. To this end, we develop a multi-camera dynamic inspector (§5.1) that dynamically orders the camera and bounding box inspection sequence to avoid identification tasks for query-irrelevant objects. We also distribute identification tasks across multiple cameras, taking into account network transmission and heterogeneous computing capacities on the fly, to minimize end-to-end latency (§5).

We prototype Argus on two Nvidia Jetson devices (AGX [8] and NX [9]) and evaluate its performance with three real-world overlapping camera datasets (CityFlowV2 [4], CAMPUS [21], and MMPTRACK [22]). The results show that Argus reduces the number of identification model executions and the end-to-end latency by up to 7.13× and 2.19× compared to the conventional per-camera processing pipeline (4.85× and 1.60× compared to the state-of-the-art spatio-temporal association), while achieving comparable tracking quality.

We summarize the contribution of this paper as follows.

- We present Argus, a novel system for robust and low-latency multi-camera video analytics with cross-camera collaboration on distributed smart cameras.
- To enable efficient cross-camera collaboration, we develop a novel fine-grained object-wise spatio-temporal association which exploits FoV overlaps across cameras to optimise redundancy in multi-camera, multi-target tracking.
- We introduce a distributed scheduling technique that dynamically schedules the inspection sequence and distributes the processing workload across multiple smart cameras. This approach optimizes end-to-end latency and ensures efficient utilization of resources on each camera.
- Extensive evaluations over three overlapping camera datasets show that Argus significantly reduces the number of identification model executions and end-to-end latency by up to 7.13× and 2.19× (4.86× and 1.60× compared to the state-of-the-art [1], [2], [3]) while achieving comparable tracking quality to baselines.

2 Background and Motivation

2.1 Multi-Camera, Multi-Target Tracking

In this work, we focus on multi-camera, multi-tracking using deep learning-based object detection and re-identification models. These models robustly track objects

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2024.3459409

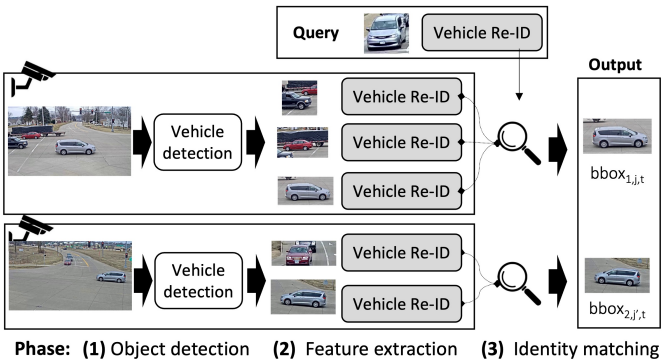


Fig. 2: Typical pipeline (example with vehicle) for multi-camera, multi-target tracking without cross-camera collaboration. Each camera runs the detection and identification independently and aggregates the output at the final stage.

TABLE 2: Identification latency on Jetson devices.

	Vehicle (ResNet-101) [26]			Person (ResNet-50) [27]		
Batch size	1	2	4	1	2	4
NX	0.119s	0.206s	0.399s	0.043s	0.045s	0.066s
AGX	0.065s	0.121s	0.217s	0.018s	0.020s	0.028s

Fig. 3: CityFlowV2's camera topology.

Fig. 4: The same vehicle captured from multiple views.

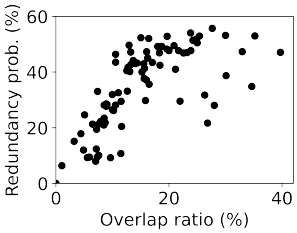


Fig. 5: Identification saving opportunity for different overlapping ratios in in CityFlowV2.

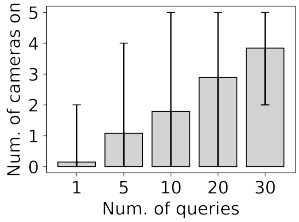


Fig. 6: Number of cameras after filtering by Spatula [2] in CityFlowV2 (5 cameras).

across multiple views even in complex scenarios, by leveraging the discriminative power of deep neural networks. They also handle occlusions, changes in appearance, and other challenges that are difficult to address with geometry-based methods. To this end, they often learn from large-scale datasets, enabling them to generalize to a wide range of scenarios and adapt to changes in the environment.

Operational flow. The key to enabling video analytics on overlapping cameras is *multi-camera, multi-target tracking*: detecting and tracking target objects (given as query images) from video streams captured by multiple cameras. This is typically achieved in three stages, as shown in Fig. 2. (i) The *object detection* stage detects the bounding boxes of objects in one frame on each camera using object detectors (e.g., YOLO [23]) or background subtraction techniques [24], [25]. (ii) The *per-camera object identification* stage extracts the appearance features of the detected objects by running the object identification (ID) model (e.g., [26]) and determines whether it matches the query image based on feature similarity (e.g., L2 distance, cosine similarity). (iii) The *result aggregation* stage aggregates the identification results across multiple cameras and generates tracklets [14] that can be used for further processing for application logic, e.g., object counting, license plate extraction and face recognition.

Compute bottleneck: per-object identification. The main compute bottleneck is the execution of identification tasks, which need to be performed for all detected objects in every frame across multiple cameras to determine identity of objects, as shown in Fig. 2. Although we envision smart cameras equipped with built-in AI accelerators, they are not yet capable of processing a number of identification tasks in real time. Table 2 shows the latency of two identification models (ResNet-101-based vehicle identification [26] and ResNet-50-based person identification [27]) with different batch sizes over two Nvidia Jetson devices. It shows that the number of identification model executions to run on one camera is quite limited. For example, if 4 vehicles are detected on every frame on average, even the powerful Jetson AGX platform can only process about 4 frames per second. The throughput would drop even further if object detection is included (we show the detailed results in §6).

2.2 Exploring Optimisation Opportunities

Redundant identification of the same objects. To explore the opportunities for optimizing the pipeline for multi-camera, multi-target tracking, we investigate the pattern of identification tasks with the CityFlowV2 dataset [4]; five cameras are installed at an intersection as shown in Fig. 3. Fig. 5 shows the redundancy probability, i.e., the probability of objects appearing simultaneously in multiple cameras for different overlap ratios; the overlap ratio is defined as the ratio of the time the object appears simultaneously in both cameras and the total time it is detected in any camera; for a target appearing in n cameras, we calculate all pairwise overlap ratios (nC_2 camera pairs) and take the average. Each point represents a different query. The results show that, as the overlap ratio increases, the probability of an object's appearance in multiple cameras also becomes higher. This means that a dense array of cameras with overlapping FoVs will have more redundant identification tasks for the same object across multiple cameras.

Spatio-temporal association. To avoid unnecessary and redundant identification tasks, we adopt *spatio-temporal association* of objects, which have been proposed in the auto-calibration techniques [28], [29] for multi-view tracking systems. Spatial-temporal association refers to the geographical and temporal association of an object to different cameras. More specifically, we associate the identity of an object across multiple cameras by matching their correlated positions on the frame, rather than matching appearance features extracted from the identification model, as shown in Fig. 2. This intuition arises from the observation that, once

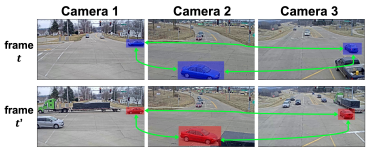


Fig. 7: Spatial association (green lines).

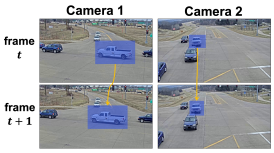
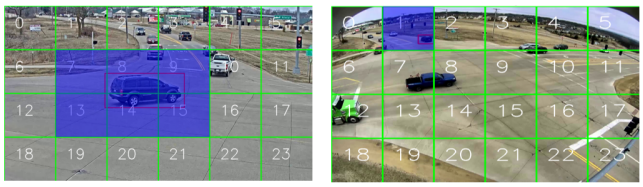


Fig. 8: Temporal association (yellow lines).



(a) Camera #1 (RoI: 6 grids). (b) Camera #2 (RoI: 1 grid).

Fig. 9: Overlapping RoI example between 2 cameras. CrossRoI [3] favours Camera #2 which has the smaller RoI size.

installed in a place, cameras' FoVs are fixed over time. We explain spatial association with an example. If the bounding box of two objects (at different times) is located at the same position in one camera's FoV, the position of their bounding boxes in other cameras will also remain the same.[1] Fig. 7 shows the spatial association obtained from the CityFlowV2 dataset [4]. Each row shows a list of images captured by three cameras (Camera 1, 2, 3) installed as in Fig. 3 at the same time. Each column shows the images taken by the same camera. The red and blue overlay boxes in each row represent the bounding box of a red vehicle and a blue vehicle, respectively. Although two vehicles crossed the intersection at different times, when two vehicles are located at a similar location on Camera 1, we can observe that the corresponding bounding boxes remain in a similar position on the other cameras. Similarly, as shown in Fig. 8, we expect the *temporal association* of an object: an object in a video stream remains in proximity within successive frames.

2.3 Limitations of Prior Works

Auto-calibration using spatio/temporal association. Auto-calibration, also known as self-calibration, has been proposed as a solution to enable multi-view tracking systems from 1990s. This technique aims to automatically estimate the camera parameters, such as intrinsic and extrinsic parameters for object tracking in multiple camera views, without the need for manual intervention or specialized calibration objects [28]. Auto-calibration methods leverage the spatio-temporal correlation of objects in multiple views as described in §2.2, taking advantage of the geometric constraints imposed by the scene and the motion of objects or the camera itself [30]. By harnessing these constraints, auto-calibration techniques can iteratively refine the camera parameters, leading to improved tracking accuracy and robustness [31]. Several auto-calibration methods have been proposed in the literature, including the self-calibration of space and time technique [29], which exploits the correlation between space and time in the image sequence to estimate the camera parameters. Other approaches [32], [33] utilize the epipolar geometry and geometric constraints to estimate the camera parameters. Additionally, the establishment of a common coordinate frame across multiple views has been proposed to improve tracking performance [34]. However, these techniques face several limitations, especially when compared to modern approaches that utilize deep learning-based object detection and identification models. Auto-calibration methods are typically based on geometric constraints, which can be sensitive to errors in feature detection and correspondence matching, leading to inaccurate camera parameter estimation. Also, these methods rely on the assumption of a static scene, which may not hold true in dynamic environments where objects and people are constantly moving and changing [29].

Camera-wise filtering in non-overlapping cameras. Spatula [2] leverages cross-camera correlation to identify a subset of cameras likely to contain the target objects and filter out unnecessary cameras (that do not contain the target objects). While it shows a significant performance benefit in its target environment (widely deployed *non-overlapping* cameras), it fails to effectively reduce redundant identification operations in *overlapping* cameras. To quantify its benefit, we analyzed the CityFlowV2 dataset [4]. Fig. 6 shows the average number of cameras out of five cameras, used by Spatula; the error bar indicates the minimum/maximum number of cameras. The results show that the benefit of Spatula-based camera-wise filtering quickly diminishes when more queries are used, i.e., fewer cameras are filtered out. This is because a higher number of objects are likely to be captured by a higher number of cameras, simultaneously.

Camera-wise filtering in overlapping cameras. REV [1] leverages spatial correlation across multiple overlapping cameras to minimize the number of processed cameras in identifying the target object. However, its goal is to *confirm the presence* of the target object within a given timestamp. As such, it cannot be applied for Argus, which not only aims to *confirm the presence* of a target object but also *extract the image crops of the target* from all cameras that capture it. Specifically, REV employs an incremental approach, starting its search from the camera that detects the most number of objects[2] and stops the search once the target is identified. Thus, it often misses the target image crops in remaining cameras, which may have been captured in superior quality.

RoI-wise filtering in overlapping cameras. CrossRoI [3] leverages spatio-temporal correlation to optimize the region of interest (RoI) of multiple video streams from overlapping

1. Of course, this argument is not always right in theory. Since a camera projects 3D space onto the 2D plane, the same bounding box of one camera at different times does not guarantee the same position of an object. The simplest case would be when two objects of different sizes are located in the same direction from the camera but a smaller object is located closer to the camera. However, in practice, such cases are very rare because the camera is often installed to look at a 2D plane (e.g., street and floor) obliquely to cover a wide area and objects of interest (e.g., vehicles and people) cannot be located at arbitrary 3D positions, as shown in Fig. 7. Also, even when such a case happens (e.g., two objects at different positions are captured in the same bounding box in Camera 1), the spatial association of two objects is not made because the position and size of the bounding boxes in other cameras (e.g., Camera 2 and 3) will be different.

2. The underlying rationale is that cameras with more bounding boxes are more likely to capture the target object.

cameras. When multiple objects are captured by a set of cameras from different views, CrossRoI extracts the smallest possible total RoI across all cameras in which all target objects appear at least once, and then reduces processing and transmission costs by filtering out unmasked RoI areas, i.e., (a) redundant appearances and (b) areas that do not contain the target objects; RoI is defined as a 6-by-4 grid. While it effectively reduces the workload to be processed, it is not suitable for multi-camera, multi-task tracking. Since it aims to minimize the RoI size that covers the overlapping FoVs, the smaller RoI that contains the object is preferred (e.g., 1 grid in Camera 2 instead of 6 grids in Camera 1 in Fig. 9). This would lead to considerable degradation of the accuracy. Furthermore, since it filters out redundant appearance in the initial stage, analytics applications cannot benefit from a holistic view, as shown in Fig. 4.

3 Argus Design

3.1 Design Goals

Low-latency and high accuracy. We aim at achieving both low latency and high accuracy in running multi-camera, multi-target tracking across overlapping cameras, which is the key requirement of various video analytics apps.

On-device processing on distributed smart cameras: Streaming videos to a cloud server for processing incurs significant networking and computing costs as well as privacy issues. We aim to run the video analytics pipeline with cross-camera collaboration fully on distributed smart cameras leveraging on-device resources.

Flexibility of tracking pipeline. We treat the AI models as a black box, thereby supporting both open-source and proprietary models and allowing analytics app developers to select the models for the purpose flexibly.

3.2 Approach

Multi-camera object-wise spatio-temporal association. Our preliminary study reveals that the computational bottleneck for multi-camera, multi-target tracking in overlapping cameras is the redundant identification of the same object (§2.1). To achieve both resource-efficient and accurate tracking, we devise a method for *object-wise association-aware identification*. As shown in Fig. 7 and Fig. 8, Argus associates the spatio-temporal correlation of *objects*' positions and identifies redundant *identification* tasks. It reduces on-camera computational costs by filtering out redundant identification tasks for the same object across multiple cameras (spatially) and over time (temporally).

On-camera distributed processing. To enable on-camera processing, we further devise two optimization techniques. First, we optimize on-camera workload by minimizing the number of model executions (both object detection and identification). By inspecting cameras and objects (bounding boxes) in order of probability of containing the target object, Argus avoids model executions that are irrelevant to the target objects; note that the tracking operation is finished when all target objects are found. Second, we further optimize end-to-end latency with parallel execution on distributed cameras. More specifically, Argus distributes the identification workload across multiple cameras on the fly and executes it in parallel.

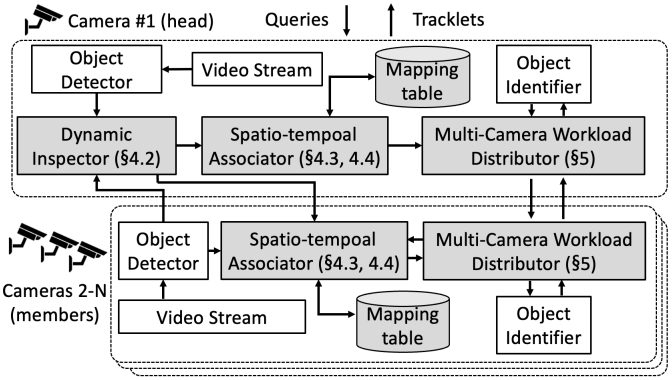


Fig. 10: Argus system architecture.

3.3 System Architecture

Fig. 10 shows the system architecture of Argus. It takes the query images as input from analytics apps and provides the tracklets (list of cropped images and bounding boxes of the detected objects) tracked from multiple cameras as output. Given the targets to track, Argus first starts by running the object detector to detect objects for identification on each frame in parallel. Afterwards, the head camera runs the Dynamic Inspector (§5.1) to determine the processing order of cameras and bounding boxes. Once the order is determined, the member cameras run the Spatio-temporal Associator (§4.1 and §4.2) opportunistically skips the inference by leveraging the spatio-temporal correlations across cameras. For the remaining unassociated bounding boxes, Multi-Camera Workload Distributor (§5) schedules the identification tasks across cameras considering the network transmission latency and heterogeneous compute capabilities.

3.4 Problem Formulation and Operational Flow

Formulation. We formalize our problem setting as follows:

- C: a set of cameras, where C^i is i^{th} camera,
- F_t: a set of image frames at time t, where F_t^i is an image frame from C^i at time t.
- E_Q: a set of id feature embedding of query images, where E_j is the feature embedding of jth query
- $bbox_{t,j}^i$: bounding boxes of the detected on C^i at time t. C^i has n_t^i objects detected at time t ($j = 1, 2, ..., n_t^i$).

Formally, the goal of multi-camera spatio-temporal association is to minimize the total number of identification operations across all cameras,

$$\min \sum_i n_{IDs}^i, \tag{1}$$

where n_{IDs}^i is number of identification operations on C^i.

Operational flow. Fig. 11 shows the operational flow of Argus's spatio-temporal association-enabled multi-camera multi-object tracking. For simplicity, we first explain the procedure for a single query.

1) **Object detection**: For each frame t, all cameras run object detection in parallel. We denote the output of the i-th

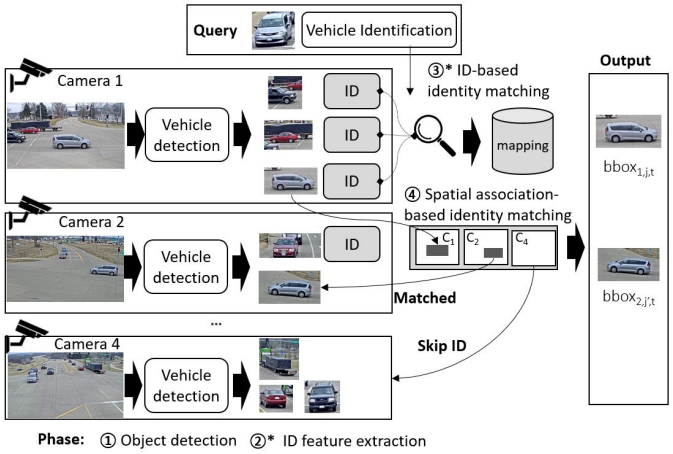


Fig. 11: Operational flow of our multi-camera spatio-temporal association. (*): operations are performed on a subset of cameras and RoIs.

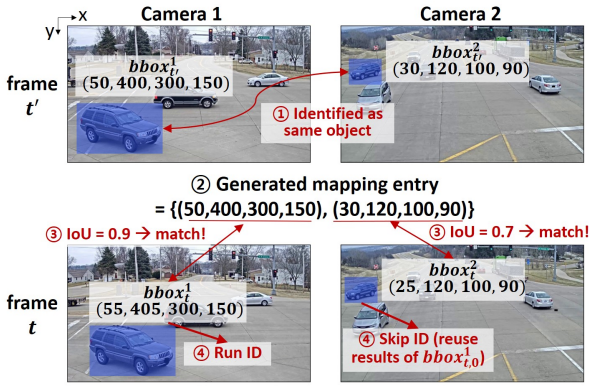


Fig. 12: Example operation of spatial association.

camera C^i as $\{bbox^i_{t,j}\}$, where $bbox^i_{t,j}$ is the bounding box coordinate for the j-th detected object with the same class label (e.g., car, person) as the query.

2) **Multi-camera dynamic inspection (§5.1).** After detection, we decide the camera and bounding box inspection order to maximize the efficiency and accuracy of spatio-temporal association. Our key insight is that prioritizing the inspection of the camera and bounding box that is most likely to capture the target at a highest quality maximizes the ID inference skipping and assures the highest association accuracy. Once the order is determined, we repeat the below steps for each camera.
3) **Spatial association (§4.1).** For each camera, we first skip ID inferences on bounding boxes that are spatially associated with previous cameras' tracking results in frame t. This is done by leveraging a *mapping table*, where each entry in a mapping table is a list of bounding boxes across cameras that were identified as the same object at the same timestamp. Fig. 12 shows an example. Assume we detected two bounding boxes in Cameras 1,2 at frame t: $bbox^1_t$ and $bbox^2_t$, and there is a mapping entry created from the past frame t': $\{bbox^1_{t'}, bbox^2_{t'}\}$. If the bounding box area overlaps for both pairs ($bbox^1_{t'}$, $bbox^1_t$) and ($bbox^2_{t'}$, $bbox^2_t$) are over a threshold, we can determine that $bbox^1_t$ and $bbox^2_t$ correspond to the same object. Thus, it suffices to only run the ID inference on $bbox^1_t$, and reuse the result for $bbox^2_t$ and skip inference.
4) **Temporal association (§4.2).** We further skip ID inferences on bounding boxes that are temporally associated with the same camera's tracking results in frame $t-1$. This is done by leveraging the temporal locality of objects; we match bounding boxes in frame t with the ones in frame $t-1$, and reuse the ID results if a match is found.
5) **ID feature extraction with distributed processing (§5.2).** Based on the bounding box inspection order, we execute the identification model and obtain the ID features for each cropped image, $\{E^i_{t,j}\}$, where $E^i_{t,j}$ is the ID appearance feature from the cropped image from bounding box $bbox^i_{t,j}$. The identification inference workload is run in parallel over distributed cameras considering their processing powers and network bandwidths.
6) **Identity matching.** We determine existence of query by computing the similarities of the extracted ID features and the query image's ID feature, E_Q. The identity matching results (bounding box coordinates and matching identity) are sent to the next camera so as to be utilized for spatial association (step 3).
7) **Aggregation.** Steps 3–7 repeats until all cameras are processed. When all cameras finish processing, the head camera aggregates the results, updates the mapping table and synchronises it with the member camera, and start from step 1 again for the next frame.

When multiple queries are given, the outputs from object detection (step 1) and ID feature extractions (step 6) are shared. Specific implementation details (e.g., bounding box and identity matching) are detailed in §4.3.

4 Spatio-Temporal Association

In this section, we describe the key ideas of our object-level spatio-temporal association (§4.1 and §4.2), and how the specific association techniques are implemented (§4.3).

4.1 Spatial Association

We first explain how we define a object-wise spatial association across multiple cameras. Once an object with the same identity is captured by multiple cameras, we create a mapping entry that contains a timestamp and a list of the corresponding bounding boxes on each camera in C. We use bounding boxes as location identifiers for fine-grained matching of the spatial association. Formally, we define a mapping entry as $entry^j = \{entry\_bbox^i_j\}$, where $entry\_bbox^i_j$ is a pair of coordinates referring to the south-western and northeastern corner of the box in C^i at the jth mapping entry. $entry\_bbox^i_j$ is set to *N/A* if the object is not found in the corresponding camera, C^i.

Utilizing the spatial association. In subsequent time intervals, we apply the identification model to the detected objects in a single camera (refer to §5.1 for determining the order of camera inspection). Upon identifying an object, we search for a mapping entry matching a bounding box of the identified object in the same camera. If such an entry is found, we examine the detected bounding boxes on the remaining cameras whose entry value is not *N/A*. If a bounding box in the remaining camera matches the located entry, we associate (i.e., reuse) the identification result from

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2024.3459409

the first camera, avoiding the need to rerun the identification model. Note that searching for a matching entry in the mapping table involves calculating the bounding box overlap, which is an extremely lightweight operation (e.g., takes <1 ms for 1,000 matches) as detailed in §4.3.

Management of the spatial association. We use bounding boxes as location identifiers for fine-grained matching of the spatial association. To facilitate quick access, we maintain the entries as a hash table. Also, if the number of entries exceeds a threshold (e.g., 100), Argus filters out duplicate or closely located entries by running non-maximum suppression on the bounding boxes of the entries. Specifically, when two entries have bounding boxes from the same cameras with significant overlap, we retain only the entry that has (i) a higher number of non-*N/A* values and (ii) a higher average identity matching score; implementation details are provided in §4.3. These mapping entries can be obtained during the offline phase with pre-recorded video clips or updated during the online phase with runtime results.

4.2 Temporal Association

We leverage temporal association to further reduce the number of identification operations. It is inspired by the observation proposed in simple online tracking methods [35], [36], that the location of an object does not change significantly within a short period of time. That is, the bounding box of an object in a video stream would remain in proximity to the bounding box with the same identity in the previous frame. For example, even in the vehicle tracking scenario in CityFlowV2 [4], the distance of a vehicle moving at a speed of 60 km/h in successive frames of a video stream at 10 Hz is about 1.7 metres, which is relatively small compared to the size of the area that a security camera usually covers.

When ID feature extraction is performed, Argus caches the ID features with their bounding box. Then, when the ID feature is needed for a new bounding box in a later frame, Argus finds the matching bounding box in the cache; we explain the implementation details of bounding box matching in §4.3. When the matching bounding box is found, Argus reuses its ID feature and updates the bounding box in the cache. We set the expiry time to one frame, i.e., the cache expires in the next frame unless it is updated.

4.3 Implementation of Association Techniques

RoI extraction. There are several options for the RoI extraction stage that can be adopted in Argus (e.g., background subtraction [24], [25] or object detection models [23]). Although the background subtraction method is more lightweight, we use the object detection method because the object detection method can effectively reduce the number of ROIs to be examined by matching the corresponding labels with the object class of the query (e.g., vehicles or people). In this paper, we use the YOLOv5n model, the lightest model in the YOLOv5 family [23] as it provides reasonable detection accuracy even for small cropped images in 1080p streams of our datasets. Note that app developers can flexibly use different RoI detection methods depending on the processing capacity of smart cameras and the service requirements.

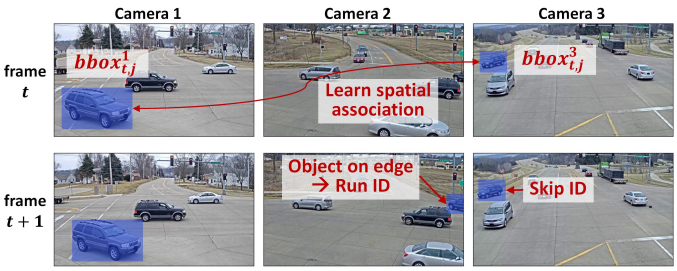


Fig. 13: Handling newly appearing objects on frame edges.

Identity matching. For identification, it is common to train the object type-specific identification models (e.g., vehicle and person) and establish correspondences by measuring the similarity between the feature vectors of the (cropped) images (e.g., Euclidean distance or cosine similarity). We use the dataset-specific identification models and similarity functions to ensure the accuracy of tracking (details in §6.1).

Bounding box matching. The key to leveraging the spatio/temporal association is to match the bounding box of a detected object with the other bounding boxes in the mapping entry and in the previous frame. We use the intersection-over-union (IoU) to measure the overlap between two bounding boxes and detect a match if the IoU value exceeds 0.5 (widely used threshold for object tracking [37]). Note that IoU calculation overhead is negligible (e.g., takes <1 ms for 1,000 matchings on Jetson AGX board).

5 System Optimization for On-Camera Distributed Processing

In this section, we describe system optimization techniques to improve the resource efficiency (namely Multi-Camera Dynamic Inspection (§5.1) and Multi-Camera Parallel Processing (§5.2)) and robustness (§5.3) of our spatio-temporal association over distributed cameras.

5.1 Multi-Camera Dynamic Inspection

5.1.1 *Inter-Camera Dynamic Inspection*

After all cameras run object detection in parallel, we first decide the camera processing order, which heavily affects the identification efficiency (i.e., the total number of required identifications). Specifically, we find that searching the cameras which most likely contain the target object first improves the search efficiency. This is because we can leverage the bounding box location of the identified target object to aggressively skip identification on non-matching bounding boxes in the remaining cameras. For example, consider a case with three cameras (Cameras 1, 2, and 3). At a given timestamp, assume that all cameras detect the same number of vehicles, e.g., four, and a target object is captured by Cameras 1 and 2. If Camera 1 is inspected first, we can find the query object within four identification and skip the identification operations for Cameras 2 and 3. However, if the inspection starts with Camera 3, we need to perform further inspections with Camera 1 and 2 just in case the target object is located out of Camera 3's FoV. Hence, eight identifications are required.

In addition to identification efficiency, the inspection order of the cameras also affects the identification accuracy because our approach relies on identification-based target matching from the first camera. Specifically, inspecting the camera where the target identification accuracy is expected to be the highest leads to the highest association accuracy in the remaining cameras. While the identification accuracy is affected by multiple attributes of the captured object (e.g., the detected object's size, pose, blur), we currently use the bounding box size as the primary indicator, as it affects the identification accuracy significantly [38]. We plan to extend the analysis to other attributes in our future work. For example, in the case of Fig. 7, we consider Camera 2 as the first camera to be inspected as the size of the box of the detected target object is the largest, i.e., has the highest probability that it is correctly identified.

Realizing our insights, however, is non-trivial, as we do not know whether the target exists or not in a camera (and its size if it exists) prior to ID inference. We leverage the temporal correlation of videos to estimate the target's existence and size. Specifically, we calculate each camera's priority at frame t from the identification results at frame $t-1$ as follows (higher value indicates higher priority),

$$\alpha \times \frac{N^i_{t-1}}{N_Q} + (1-\alpha) \times \sum_j^{N^i_{t-1}} \frac{size(bbox^i_{t-1,j})}{H \cdot W}, \tag{2}$$

where N^i_{t-1} is the number of target objects found in C^i at time $t-1$, and N_Q is the number of queries; the higher the ratio is at frame $t-1$, the higher the chances that C^i contains the most number of targets at time t. $size()$ is a function that returns the size (number of pixels) of the bounding box $bbox^i_{t-1,j}$, H and W are the input height and width size of the ID model, respectively; similarly, higher ratio indicates that C^i captures the targets at the largest size. α is a variable that determines the weight of resource efficiency and identification accuracy (set as 0.5 in our evaluation).

5.1.2 Intra-Camera Bounding Box Dynamic Inspection

After spatio-temporal association, we determine in which order we run the ID inference on the non-associated bounding boxes. Specifically, inspecting the detected bounding boxes close to the expected location of the target object is more beneficial, as we can skip the identification on the remaining boxes as soon as the target object is identified. Thus, we order the sequence of boxes to be examined by leveraging temporal association, i.e., sorting

$$\sum_j min\_dist(bbox^i_{t,j}, B^i_{t-1}) \tag{3}$$

for each bounding box $bbox^i_{t,j}$ in C^i at time t in ascending order, where B^i_{t-1} is a set of bounding boxes of the target objects detected in C^i at time $t-1$ and $min\_dist(bbox, B)$ is a function that returns the minimum distance from $bbox$ to any bounding box in B. Note that this only affects the order of the boxes to be examined, but not the tracking result.

5.2 Multi-Camera Distributed Processing

The key challenge of running spatio/temporal association on distributed cameras is the long execution time. The end-to-end execution time may increase if the target objects are not found in the previously inspected cameras due to the sequential execution of the inspection operations. We apply the following techniques that exploit the resources of the distributed cameras to prevent this.

1) Given an image, we perform spatial association-irrelevant tasks on the cameras in parallel, i.e., object detection and ID feature extraction of newly appeared objects at the edges of the frame.
2) If the number of objects in a frame exceeds the predefined batch size (e.g., 4), we distribute the identification tasks to nearby cameras and execute them *in parallel*. Such distribution has a beneficial effect on the end-to-end execution time because 1) current AI accelerators do not support parallel execution of AI models [5][3] and 2) the network latency is relatively much shorter since we need to send only the cropped image (e.g., 85×141), not the full-frame image (e.g., 1080p).

Formally, we define this problem as follows. Given the inspection order determined in §5.1, suppose that we are currently processing Camera i, where a total of N bounding boxes need ID inference after spatio-temporal association. We distribute the ID tasks across K cameras to minimize the total execution time as:

$$\min_{n^1,n^2,\ldots,n^K} \max_j \left(TD(C^i, C^j, n^j) + BP(C^j, n^j) \right), \quad \text{where} \sum_i n^j = N. \tag{4}$$

where n^j is the number of bounding boxes to distribute to Camera j (C^j) to extract the ID features, $TD(C^i, C^j, n^j)$ is a function that returns the network transmission delay to distribute n^j cropped images from C^i to C^j (note that $TD(C^i, C^i, n^i)$ is zero as no transmission is required), and $BP(C^j, n^j)$ is the batched processing latency the identification model on C^j. $TD(C^i, C^j, n^j)$ and $BP(C^j, n^j)$ vary for each C^j depending on its processing capability and network bandwidth; we predict $TD(C^i, C^j, n^j)$ and $BP(C^j, n^j)$ as follows. First, transmission delay for the workload distribution $TD(C^i, C^j, n^j)$ is calculated as

$$TD(C^i, C^j, n^j) = H \cdot W \cdot n^j / BW^j_i, \tag{5}$$

where H, W is the height and width of the image crop (resized to the input size of the identification model) and BW^j_i is the network bandwidth between C^i and C_j (BW^i_i is set as ∞). For each network transmission event between C^i and C_j, we estimate BW^j_i by the transmitted data size divided by the transmission latency (similar to [39]), and update it with Exponential Weighted Moving Average (EWMA) filtering for future prediction. In our current

3. Please note that, while Nvidia offers multi-process service (MPS) and multi-instance GPU (MIG) software packages to facilitate model co-running on their GPUS on cloud servers, they are not supported in the Jetson family devices [8], [9] designed for edge AI. The other AI accelerators such as Google Coral TPU and Intel NCS 2 also do not support parallel execution on the accelerator chip.

implementation, the cameras are connected with a Gigabit wired connection similar to [40], and the distribution latency is negligible compared to the identification model inference latency (e.g., ≈0.3 ms to transmit a 128×128 cropped image over 1 Gbps connection, whereas single identification takes >100 ms as shown in Table 2).

Next, identification latency with batched processing $BP(C^i, n^i)$ is calculated as

$$BP(C^i, n^i) = \lceil n^i / n^i_{batch} \rceil \times T(C^i, n^i_{batch}), \quad (6)$$

where n^i_{batch} is the batch size on C^i, and $T(C^i, n^i_{batch})$ is the identification model latency on C^i with batch size n^i_{batch}. n^i_{batch} is determined at the offline stage by running the identification model on each C^i with different batch sizes and determining the one that maximizes the throughput. At runtime, we update $T(C^i, n^i_{batch})$ upon each inference using the EWMA to account for dynamic resource fluctuation (e.g., due to thermal throttling) similar to [41].

5.3 Improving Robustness

Handling newly appearing objects. One practical issue that needs to be considered when applying spatial association is how to deal with objects that appear in the FoV for the first time. Fig. 13 shows an example. At time t (first row), a target vehicle is found only in Cameras 1 and 3, so the mapping entry is made as $\{bbox^1_{t,j}, N/A, bbox^3_{t,j'}\}$. At time $t+1$ (second row), the target starts to appear in Camera 2. However, if the target is found in Camera 1 and its mapping entry matches the one at time t, the target in Camera 2 will not be inspected. To avoid such a case, we skip mapping-based identity matching for objects that appear in the frame for the first time (i.e., we perform an identification task for a vehicle in the blue box in Camera 2 of the second row in Fig. 13) and match its identity based on identification feature matching. Note that we apply for the mapping-based identity matching for other cameras (e.g., Camera 3).

To effectively identify objects when they first appear, we devise a simple and effective heuristic method. Inspired by the observation that an object appears in the camera's frame by moving from out-of-FoV to FoV, we consider the bounding boxes that are newly located at the edge of the frame as potential candidates, and perform the ID feature extraction regardless of the matching mapping entry if no corresponding identification cache is found.

Handling occlusion. Depending on a camera's FoV, a target object might be obscured by another moving object. For instance, in a camera with a FoV perpendicular to the road, a vehicle in the front lane could occlude a vehicle in the rear lane. Under such circumstances, the detection model might fail to identify the target object. To handle errors resulting from sudden, short-term occlusions, we develop an interpolation technique that leverages the detection results from the preceding frame in the same camera and/or time-synchronised frames from other cameras. Specifically, during a sudden, short-term occlusion, the target object might be visible up to a certain point in the frame, then abruptly disappear mid-frame. If the object remains visible in other cameras, we can estimate the existence of the occluded object by comparing the current mapping entry with past mapping entries. For example, if an object suddenly disappears in Camera 1, Argus searches for a prior mapping entry containing the object located in the previous frame of Camera 1 and extracts the position of the object in other cameras. If corresponding bounding boxes are found in all other cameras, Argus performs object detection and identification on the other cameras. Where occlusion persists for an extended period, we employ periodic cache refreshing (details provided subsequently). It is important to note that such occlusions are rare in practical settings, as objects move at varying speeds and cameras are often installed to monitor the target scene from a high vantage point (e.g., mounted on a traffic light as shown in Fig. 13).

Fig. 14: Snapshots of CAMPUS-Garden1 [21] dataset.

Periodic cache refreshing. To avoid error propagation in our association-based identification (due to occlusion as well as the failure of identification model inference), we limit the maximum number of consecutive skips and perform identification task regardless of a matching mapping entry at the predefined interval (e.g., every 2s). This variable controls the trade-off between efficiency and accuracy.

Time synchronisation. For spatial association, it is important for all video streams to be time synchronised. Argus periodically synchronises the camera clock times using the network time protocol (NTP) and aligns frames based on their timestamps, i.e., two frames are considered time-synchronised if the difference is below the threshold (in the current implementation we set it to 3 ms).[4] Leveraging the synchronized clocks, we match frames across different cameras with the smallest timestamp difference to handle cases where the cameras have different frame rates.

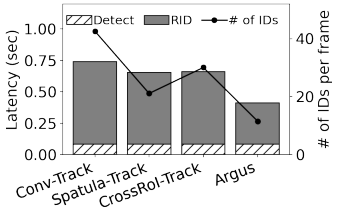
6 Evaluation

6.1 Experimental Setup

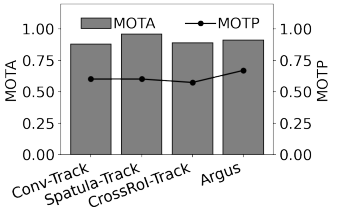
Datasets. We use three real-world overlapping camera datasets for the evaluation: CityFlowV2 [4], CAMPUS [21], and MMMPTRACK [22]. When spatial association is used, we use the pre-generated mapping entries learned with 10% of the data in the dataset. Unless stated otherwise, we report performance on CityFlowV2.

- **CityFlowV2** [4] consists of video streams from five heterogeneous overlapping cameras at a road intersection. The cameras are located to cover the intersection from different sides of the road (Fig. 3). 4 videos are recorded at 1080p@10fps and 1 video is recorded at 720p@10fps with a fisheye lens. Each video stream is ≈3 minutes long and the ground truth data contains 95 unique vehicles.
- **CAMPUS** [21] consists of overlapping video streams recorded in four different scenes. We use the *Garden1* scene, which consists of 4× 1080@30fps videos capturing

4. While we currently assume wired gigabit connection across cameras, recent solutions (e.g., libsoftwaresync [42]) enable sub-ms-level precise clock synchronisation even over wireless networks (§8.1).

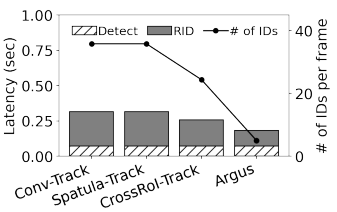


(a) Resource cost.

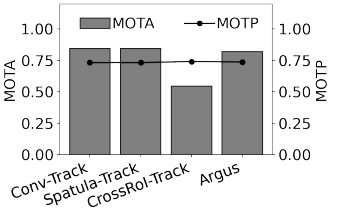


(b) Tracking quality.

Fig. 15: Overall performance on CityFlowV2.



(a) Resource cost.



(b) Tracking quality.

Fig. 16: Overall performance on CAMPUS.

a garden and its perimeter (Fig. 14). We resized the images to 720p as they show comparable object detection performance to the original 1080p at a lower cost. Each video is ≈100s and the ground truth contains 16 unique individuals. Since the dataset provides inaccurate ground truth labels and bounding boxes, we manually regenerated the ground truth for three targets (id 0, 2, 9).

- **MMPTRACK [22]** is composed of overlapping video streams recorded from 5 different scenes: *cafe shop, industry safety, office, lobby, and retail*. In total, there are 23 scene samples (3-8 samples per scene), and each sample is composed of 4-6 overlapping video streams capturing 6-8 people. Each video stream is 360p@15fps and ≈400 seconds (in total 133k frames = 8,800 seconds). We use this dataset to evaluate the robustness of Argus in §6.7.

Queries. For queries, we randomly chose ten vehicles for CityFlowV2, three people for CAMPUS, and two people for MMPTRACK. In the in-depth analysis, we also examine performance with different numbers of queries.

Object detection and identification models. For object detection, we use YOLO-v5 [23]. For vehicle identification in CityFlowV2, we use the ResNet-101-based model [26] trained on the CityFlowV2-ReID dataset [4]. For person identification in CAMPUS and MMPTRACK, we trained the ResNet-50-based model using the dataset. Note that the performance of the re-id model is not the focus of this work and different models can be used. All models are implemented in PyTorch 1.7.1.

Metrics. To measure system resource costs, we evaluate the end-to-end latency and the number of identification model inferences. To measure tracking quality, we use two metrics that are widely used in multi-object tracking [43]: Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA).

- **End-to-end latency** is the total latency for generating multi-camera, multi-target tracking results. Note that the latency includes all the operations required for the system, i.e., image acquisition, model inference, uploading the cropped images to other cameras, and cross-camera communication time.
- **Number of IDs** is the total number of identification model inferences required across all cameras for each timestamp.
- **MOTP** quantifies how precisely the tracker estimates object positions. It is defined as $\frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$, where c_t is the number of matches in frame t and $d_{t.i}$ is the overlap of the bounding box (IoU) of target i with the ground truth. For each frame, we compute the MOTP for each camera separately and report its average.
- **MOTA** measures the overall accuracy of both the detector and the tracker. We define it as $1 - \frac{\sum_t (FN_t + FP_t + MM_t)}{\sum_t T_t}$, where t is the frame index and T_t is the number of target objects in frame t. FN, FP, and MM represent false-negative, false-positive and miss-match errors, respectively. Similarly, we calculate the average MOTA across multiple cameras and report their average.

Baselines. We evaluate Argus against the following state-of-the-art methods. The baselines perform all model operations on the camera where the corresponding frame is captured.

- **Conv-Track** is the conventional pipeline of multi-camera, multi-target tracking (e.g., [14]), which identifies the query object on each camera separately and aggregates the identification results across multiple cameras (Fig. 2).
- **Spatula-Track** adopts the camera-wise filtering approach proposed in Spatula [2] for object tracking. For each timestamp, it first filters out the cameras that do not contain the target objects and then performs the Conv-Track pipeline for the selected cameras. We use ground truth labels for correlation learning and camera filtering, assuming the ideal operation of Spatula [2].
- **CrossRoI-Track** adopts the RoI-wise filtering approach proposed in CrossRoI [3] for tracking. Offline, it learns the minimum-sized RoI that contains all objects at least once over deployed cameras. At runtime, it performs the Conv-Track pipeline only for the masked RoI areas. We use the ground truth labels for the optimal training of the RoI mask, assuming the ideal operation of CrossRoI [3].

Hardware. We use two platforms for smart cameras: Nvidia Jetson AGX and NX. Jetson AGX hosts an 8-core Nvidia Carmel Arm, a 512-core Nvidia VoltaTM GPU with 64 Tensor Cores and 32 GB of memory. Jetson NX hosts a 6-core Nvidia Carmel Arm, a Volta GPU with 384 NVIDIA CUDA cores and 48 Tensors, and 8 GB of memory. We used Jetson AGX for the CityFlowV2 dataset and the MMMPTRACK dataset, and Jetson NX for the CAMPUS dataset. We connected the Jetson devices with a Gigabit wired connection, which is commonly used for existing CCTV networks.

6.2 Overall Performance

Fig. 15 and Fig. 16 show overall performance on CityFlowV2 and CAMPUS respectively. In Fig. 15a and Fig. 16a, the bar chart represents the average end-to-end latency (Detect: object detection latency, ID: identification latency) and the line chart shows the average number of IDs. In Fig. 15b and Fig. 16b, the bar and line charts represent the average MOTA and MOTP respectively.

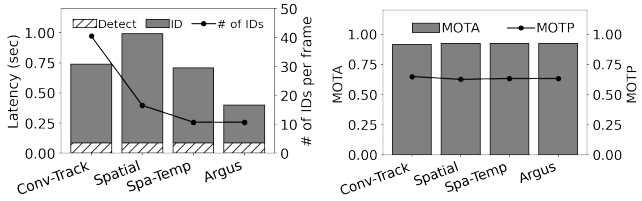
6.2.1 Resource Efficiency

Overall, Argus achieves significant resource savings by adopting the spatio-temporal association and workload distribution, while not compromising the tracking quality. We first examine the resource saving of Argus. In CityFlowV2 with five cameras are involved, Fig. 15a shows that the average number of IDs decreases from 42.6 (Conv-Track) to 21.1 (Spatula-Track), 30.1 (CrossRoI-Track) and 11.6 (Argus). The end-to-end latency also decreases from 740 ms to 650 ms, 660 ms and 410 ms, respectively; Argus is 1.8×, 1.59× and 1.61× faster than Conv-Track, Spatula-Track and CrossRoI-Track, respectively, which are the state-of-the-art multi-camera tracking solutions. We find several interesting observations. First, the latency does not decrease proportionally to the number of IDs because all baselines need to commonly perform object detection in every frame. However, even when object detection is taken into account, Argus significantly decreases the end-to-end latency by 49% by reducing the number of IDs by 73%, compared to Conv-Track. Second, both Spatula-Track and CrossRoI-track significantly reduce the number of IDs by selectively using cameras and RoI areas, respectively. However, the reduction in end-to-end latency is not significant (about 10%). This is because the latency is tied to the longest execution time of all cameras due to the lack of distributed processing capability.

Fig. 16a compares the resource costs for the CAMPUS dataset. The results show a similar pattern to the CityFlowV2 dataset, but the saving ratio of Argus is much higher. Argus reduces the average number of IDs by 7.13× (35.8 to 5.0) compared to Conv-Track and Spatula-Track and 4.86× (24.4 to 5.0) compared to CrossRoI-Track. The end-to-end latency also decreases by 1.72× (from 310 ms to 180 ms) and 1.43× (from 258 ms to 180 ms), respectively. The larger saving is mainly because the moving speed of the target objects (here, people in the CAMPUS dataset) is relatively slow, compared to vehicles in the CityFlowV2 dataset. Therefore, there are fewer newly appearing objects (at the edge) and most of the the identification tasks can be done by spatial and temporal association matching. Interestingly, Spatula-Track shows the same performance as Conv-Track, which is different from the CityFlowV2 case. This is because all target people are captured by all four cameras all the time and thus cameras are not filtered out. CrossRoI-Track reduces both latency and the number of IDs compared to Conv-Track, but its efficiency is still lower than Argus; for CrossRoI-Track, the number of IDs and latency are 24.9 and 419 ms, respectively.

6.2.2 Tracking Quality

We next investigate how spatial and temporal association affects tracking quality. Fig. 15b and Fig. 16b show the MOTP and MOTA on CityFlowV2 and CAMPUS, respectively. Overall, Argus achieves comparable tracking quality, even with significant resource savings. Interestingly, in CityFlowV2, Argus increases both MOTA and MOTP compared to Conv-Track; MOTA increases from 0.88 to 0.91 and MOTP increases from 0.60 to 0.67. This is because several small cropped vehicles are identified by associating with their position from other cameras, which failed to be identified by matching their appearance features from the identification model in the baselines. In CAMPUS, Argus shows almost the same tracking quality as Conv-Track, but MOTA drops slightly from 0.85 (Conv-Track) to 0.82. There were some cases where a target person was suddenly occluded by another person in some cameras. However, Argus identifies the occluded person in the next frame using the robustness techniques (§5.3) to minimize the error.



(a) Resource cost. (b) Tracking quality.

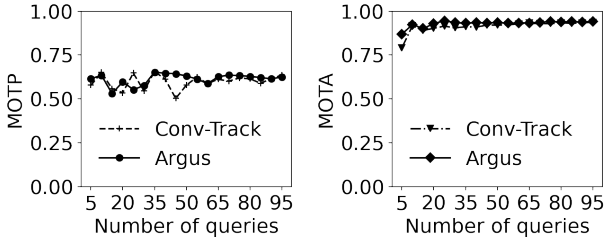
Fig. 17: Performance breakdown on CityFlowV2.

We investigate the benefit of cross-camera collaboration in more detail. In CAMPUS, Spatula also increases the tracking accuracy (MOTA) compared to Conv-Track by filtering out query-irrelevant cameras, thereby avoiding false-positive identifications. However, in CAMPUS, its quality is identical to Conv-Track as no cameras are filtered out. Unlike Spatula-Track, CrossRoI degrades the tracking quality on both MOTA and MOTP; for example, in CAMPUS, CrossRoI shows 0.55 of MOTA, while other baselines including Argus show 0.85 of MOTA. This is because, for object detection and identification, CrossRoI uses the smallest RoI across all cameras in which the target objects appear at least once. Therefore, these tasks sometimes fail due to the small size of the objects in the generated RoI areas.

6.3 Performance Breakdown

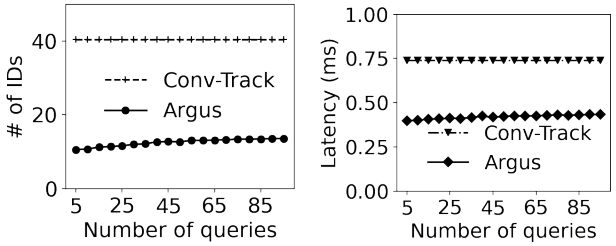
Next, we conduct performance breakdown of Argus to analyze the benefits of Spatio-Temporal association and on-camera distributed processing. For this, we developed two variants of Argus, namely *Argus-Spatial* and *Argus-Spa-Temp*. They utilize spatial and spatio-temporal association for identification optimization on Conv-Track, respectively. They also utilize dynamic inspection of cameras and bounding boxes (§5.1). Argus-Spatial, Argus-Spa-Temp, and Argus sequentially demonstrate the effects of spatial association, temporal association, and multi-camera parallel processing, in comparison with the preceding variant.

Fig. 17a shows the resource cost in CityFlowV2. The spatial association reduces the number of IDs from 40.5 (Conv-Track) to 16.5 (Argus-Spatial, denoted as Spatial) and the temporal association further decreases to 11.6 (Argus-Spa-Temp, denoted as Spa-Temp). These results show that our spatial and temporal association techniques make a significant contribution to overall resource savings. Interestingly, despite the reduction in the number of IDs by Spatial, the latency increases from 854 ms (Conv-Track) to 992 ms (Argus-Spatial) due to the sequential operations on the cameras. However, we observe that the temporal association and workload distribution successfully reduce the latency in turn, to 624 ms and 317 ms, respectively. Fig. 17b shows the tracking quality of CityFlowV2. It confirms again that both spatial and temporal associations do not compromise the tracking quality. We observe similar trends for CAMPUS.



(a) MOTP. (b) MOTA.

Fig. 18: Impact of the number of queries.



(a) Number of IDs. (b) End-to-end latency.

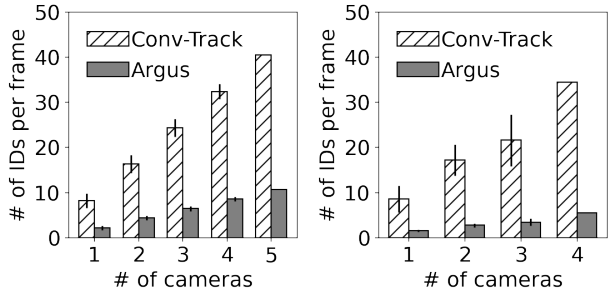
Fig. 19: Impact of the number of queries.

6.4 Impact of Number of Queries

We investigate the impact of the number of queries on system performance. For CityFlowV2, we vary the number of queries from 5 to 95 with an interval of 5. For each number of queries, we randomly select three sets of queries (except the entire set) and report their average performance. Fig. 18a and Fig. 18b show MOTP and MOTA for CityFlowV2, respectively; we observe a similar trend in CAMPUS. For Conv-Track, both the MOTP and MOTA are not significantly affected by the number of queries. This is because Conv-Track runs the identification model on all the detected objects across all cameras regardless of the number of queries; the identification matching accuracy with the query images does not vary with the number of queries as they are randomly selected and averaged. Argus also shows comparable accuracy with Conv-Track (with significantly reduced number of ID operations as shown in Fig. 19, showing that it effectively reduces the identification workload without accuracy drop. Even with a large number of matching attempts with other cameras and queries, Argus identifies the objects accurately.

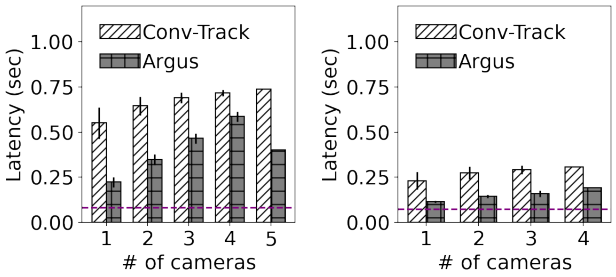
Fig. 19 shows how the number of IDs and the end-to-end latency change depending on the number of queries. The number of IDs and the latency in Conv-Track do not change because all objects in the frame must be examined regardless of the number of queries. In contrast to Conv-Track, Fig. 19a shows that the number of IDs in Argus increases with the number of queries. This is because, at each time, if all target objects are not found on the previously inspected cameras, Argus has to perform the identification operation for all detected objects (which are not filtered out of the spatio/temporal association). This probability increases when the number of queries is large, thereby increasing the number of IDs. However, it saturates when the number of queries is around 50 and, more importantly, it is still much lower than Conv-Track.

Fig. 19 also shows an interesting result. While the number of IDs of Argus increases by 29% from 10.6 to 13.7 when the number of queries is 5 and 95, respectively, in Fig. 19a, the increase in latency is much lower in Fig. 19b, i.e., by 8% from 399 ms to 433 ms. If we exclude the latency for object detection for the analysis, the execution time for identification increases by only 10%, from 315 ms to 349 ms. This result shows the benefit of the Argus's distribution of the identification operations to other cameras.



(a) CityFlowV2. (b) CAMPUS.

Fig. 20: Impact of number of cameras on number of IDs.



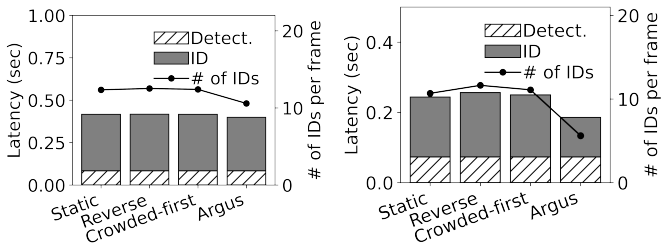
(a) CityFlowV2. (b) CAMPUS.

Fig. 21: Impact of the number of cameras on latency; purple line is the execution time of the object detection.

6.5 Impact of Number of Cameras

We examine the impact of the number of cameras on resource saving. We consider all possible combinations and report their average performance; for example, in the case of three cameras in the CityFlowV2, we report the average result for all 10 ($=_5C_3$) combinations. In this subsection, we do not report the tracking quality results because it is not fair to compare tracking quality for different number and topology of cameras.

Fig. 20 shows the total number of IDs for Conv-Track and Argus. As expected, the number of IDs required for both Conv-Track and Argus increases when more cameras are used. As shown in Fig. 20a, in Conv-Track the number of IDs increases from 8.2 (1 camera) to 42.5 (5 cameras) in CityFlowV2, i.e., by 418%. Similarly, in Argus, it increases from 2.2 to 11.6, i.e., by 423%. Fig. 20b also shows that in CAMPUS, the total number of IDs increases by 316%, from 8.6 (1 camera) to 35.8 (4 cameras) in Conv-Track, while in Argus, it increases from 1.6 to 5.0, i.e., by 213%. However, interestingly, Argus shows a much lower standard deviation across different combinations. This is because the number of IDs in Conv-Track is proportional to the number of objects in a frame. In contrast, the number of IDs in Argus is



(a) CityFlowV2. (b) CAMPUS.

Fig. 22: Impact of inspection order.

determined by the spatial association of the target objects, and is therefore more dependent on the number of queries.

We further investigate the impact of the number of cameras on end-to-end latency. Fig. 21a and 21b show the latency in the CityFlowV2 and CAMPUS datasets, respectively. While Conv-Track performs the identification operations on each camera individually, the latency increases as the number of cameras increases. This is because Conv-Track's latency is tied to the maximum latency across all cameras. Argus also increases latency for both CityFlowV2 and CAMPUS when more cameras are involved, as the waiting time for the entry matching of previously inspected cameras also increases. Nevertheless, the latency of Argus in both Fig 21a and Fig. 21b is still much lower than that of Conv-Track. We observe an interesting case in the CityFlowV2 dataset. The latency of Argus decreases from 587 ms to 400 ms when the number of cameras increases from 4 to 5, even though the number of IDs increases from 8.6 to 11.6. We conjecture that more cameras provide more opportunities for the parallel processing of IDs across cameras, and the benefit becomes apparent when all five cameras are involved.

6.6 Impact of Inspection Order

We investigate the impact of the inspection order on system performance. For the study, we developed three variants of Argus, namely *Static*, *Reverse*, and *Crowded-first*. All of them are built upon the original Argus system. The Static and Reverse variants inspect the cameras and bounding boxes in a predefined static order and in the reverse order of Argus, respectively; Reverse is used to establish the performance lower bound and validate our design choice. The Crowded-first variant, inspired by REV [1], inspects the cameras in a descending order based on the number of bounding boxes. The underlying rationale is that cameras with more bounding boxes are more likely to capture objects of interest. For the bounding box inspection order, we employed the same order as used in Static.

Fig. 22a and Fig. 22b show the latency and the total number of IDs in CityFlowV2 and CAMPUS, respectively. We omit the result of MOTP and MOTA as the differences were marginal. The results validate our design choice. In both datasets, Argus shows shorter latency and fewer identification operations. As expected, Static and Crowded-first show better performance than Reverse, though their effect is still lower than Argus. This is primarily due to a lack of considerations for the relevance of target objects in a scene. This advantage is more evident in CAMPUS. The number of IDs of Reverse is 11.6, while Argus's number is 5.0. Similarly, the latency decreases from 245 ms to 186 ms. This is mainly because the target people mostly remain in one of the cameras during the video stream. Therefore, Argus is capable of reducing the number of IDs by initiating the inspection with potential target objects.

TABLE 3: Performance of Argus in the real-world case study.

	Resource Efficiency		Tracking Quality	
	Latency	Number of IDs	MOTP	MOTA
Parking lot	0.21s	3.2	0.71	0.95

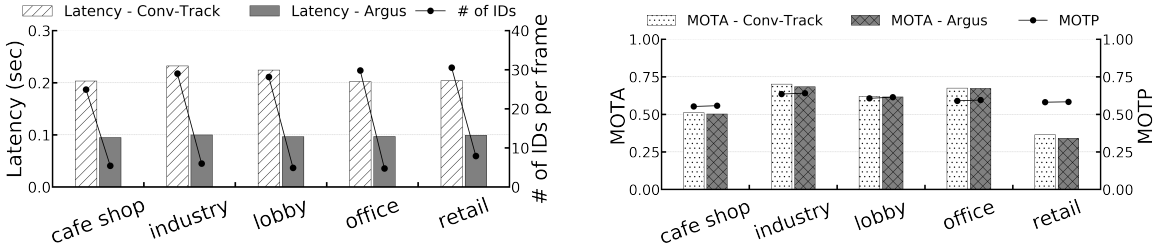
6.7 Robustness on Large Scale Benchmark

We perform large-scale evaluation on the MMPTRACK dataset to validate the robustness of Argus. Fig. 23 compares the resource cost and tracking quality results of Conv-Track and Argus; we omit the results of Spatula-Track and CrossRoI-Track as we observe the similar performance trend in the previous experiments in Fig. 15 and Fig. 16. First, Fig. 23a shows that Argus achieves 2.19× latency gain, which mainly comes from reducing the number of IDs from 28.47 (4-8 objects×4-6 cameras) to 5.79. Next, Fig. 23b shows the tracking quality of Conv-Track and Argus. The base accuracy of Conv-Track varies across scenes depending on ground truth labeling granularity and detector performance (e.g., *retail* scenes contain a lot of occlusions resulting in detection failure, whereas the ground truth label is provided for all objects regardless of occlusion). Argus consistently shows marginal accuracy drop compared to Conv-Track, showing the robustness of our spatio-temporal association.

6.8 Real-world Case Study & System Overhead

To investigate the performance of Argus in practical deployment scenario, we installed four cameras and four Jetson AGX boards within a parking lot of the institute under the consent, employing them to record videos at a resolution of 1080p with a frame rate of 10 frames per second. The parking lot selected for this study spans an approximate area of 50 metres by 25 metres. To ensure a comprehensive coverage, the cameras were positioned at the corners of the parking lot at a height of 3 metres; the corresponding AGX board is connected to the camera via the Ethernet cable and put on the ground. Fig. 24 shows the snapshots of four cameras. Each video stream has an approximate duration of an hour and contains 60 vehicles in total.

Table 3 shows the Argus's overall performance in the real-world case study. When contrasting the results obtained from CityFlowV2, it is interesting to note that the parking lot exhibited marginally superior performance in the aspects both of resource efficiency and tracking quality; we did not compare with the CAMPUS dataset due to the discrepancy in target objects and their respective characteristics. For instance, the average number of identification tasks within the parking lot is 3.2, while the CityFlowV2 dataset showed a higher figure of 10.7. This difference is interesting, especially given that the average number of vehicles captured per video frame in the parking lot exceeded the count of vehicles in the CityFlowV2 dataset. We conjecture this is primarily



(a) Resource cost. (b) Tracking quality.

Fig. 23: Overall performance on MMPTRACK.

Fig. 24: Snapshots of real-world case study.

TABLE 4: Component-wise microbenchmark.

Device	Detection (YOLOv5n [23])		Identification (batch size 4)	
	1920 × 1080	1280 × 720	ResNet 101 [26]	ResNet 50 [27]
NX	0.359s	0.073s	0.399s	0.063s
AGX	0.084s	0.038s	0.217s	0.027s

due to the largely stationary nature of vehicles in the parking lot, allowing the benefit of our spatio-temporal association to be maximized. Similarly, the tracking quality in the parking lot is higher than that in the CityFlowV2 dataset. The MOTP values in the parking lot and the CityFlowV2 dataset were 0.71 and 0.63, respectively. We attribute this to the relatively shorter distance between the camera and the vehicles in the parking lot, enabling the capture of vehicles at a larger scale.

We also delve deeper into the system overhead of Argus with this deployment setup. Aside from object detection and identification, the principal operations of the Argus encompass two elements: (1) mapping-entry matching and (2) workload distribution decision-making. However, according to our measurements derived from the real-world case study, the overhead associated with both these operations is negligible, quantified as less than a few milliseconds. This minimal overhead can be attributed to our efficient management of mapping entries via a hash table for the first operation. Additionally, for the second operation, the system only needs to consider a relatively small number of cases—typically fewer than five identification operations—when making distribution decisions. This streamlined approach contributes to the overall efficiency and effectiveness of the Argus system.

6.9 Microbenchmark

We perform a micro-benchmark to better understand the resource characteristics of model inference on smart cameras. Table 4 shows the inference latencies on Jetson NX and AGX. While the processing capability of smart cameras is still limited compared to the cloud environment, performance can be optimized by applying the right configurations depending on the requirement, e.g., 720p images with people tracking on Jetson NX. We also showed that Argus can further optimize the latency (and corresponding throughput) by leveraging the spatio-temporal association.

7 Related Work

7.1 Cross-Camera Collaboration

7.1.1 *Multi-view Tracking using Camera Geometry*

Camera geometry, also referred to as the geometry of multiple views, has been studied for multiple decades to enable accurate tracking of objects from different camera views. It deals with the mathematical relationships between 3D world points and their 2D projections onto the image plane [28]. By understanding these relationships, the 3D structure of a scene, object, or person has been able to be recovered from multiple 2D views, which enables the tracking of objects even when they move out of one camera's FoV and into another [44]. Camera geometry has been applied in various fields, such as robotics, computer vision, and motion capture, where the use of multiple synchronized cameras with overlapping FoVs can improve the tracking accuracy and robustness of the system [45].

The foundation of multi-view tracking is the estimation of the fundamental matrix, which encodes the geometric relationship between the views of two cameras [28]. This matrix can be used to compute the epipolar geometry, which describes the relationship between corresponding points in the two images and can be utili'ed to find the corresponding point in the other view when a point is detected in one view [44]. By using the fundamental matrix, triangulation techniques can be employed to estimate the 3D position of the tracked object in the scene [28]. Also, bundle adjustment, a non-linear optimisation technique, has been used to refine camera parameters and 3D structure of the scene, leading to a more accurate estimation of the object's position [46].

Despite the advantages of camera geometry in enabling tracking from multi-camera views, there are several limitations in its deployment. One major challenge is the sensitivity to camera calibration errors, which can lead to inaccurate 3D reconstruction and subsequently impact the tracking performance [28]. The calibration process requires the precise estimation of intrinsic camera parameters, such as focal length and lens distortion, and extrinsic parameters, like camera pose and orientation, which can be difficult to obtain in practical applications [44].

7.1.2 Systems for Cross-Camera Collaboration

Enriched video analytics. One direction for cross-camera collaboration is to provide enriched and combined video analytics from different angles and areas of multiple cameras [47], [48], [49]. Caesar [47] detects cross-camera complex activities by designing an abstraction for these activities and combining DNN-based activity detection from *non-overlapping* cameras. Li et al. [48] performs active object tracking by exploiting the intrinsic relationship between camera positions. Visage [49] enables 3D image analytics from multiple video streams from drones. Our work can serve as an underlying on-camera framework for these works, providing multi-camera, multi-task tracking as a primitive task on distributed smart cameras.

Resource efficiency. Another direction for cross-camera collaboration is to reduce the computational and communication costs of multiple video streams by exploiting their spatial and temporal correlation [2], [3], [20], [50]. Spatula [2], [20] aims at a cross-camera collaboration system that targets wide-area camera networks with *non-overlapping* cameras and limits the amount of video data and corresponding communication to be analysed by identifying only those cameras and frames that are likely to contain the target objects. REV [1] also aims at reducing the number of cameras processed by incrementally searching the cameras within the overlapping group and opportunistically skipping processing the rest as soon as the target has been detected. CrossRoI [3] and Polly [51] leverages spatial correlation to extract the minimum-sized RoI from overlapping cameras and reduces processing and transmission costs by filtering out unmasked RoIs. All such works share the same high-level goal as Argus in that they leverage spatio-temporal correlation from multiple cameras, but Argus differs in several aspects, as shown in Table 1.

Distributed processing. There have been several attempts to distribute video analytics workloads from large-scale video data to distributed cameras [40], [52]. VideoEdge [52] optimises the trade-off between resources and accuracy by partitioning the analytics pipeline into hierarchical clusters (cameras, private clusters, public clouds). Distream [40] adaptively balances workloads across smart cameras. Although this work provided a foundation for the development of distributed video analytics systems, it mainly focused on the video analytics pipeline with one camera as a main workload. In this work, we identify that multi-camera, multi-target tracking is a primary underlying task for overlapping camera environments, and propose an on-camera distributed processing strategy tailed to it.

7.2 Resource-Efficient Video Analytics Systems

On-device processing. Many video analytics systems have been proposed to efficiently process a large volume of video data on low-end cameras, e.g., by adopting on-camera frame filtering [53], [54], [55], pipeline adaptation [38], [56], edge-cloud collaborative inference [39], [57], [58], [59], [60] RoI extraction [61], [62], [63], [64], [65]. On-camera frame filtering techniques filter out the computationally intensive execution of vision models in the early stages, e.g., by dynamically adapting filtering decisions [54] and leveraging cheap CNN classifiers [53]. EagleEye [38] selectively uses different face detection and recognition models depending on the quality of face images. MARLIN [56] selectively performs AI model inference for energy-efficient object tracking.

Computation offloading. Several attempts have been made to dynamically adjust video bitrate to optimise the network bandwidth consumption to enable low-latency offloading [64], [66], [67], optimise the video streaming protocol [65], and design DNN-aware video compression methods [68], [69]. The other direction for efficient processing is DNN inference scheduling from multiple video streams on the GPU cluster [17], [18], [70], DNN merging for memory optimisation [71], privacy-aware video analytics [72], [73], and resource-efficient continual learning [74], [75]. While these works manage to achieve remarkable performance improvement, their attempts usually focus on a single camera (or its server). In contrast to these works, we target an environment where multiple cameras are installed in close proximity, and focus on optimising cross-camera operations by leveraging the spatio/temporal association of objects.

8 Discussion

8.1 Practicality

Assumption on on-device processing power. We assume smart cameras with on-device processing power equivalent to Jetson NX–AGX (Table 4) for real-time latency (e.g., 100-200 ms), albeit the performance gain is consistent even for cameras with weaker processing power. We believe that this assumption is practical, considering the recent trend in the increased popularity of on-camera AI processing [40], [76], [77], [78] over cloud-based solutions due to reduced operational costs (e.g., 100K inferences of ResNet-18 costs 0.82 USD on Amazon EC2; six-month deployment of over 1,000 cameras [77] producing 3M hours of 30fps videos incur 3.83M USD, let alone the network streaming and storage costs) and enhanced privacy. This is further enabled by two recent technology trends. First, low-cost, low-power and programmable on-board AI accelerators are becoming available [5], such as Nvidia Jetson, Google TPU and Analog MAX78000. Second, lightweight and accurate embedded-ML models are emerging [12], [13]. With such trends, commercial smart cameras support high-throughput AI inference; for example, Google Nestcam IQ is embedded with Qualcomm QCS605 SoC with Qualcomm Hexagon 685 DSP, which supports MobileNet-v2 inference at only 5-6 ms latency [79]. Thus, Argus's distributed and parallel processing can be practically deployed in real-world scenarios.

Maintaining precise clock synchronisation. We currently assume gigabit wired connection across cameras (§6.1), which is already commonly used for existing CCTV networks (e.g., at an intersection [4] and a campus [21]). However, recent solutions also support robust and accurate clock synchronisation of distributed cameras even over wireless networks (e.g., libsoftwaresync [42] supports $<$1 ms synchronisation for multi-camera motion capture). Argus can utilize such solutions in various real-world deployment scenarios for precise clock synchronisation.

8.2 Generality

Impact of latency gain on app performance. Argus's end-to-end latency gain (≈1.6× over state-of-the-art baselines) offer significant impact in the application perspective. For example, consider a scenario where a police officer is tracking a crime suspect in an urban city, and his AR glasses receive the tracking results from nearby CCTVs and display the bounding box on screen. To ensure that the displayed bounding box does not become stale and outdated from the moving suspect's current position (also referred to as the streaming accuracy [80]), the tracking latency should be kept below 200 ms [80], [81]. Argus reduces the latency from 258 ms (CrossROI [3]) to 180 ms for CAMPUS dataset (urban city scene), thus satisfying the latency requirement.

Generality to other multi-camera tasks. Various multi-camera downstream tasks and apps (e.g., action recognition and surveillance [47], [82], depth estimation and 3D reconstruction [83], [84]) generally operate by (i) associating the same objects across multiple cameras, and (ii) fusing the extracted information (e.g., image crops or features) of the same object across from multiple viewpoints to achieve higher analysis accuracy (e.g., action recognition) or higher-level representations (3D meshes). Thus, multi-camera, multi-object tracking serves as a foundational task for various multi-camera systems; Argus can be generally applied to their object tracking stages to improve latency. For example, 3D motion capture and point cloud reconstruction of a multi-person scene from multi-view RGBD cameras [83], [84] first requires tracking each person's trajectory in each camera and mapping the ones with the same identity. Similarly, multi-camera surveillance pipeline composed of person detection, identification, and action recognition can benefit from Argus to skip redundant inferences.

8.3 Limitations and Future Works

Improving fault tolerance. Argus currently assumes a star topology for cross-camera collaboration, where the most powerful camera becomes the *head* in a group to schedule camera processing order (§5.1), aggregate tracking results, and deliver them to the user. While the coordination overhead is marginal, such architecture is vulnerable to single point of failure when the head camera failure occurs (e.g., due to network disconnection or battery outage). We plan to incorporate diverse fault tolerance mechanisms (e.g., primary-backup [85] or state machine replication [86]) for practical service deployment. Similarly, we plan to improve robustness of our distributed parallel inference technique (§5.2) under sudden network bandwidth drops (e.g., upon encountering unexpected delay in receiving results of the ID inference scheduled to other cameras according to Equation (4), run the missing ID inference in local).

Improving mapping entry representation. For spatio-temporal association, our mapping entry table uses an axis-aligned bounding box representation, which is a common output of conventional object detectors (e.g., YOLOv5). However, we observed occasional failure cases: when the object is not axis-aligned in the frame, the object detector returns an axis-aligned bounding box with large shape mismatch (e.g., overly large one that covers all the object, or overly small one that only covers part of it). We plan to improve our association mechanism by incorporating more enhanced representations (e.g., oriented bounding boxes [87] or segmentation masks [88]) and matching algorithms.

Extension to camera quality assessment and adaptation. Camera configurations (e.g., resolution, frame rate, camera location and height, pan-tilt-zoom (PTZ), AI model) also affect the performance of Argus. While we verified the effectiveness of Argus across three real-world datasets (§6.1) with various configurations including camera types (normal/fisheye lens), capturing heights (street lamp/handheld/ceiling), resolutions (360p/720p/1080p), and frame rates (10/15/30 fps), we can leverage existing systems for adaptive camera quality assessment and configuration (e.g., camera placement simulator [89], runtime PTZ adapter [90], AI inference accuracy profiler [18]) to further improve performance.

9 Conclusion

We presented Argus, a first-kind-of distributed system for robust and low-latency video analytics with cross-camera collaboration on multiple cameras. We developed a novel object-wise spatio-temporal association that optimises the multi-camera, multi-target tracking by intelligently filtering out unnecessary, redundant identification operations. We also developed a distributed scheduling technique that dynamically orders the sequence of camera and bounding box inspection and distributes the identification workload across multiple cameras. Evaluation on three real-world overlapping camera datasets shows that Argus reduces the number of identification model executions and end-to-end latency by up to 7.13× and 2.19× (4.86× and 1.60× compared to the state-of-the-art baselines).

References

[1] T. Xu, K. Shen, Y. Fu, H. Shi, and F. X. Lin, "Rev: A video engine for object re-identification at the city scale," in *IEEE/ACM SEC*, 2022.

[2] S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez, "Spatula: Efficient cross-camera video analytics on large camera networks," in *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2020, pp. 110–124.

[3] H. Guo, S. Yao, Z. Yang, Q. Zhou, and K. Nahrstedt, "Crossroi: Cross-camera region of interest optimization for efficient real time video analytics at scale," *arXiv preprint arXiv:2105.06524*, 2021.

[4] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez *et al.*, "The 5th ai city challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4263–4273.

[5] M. Antonini, T. H. Vu, C. Min, A. Montanari, A. Mathur, and F. Kawsar, "Resource characterisation of personal-scale sensing models on edge accelerators," in *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, 2019, pp. 49–55.

[6] A. Moss, H. Lee, L. Xun, C. Min, F. Kawsar, and A. Montanari, "Ultra-low power dnn accelerators for iot: Resource characterization of the max78000," in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 2022, pp. 934–940.

[7] T. Gong, S. Y. Jang, U. G. Acer, F. Kawsar, and C. Min, "Collaborative inference via dynamic composition of tiny ai accelerators on mcus," *arXiv preprint arXiv:2401.08637*, 2023.

[8] "Nvidia jetson agx," https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-agx-xavier/, accessed: 24 Jan. 2024.

[9] "Nvidia jetson nx," https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-xavier-nx/, accessed: 24 Jan. 2024.

[10] "Google coral," https://coral.ai/, accessed: 24 Jan. 2024.

[11] "Analog max78000," https://www.analog.com/en/products/max78000.html, accessed: 24 Jan. 2024.

[12] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "Morphnet: Fast & simple resource-constrained structure learning of deep networks," in *IEEEE CVPR*, 2018.

[13] P. Guo, B. Hu, and W. Hu, "Mistify: Automating dnn model porting for on-device inference at the edge," in *USENIX NSDI*, 2021.

[14] C. Liu, Y. Zhang, H. Luo, J. Tang, W. Chen, X. Xu, F. Wang, H. Li, and Y.-D. Shen, "City-scale multi-camera vehicle tracking guided by crossroad zones," in *IEEE/CVF CVPR*, 2021.

[15] K. Shim, S. Yoon, K. Ko, and C. Kim, "Multi-target multi-camera vehicle tracking for city-scale traffic management," in *IEEE/CVF CVPR*, 2021.

[16] Y.-L. Li, Z.-Y. Chin, M.-C. Chang, and C.-K. Chiang, "Multi-camera tracking by candidate intersection ratio tracklet matching," in *IEEE/CVF CVPR*, 2021.

[17] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: a gpu cluster engine for accelerating dnn-based video analysis," in *ACM SOSP*, 2019.

[18] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman, "Live video analytics at scale with approximation and delay-tolerance," in *USENIX NSDI*, 2017.

[19] C. Min, A. Mathur, U. G. Acer, A. Montanari, and F. Kawsar, "Sensix++: Bringing mlops and multi-tenant model serving to sensory edge devices," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 6, nov 2023. [Online]. Available: https://doi.org/10.1145/3617507

[20] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez, "Scaling video analytics systems to large camera deployments," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, 2019, pp. 9–14.

[21] Y. Xu, X. Liu, Y. Liu, and S.-C. Zhu, "Multi-view people tracking via hierarchical trajectory composition," in *IEEE CVPR*, 2016.

[22] X. Han, Q. You, C. Wang, Z. Zhang, P. Chu, H. Hu, J. Wang, and Z. Liu, "Mmptrack: Large-scale densely annotated multi-camera multiple people tracking benchmark," in *IEEE/CVF WACV*, 2023.

[23] "Yolo-v5," "https://pytorch.org/hub/ultralytics_yolov5/", 24 Jan. 2024.

[24] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," *arXiv preprint arXiv:1302.1539*, 2013.

[25] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[26] H. Luo, W. Chen, X. Xu, J. Gu, Y. Zhang, C. Liu, Y. Jiang, S. He, F. Wang, and H. Li, "An empirical study of vehicle re-identification on the ai city challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4095–4102.

[27] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *IEEE/CVF CVPR*, 2019.

[28] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] G. P. Stein, "Tracking from multiple view points: Self-calibration of space and time," in *IEEE CVPR*, 1999.

[30] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, pp. 207–232, 2004.

[31] P. Sturm and B. Triggs, "A factorization based algorithm for multi-image projective structure and motion," in *ECCV*. Springer, 1996.

[32] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *IEEE CVPR*, 2004.

[33] J. Black, T. Ellis, and P. Rosin, "Multi view image surveillance and tracking," in *Workshop on Motion and Video Computing, 2002. Proceedings.* IEEE, 2002, pp. 169–174.

[34] L. Lee, R. Romano, and G. Stein, "Monitoring activities from multiple video streams: Establishing a common coordinate frame," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 758–767, 2000.

[35] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[36] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[37] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[38] J. Yi, S. Choi, and Y. Lee, "Eagleeye: Wearable camera-based person identification in crowded urban spaces," in *ACM MobiCom*, 2020.

[39] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "Spinn: synergistic progressive inference of neural networks over device and cloud," in *ACM MobiCom*, 2020.

[40] X. Zeng, B. Fang, H. Shen, and M. Zhang, "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence," in *ACM SenSys*, 2020.

[41] J. S. Jeong, J. Lee, D. Kim, C. Jeon, C. Jeong, Y. Lee, and B.-G. Chun, "Band: coordinated multi-dnn inference on heterogeneous mobile processors," in *ACM MobiSys*, 2022.

[42] S. Ansari, N. Wadhwa, R. Garg, and J. Chen, "Wireless software synchronization of multiple distributed cameras," *ICCP*, 2019.

[43] K. Bernardin, A. Elbs, and R. Stiefelhagen, "Multiple object tracking performance metrics and evaluation in a smart room environment," in *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, vol. 90, no. 91. Citeseer, 2006.

[44] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[45] K. Kanatani, *Geometric computation for machine vision*. Oxford University Press, Inc., 1993.

[46] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer, 2000, pp. 298–372.

[47] X. Liu, P. Ghosh, O. Ulutan, B. Manjunath, K. Chan, and R. Govindan, "Caesar: cross-camera complex activity recognition," in *ACM SenSys*, 2019.

[48] J. Li, J. Xu, F. Zhong, X. Kong, Y. Qiao, and Y. Wang, "Pose-assisted multi-camera collaboration for active object tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 759–766.

[49] S. Jha, Y. Li, S. Noghabi, V. Ranganathan, P. Kumar, A. Nelson, M. Toelle, S. Sinha, R. Chandra, and A. Badam, "Visage: Enabling timely analytics for drone imagery," in *ACM Mobicom*, 2021.

[50] S. Y. Jang, U. G. Acer, C. Min, and F. Kawsar, "Deploying collaborative machine learning systems in edge with multiple cameras," in *2021 Thirteenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. IEEE, 2021, pp. 1–6.

[51] J. Li, L. Liu, H. Xu, S. Wu, and C. J. Xue, "Cross-camera inference on the constrained edge," in *Proc. IEEE INFOCOM*, 2023.

[52] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *IEEE/ACM SEC*, 2018.

[53] 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 *USENIX OSDI*, 2018.

[54] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali, "Reducto: On-camera filtering for resource-efficient real-time video analytics," in *ACM SIGCOMM*, 2020.

[55] M. Xu, T. Xu, Y. Liu, and F. X. Lin, "Video analytics with zero-streaming cameras," in *USENIX ATC*, 2021.

[56] K. Apicharttrisorn, X. Ran, J. Chen, S. V. Krishnamurthy, and A. K. Roy-Chowdhury, "Frugal following: Power thrifty object detection and tracking for mobile augmented reality," in *ACM SenSys*, 2019.

[57] Y. Wang, W. Wang, D. Liu, X. Jin, J. Jiang, and K. Chen, "Enabling edge-cloud video analytics for robotics applications," *IEEE Transactions on Cloud Computing*, 2022.

[58] M. Almeida, S. Laskaridis, S. I. Venieris, I. Leontiadis, and N. D. Lane, "Dyno: Dynamic onloading of deep neural networks from cloud to device," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 6, pp. 1–24, 2022.

[59] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[60] J. Yi, S. Kim, J. Kim, and S. Choi, "Supremo: Cloud-assisted low-latency super-resolution in mobile devices," *IEEE Transactions on Mobile Computing*, 2020.

[61] W. Zhang, Z. He, L. Liu, Z. Jia, Y. Liu, M. Gruteser, D. Raychaudhuri, and Y. Zhang, "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading," in *ACM MobiCom*, 2021.

[62] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "Flexible high-resolution object detection on edge devices with tunable latency," in *ACM MobiCom*, 2021.

[63] K. Yang, J. Yi, K. Lee, and Y. Lee, "Flexpatch: Fast and accurate object detection for on-device high-resolution live video analytics," in *IEEE INFOCOM*, 2022.

[64] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *ACM MobiCom*, 2019.

[65] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang, "Server-driven video streaming for deep learning inference," in *ACM SIGCOMM*, 2020.

[66] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *ACM SIGCOMM*, 2018.

[67] R. Lu, C. Hu, D. Wang, and J. Zhang, "Gemini: a real-time video analytics system with dual computing resource control," in *IEEE/ACM SEC*, 2022.

[68] X. Xie and K.-H. Kim, "Source compression with bounded dnn perception loss for iot edge computer vision," in *ACM MobiCom*, 2019.

[69] K. Du, Q. Zhang, A. Arapin, H. Wang, Z. Xia, and J. Jiang, "Accmpeg: Optimizing video encoding for accurate video analytics," *Proceedings of Machine Learning and Systems*, 2022.

[70] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica, "Chameleon: scalable adaptation of video analytics," in *ACM SIGCOMM*, 2018.

[71] A. Padmanabhan, N. Agarwal, A. Iyer, G. Ananthanarayanan, Y. Shu, N. Karianakis, G. H. Xu, and R. Netravali, "Gemel: Model merging for memory-efficient, real-time video analytics at the edge," in *USENIX NSDI*, April 2023.

[72] F. Cangialosi, N. Agarwal, V. Arun, S. Narayana, A. Sarwate, and R. Netravali, "Privid: Practical,{Privacy-Preserving} video analytics queries," in *USENIX NSDI*, 2022.

[73] R. Lu, S. Shi, D. Wang, C. Hu, and B. Zhang, "Preva: Protecting inference privacy through policy-based video-frame transformation," in *IEEE/ACM SEC*, 2022.

[74] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *USENIX NSDI*, 2022.

[75] K. Mehrdad, G. Ananthanarayanan, K. Hsieh, J. J. , R. N. , Y. Shu, M. Alizadeh, and V. Bahl, "Recl: Responsive resource-efficient continuous learning for video analytics," in *USENIX NSDI*, 2023.

[76] M. Xu, Y. Liu, and X. Liu, "A case for camera-as-a-service," *IEEE Pervasive Computing*, vol. 20, no. 2, pp. 9–17, 2021.

[77] M. Xu, T. Xu, Y. Liu, and F. X. Lin, "Video analytics with zero-streaming cameras," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 459–472.

[78] J. Yi, C. Min, and F. Kawsar, "Vision paper: Towards software-defined video analytics with cross-camera collaboration," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 474–477. [Online]. Available: https://doi.org/10.1145/3485730.3493453

[79] "Deep learning on mobile devices: An in-depth overview," https://data.vision.ee.ethz.ch/cvl/aim21/slides/Andrey-Ignatov-AIMTalk-10-16-2021.pdf, accessed: 24 Jan. 2024.

[80] M. Li, Y.-X. Wang, and D. Ramanan, "Towards streaming perception," in *ECCV*. Springer, 2020.

[81] D. Xu, A. Zhou, G. Wang, H. Zhang, X. Li, J. Pei, and H. Ma, "Tutti: coupling 5g ran and mobile edge computing for latency-critical video analytics," in *ACM MobiCom*, 2022.

[82] W. Li, Y. Wong, A.-A. Liu, Y. Li, Y.-T. Su, and M. Kankanhalli, "Multi-camera action dataset for cross-camera action recognition benchmarking," in *IEEE WACV*, 2017.

[83] K. Lee, J. Yi, and Y. Lee, "Farfetchfusion: Towards fully mobile live 3d telepresence platform," in *ACM MobiCom*, 2023.

[84] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh, "Panoptic studio: A massively multiview system for social motion capture," in *IEEE ICCV*, 2015.

[85] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High availability via asynchronous virtual machine replication," in *in USENIX NSDI*, 2008.

[86] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX annual technical conference (USENIX ATC 14)*, 2014, pp. 305–319.

[87] J. Yi, P. Wu, B. Liu, Q. Huang, H. Qu, and D. Metaxas, "Oriented object detection in aerial images with box boundary-aware vectors," in *IEEE/CVF WACV*, 2021.

[88] "Instance segmentation and tracking using ultralytics yolov8," https://docs.ultralytics.com/guides/instance-segmentation-and-tracking/, accessed: 24 Jan. 2024.

[89] "VideoCAD," https://www.cctvcad.com/videocad_help/. Accessed: 28th Jun. 2024.

[90] M. Wong, M. Ramanujam, G. Balakrishnan, and R. Netravali, "MadEye: Boosting live video analytics accuracy with adaptive camera configurations," in *USENIX NSDI*, 2024.

Juheon Yi is a research scientist at Nokia Bell Labs, Cambridge, UK. He received his Ph.D. from the Department of Computer Science, Seoul National University (SNU), Korea in 2024. His research interests include edge AI and video analytics systems. He won the Best PhD Dissertation Award at SNU and is also a recipient of the Microsoft PhD Fellowship and Best Paper Award at the Students in MobiSys 2021 Workshop.

Utku Günay Acer is a principal research scientist with Nokia Bell Labs in Antwerp, Belgium. He received his PhD degree from the Electrical, Computer and Systems Engineering department of Rensselaer Polytechnic Institute, Troy, NY. His research interests lie broadly in pervasive systems and edge computing. His current work focuses on collaborative and distributed sensing with on-device ML inference for next-generation mobile, wearable, and embedded devices.

Fahim Kawsar currently leads Pervasive Systems Research with Nokia Bell Labs, Cambridge. He holds a mobile systems professorship in computing science from the University of Glasgow. He studies the forms and intelligence of emerging mobile, IoT, wearable devices.

Chulhong Min is a Principal Research Scientist leading the Device Systems team at Nokia Bell Labs in Cambridge, UK. His current research explores next-generation sensory systems to realize multi-modal, multi-device, and multi-sensory functionalities for collaborative and interactive services. Broadly, his research interests include mobile systems, edge computing, on-device AI, and IoT. Chulhong received his PhD in Computer Science from KAIST in 2016.