7-2024

JIGSAW: Edge-based streaming perception over spatially overlapped multi-camera deployments

Ila GOKARN
Singapore Management University

Yigong HU
University of Illinois

Tarek ABDELZAHER
University of Illinois

Archan MISRA
Singapore Management University, archanm@smu.edu.sg

Citation

GOKARN, Ila; HU, Yigong; ABDELZAHER, Tarek; and MISRA, Archan. JIGSAW: Edge-based streaming perception over spatially overlapped multi-camera deployments. (2024). *Proceedings of the IEEE Conference on Multimedia and Expo (ICME), Niagara Falls, Canada, 2024 July 15-19*. 1-6.
Available at: https://ink.library.smu.edu.sg/sis_research/9222

JIGSAW: Edge-based Streaming Perception over Spatially Overlapped Multi-Camera Deployments

Ila Gokarn
Singapore Management University
ingokarn.2019@phdcs.smu.edu.sg

Yigong Hu
UIUC, USA
yigongh2@illinois.edu

Tarek Abdelzaher
UIUC, USA
zaher@illinois.edu

Archan Misra
Singapore Management University
archanm@smu.edu.sg

***Abstract*—We present *JIGSAW*, a novel system that performs edge-based streaming perception over multiple video streams, while additionally factoring in the redundancy offered by the spatial overlap often exhibited in urban, multi-camera deployments. To assure high streaming throughput, *JIGSAW* extracts and spatially multiplexes multiple regions-of-interest from different camera frames into a smaller canvas frame. Moreover, to ensure that perception stays abreast of evolving object kinematics, *JIGSAW* includes a utility-based weighted scheduler to preferentially prioritize and even skip object-specific tiles extracted from an incoming stream of camera frames. Using the CityflowV2 traffic surveillance dataset, we show that *JIGSAW* can simultaneously process *25 cameras* on a single Jetson TX2 with a 66.6% increase in accuracy and a simultaneous 18x (1800%) gain in cumulative throughput (475 FPS), far outperforming competitive baselines.**

***Index Terms*—Edge AI, Machine Perception, Canvas-based Processing**

I. Introduction

Fast *and* accurate streaming machine perception is a cornerstone for many real-time urban applications, such as camera based vehicular tracking and real-time collision avoidance [1]. To ensure that the machine-perceived state does not lag behind the state of the real world, such streaming perception needs to optimize *both* inference accuracy and latency. In fact, the recently proposed *streaming perception paradigm* [2] proposes a novel *streaming accuracy metric* (Figure 1) that balances object localization accuracy and latency, and shows how selectively discarding frames can optimize this metric. This task is especially challenging for multi-tenancy edge computing deployments, where a single resource-constrained, GPU-equipped edge device (e.g., a Jetson TX2) concurrently executes DNN-based perception over video streams from *multiple* $N \geq 2$ vision sensors. While exemplar approaches such as TETRIS [3] and, more recently, MOSAIC [4], utilize the concept of *spatial multiplexing*, to mediate the contention for shared GPU resources across such N streams, they are unsuitable for multi-tenancy streaming perception, as they do not directly consider the latency impacts of stream processing.

We propose a new edge computing paradigm, called *JIGSAW*, that explicitly optimizes DNN-based inference for **multi-camera streaming video applications**, such as traffic surveillance. While *JIGSAW* builds upon the conceptual model of canvas frames introduced in MOSAIC [4], it possesses two key novel features. (a) First, it significantly extends the notion of canvas-based spatial multiplexing to support *spatiotemporal multiplexing*, whereby frames arriving from multiple camera sensors are pre-processed to not only share the pixels within a single canvas frame, but are also differentially interleaved in time (and even dropped) to optimally utilize pixels across multiple consecutive canvas frames. (b) Second, it explicitly accounts for the reality that multi-camera urban deployments often exhibit non-trivial *spatial overlap* between cameras. *JIGSAW* monitors and exploits resulting object-level redundancy, across the N camera views, to reduce the regions of interest that must be squeezed within individual canvas frames, thereby improving the system's maximum camera capacity.

Figure 2 illustrates the high-level operation of *JIGSAW*. Frames from different camera sensors are processed, using very lightweight techniques, to create a set of tiles (Figure 2(b)), indexed by each unique object in the sensing field, that capture an object at multiple spatial scales and possibly from multiple perspectives. A subset of such tiles, corresponding to multiple different objects, are then packed into a sequence of canvas frames (Figure 2(c)) after suitable curation (including potentially discarding all tiles for some objects that the system determines is best processed at a later time instant) and resizing, with DNN inferencing then executed on this sequence of composite canvas frames. *JIGSAW* effectively enhances MOSAIC's innovation in *how* to spatially apportion a shared canvas frame across multiple object-specific tiles by determining (a) *when* and (b) *what* regions of interest should be multiplexed over a sequence of such canvas frames.

Key Contributions:

- *Judiciously exploit multi-camera spatial overlap*: By developing a utility maximizing formalism for tile curation for environments where cameras exhibit varying degrees of partial spatial overlap, *JIGSAW* offers a superior accuracy-vs.-overhead calculus compared to prior extremes that either (a) effectively partition the sensing field across cameras (e.g., [5]), selecting only one tile and discarding others to save computation, or (b) include all tiles across cameras (e.g., [4]), to maximize perception accuracy while ignoring computational cost. Empirical results on the benchmark CityflowV2 [6] multi-view traffic surveillance dataset shows that (i) compared to (b), *JIGSAW* effectively reduces the number of tiles/pixels contending for DNN inferencing by 32% with a 4.3% gain accuracy, and (ii) compared to (a), *JIGSAW* offers 9.55% improvement in task accuracy with only 2.3 ms higher tile processing overhead on average.

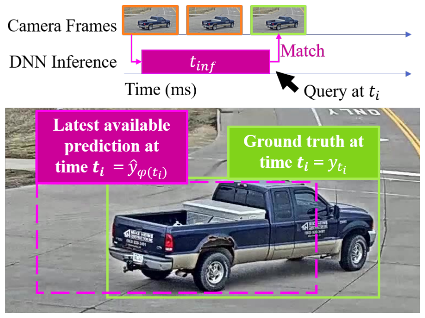
Camera Frames
DNN Inference
t_{inf}
Match
Time (ms)
Query at t_i
Latest available prediction at time $t_i = \hat{y}_{\varphi(t_i)}$
Ground truth at time $t_i = y_{t_i}$

Fig. 1. Streaming Perception: When DNN inference completes with t_{inf} latency, the car has moved (green box) from its predicted location (magenta box). Streaming perception queries the state of the predicted world at time t_i and matches the groundtruth y_{t_i} with the latest available prediction i.e. $\hat{y}_{\phi(t_i)}$.

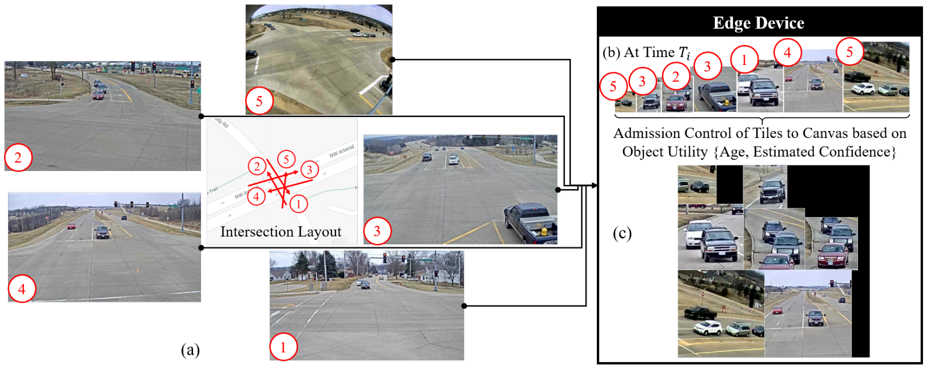
Edge Device
(b) At Time T_i
5
3
2
3
1
4
5
Admission Control of Tiles to Canvas based on Object Utility {Age, Estimated Confidence}
(c)
Intersection Layout
1
2
3
4
5
(a)

Fig. 2. *JIGSAW's* Overall Functionality and Target Application: (a) Multiple spatially overlapped cameras deployed at a traffic intersection are processed by a single edge device (b) At DNN inference time T_i, estimated regions of interest (tiles) are extracted at their appropriate scale from the source camera frames and (c) evaluated for their utility to the streaming perception task before being packed onto a canvas frame.

- *Develop a spatiotemporal pipeline for maximizing multi-camera streaming accuracy:* We show how *JIGSAW* uses a multi-stage, *computationally lightweight* pipeline to both compute *when* object specific tiles should be scheduled for DNN inferencing and *how* such tiles should be spatially combined/squeezed into individual canvas frames. Experimental studies on CityflowV2 show that, for $N = 25$ cameras, *JIGSAW's* bin-packing achieves 28.7% greater streaming object recall (sAR) compared to MOSAIC, which naively packs all tiles from all cameras into a canvas frame.
- *Quantify* JIGSAW's *performance gains and flexibility:* We use an Nvidia Jetson TX2-based [7] implementation of *JIGSAW* to quantify its superior performance. Using CityflowV2, we show that, in contrast to a baseline FCFS processing, *JIGSAW* can simultaneously process ***25 cameras*** on a single TX2 GPU with a 66.6% increase in accuracy and a simultaneous 18x gain in throughput to 19 FPS per camera. Likewise, in contrast to MOSAIC (where all incoming frames are spatially multiplexed into a canvas frame), *JIGSAW* achieves 42.3% increase in streaming perception accuracy without any loss in throughput.

II. Related Work

Recent work on real-time perception on resource-constrained devices can be classified into three distinct categories.

1. Optimising DNN Model and Scheduling. This body of work focuses on modification of DNN structure and operation to reduce/optimise processing latency, including anytime or progressive DNN execution [8], early exit [9], cloud-edge partitioning [10], and models for streaming perception [11].

2. Input Sensor Stream Modifications. To reduce the volume of input sensor streams and reduce computation demands, researchers propose approaches such as explore adaptive content masking [12], frame sampling at the camera for intelligent frame transfer to the edge [13], and multi-resolution frame transfer that preserves resolutions for regions of interest [14].

3. System-level Optimizations. With no modification to the model or the sensor input stream, these systems employ *attention scheduling* to differentially process regions/sections of the input stream [15]. While such work utilizes parallelization of GPU resources and task batching, systems such as MOSAIC [4] and TETRIS [3] extend the philosophy of attention scheduling to curate and extract regions of interest for joint processing on a single GPU. Another body of work in explores the utilization of cross-camera spatiotemporal similarities, for multi-camera deployments, at the edge during computation [5].

Our work cuts across the above three categories, with *JIGSAW* selectively discarding frames from multiple camera input streams, scheduling the creation of the canvas frame for inference with spatiotemporal multiplexing, and ensuring that the evaluated state of the world (as observed from multiple sensors) remains temporally aligned with the physical world.

III. *JIGSAW*

We now describe *JIGSAW's* edge-based, spatiotemporally aware streaming perception pipeline. Figure 3 (best viewed in colour) illustrates the components.

A. Design Choices

To achieve its goal of maximizing the *streaming accuracy* metric under diverse real-world artifacts, *JIGSAW* evaluates the state of the observed world just prior to canvas construction to determine the marginal utility of including specific objects in the canvas frame. Such a design is accommodating of hardware or network heterogeneity, which can require *JIGSAW* to support camera streams with varying frame rates and also generalizes across different deployment characteristics (e.g., observer-object distance and object velocity). *JIGSAW* also makes no a-priori assumption on the presence/absence of overlap between cameras and can operate seamlessly under different deployment conditions.

B. Dynamic Scheduler

The Dynamic Scheduler (DS) orchestrates the execution of other sub-components to support canvas-based streaming

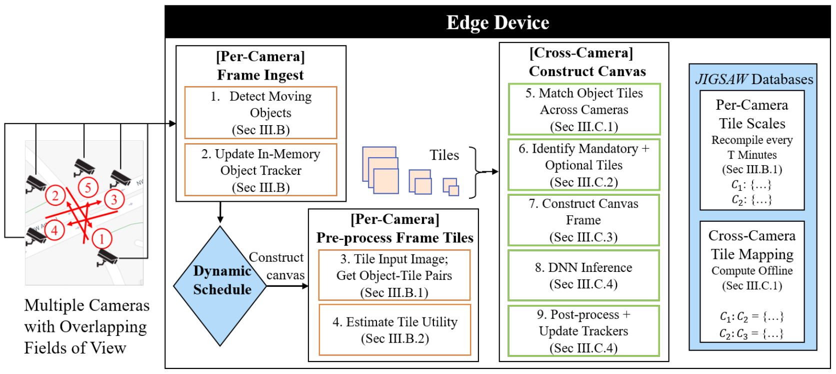
Edge Device
[Per-Camera] Frame Ingest
1. Detect Moving Objects (Sec III.B)
2. Update In-Memory Object Tracker (Sec III.B)
Tiles
Dynamic Schedule
Construct canvas
[Per-Camera] Pre-process Frame Tiles
3. Tile Input Image; Get Object-Tile Pairs (Sec III.B.1)
4. Estimate Tile Utility (Sec III.B.2)
[Cross-Camera] Construct Canvas
5. Match Object Tiles Across Cameras (Sec III.C.1)
6. Identify Mandatory + Optional Tiles (Sec III.C.2)
7. Construct Canvas Frame (Sec III.C.3)
8. DNN Inference (Sec III.C.4)
9. Post-process + Update Trackers (Sec III.C.4)
JIGSAW Databases
Per-Camera Tile Scales Recompile every T Minutes (Sec III.B.1) C_1: {…} C_2: {…}
Cross-Camera Tile Mapping Compute Offline (Sec III.C.1) C_1:C_2 = {…} C_2:C_3 = {…}
Multiple Cameras with Overlapping Fields of View
1
2
3
4
5

Fig. 3. *JIGSAW's* system block diagram (Best viewed in colour). *JIGSAW* processes spatially overlapped multi-camera streams (such as from a traffic intersection) with Per-Camera (in orange) and Cross-Camera (in green) run-time operations, with assistance from databases (in blue) built offline prior to deployment

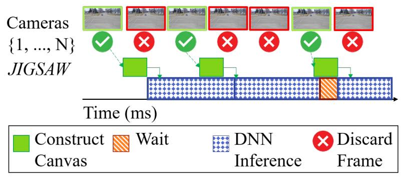

Fig. 4. Conceptual schedule of *JIGSAW's* edge-based streaming perception components

Fig. 5. *JIGSAW's* System Test Bed with 2x NVIDIA Jetson TX2 devices

perception across the N cameras. DS is based on the key observation [2] that the physical world changes dynamically during DNN inference, and that streaming perception is thus often enhanced by skipping prior, *stale* frames and instead running *timely inference* on more recent frames. Accordingly, to minimize processing latency, DS occasionally performs idle-wait, preserving GPU cycles for anticipated fresher frames (instead of processing stale frames). Figure 4 illustrates the non-work conserving DS operation where it uses estimated DNN inference latency to decide whether or not to (a) stitch a new canvas frame from N cameras, (b) discard stale frames after the initial processing described in Section III-C, or (c) idly wait for the imminent arrival of fresher frames.

C. Per Camera Operation

JIGSAW uses a per-camera frame ingest pipeline (orange components in Figure 3) to pre-process frames from N cameras in parallel. The i^{th} ingest pipeline receives the latest frame transmitted from camera C_i and prepares it for downstream canvas-based inferencing, while also maintaining a per-frame representation of the perceived state of the physical world. Objects in motion are estimated using a background subtraction mechanism and the resulting bounding boxes are updated using a Kalman Filter-based Centroid tracker. The tracker maintains a memory function over the last N timestamps (N=5 by default) and retains objects in memory even if are missed by the background subtraction mechanism (e.g., if they are briefly stationary).

Tiling Scales per Camera: *JIGSAW* adopts MOSAIC's [4] mechanism of decomposing each frame into a bag of tiles (at multiple spatial scales), each of which may contain objects of interest (OoI). This tiling operation is performed on-demand, once DS decides to proceed with canvas construction, using the freshest (most recent) camera frames. The number of tiles and their scale dimensions are re-computed periodically during a periodic *profiling* phase. This phrase, executed every T minutes and depicted in blue in Figure 3) uses full-frame detection, on a subset of frames from camera C_i, to profile the observed OoI size (i.e. {height, width}) distribution, followed by K-means clustering and computation of cluster centroids to determine k distinct tiling scales (similar to [4]). Note that a single OoI may be contained in multiple tiles, at different scales, while a single tile can contain multiple objects.

Evaluating Tile Utility: After creating such per-object, OoI-tile mappings, *JIGSAW* next evaluates the *marginal utility* of each tile for each contained object. Given the desire to both maximize accuracy and minimize the inference latency, we split the calculation of such a marginal utility into two sub-operations. Within the the per-camera pipeline, the first-half of the utility function estimates the object detection confidence that can be achieved for each tile (and its contained OoI(s)) from camera C_i if it were to be included onto the canvas frame, *without being squeezed*. We define the tile utility as the ratio of this estimated confidence of the OoI-tile pair to the area consumed by the tile without any squeeze on a canvas frame. This enables *JIGSAW* to prioritize those tiles that provide the highest detection confidence for an encapsulated object while also moderating the object's canvas utilization, allowing fair space allocation across all selected tiles during canvas frame construction. *JIGSAW* also estimates the tile-specific spatial sizing bounds (min & max) for any subsequent squeezing or upscaling–this is empirically set to (1x, 1.1x) for the smallest-scale tile, (0.7x, 1.2x) for the next/medium scales and (0.5x, 1x) for the largest-scale tile. The second-half of the utility function, described later in Section III-D, seeks to minimize per-object processing latency by prioritizing "older" objects (for which DNN inference has not been executed recently), across all N cameras, for inclusion on the canvas frame.

Predicting OoI-Tile Detection Confidence: To estimate the detection confidence for every OoI-tile pair, we utilize a Random Forest Regressor (RFR) trained offline using the following ground-truth features from he CityflowV2 dataset: (i) OoI width, height and area, (ii) ratio of OoI width to OoI height (iii) tile width, height and area, (iv) ratio of object width—height—area to tile width—height—area, respectively, (v) tile coordinates in the original camera frame (i.e. $tile_{xmin}$ & $tile_{ymin}$), and (vi) object class. The RFR estimates an object's detection confidence, assuming the use of a YOLOv5s model (the default model in our *JIGSAW* implementation). Our model achieves an accuracy of 91.46% and a Mean Absolute Error (MAE) of 5% for the predicted detection confidence per object-tile pair.

At runtime, *JIGSAW* evaluates the OoI-tile mappings from camera C_i to estimate the detection confidence per OoI-tile pair, the resulting tile utility relative to canvas frame utilization, and the spatial sizing bounds of the tile. Every camera C_i then opportunistically submits the per-object top-k (user-defined, default $k = 1$) tiles of the highest utility to *JIGSAW* for canvas construction, representing its best possible estimation of the physical world in its FoV.

D. Cross Camera Operation

Cross-Camera Tile Mapping Database: Once the per-camera pipelines have culminated, *JIGSAW* collates the OoI-tile mappings across all N cameras. A Cross-Camera Tile Mapping database therefore assists *JIGSAW* in matching potential tiles, corresponding to the same object but captured from different perspectives, across all N cameras. This database (depicted in blue in Figure 3) is pre-computed prior to system deployment by applying object-ReID algorithms over time-synchronized frames across each of the $\binom{N}{2}$ camera pairs.

At runtime (depicted in green in Figure 3), *JIGSAW* assembles the cross-camera mapping for each object with pair-wise evaluation of cameras (using a computationally efficient quadtree-based indexing technique) that have valid super-region tuples (120×120 pixel blocks across camera pairs that are observed to have at least one common object). As a single tile may contain multiple OoIs, *JIGSAW* then uses SIFT descriptors to find the closest matching unique object in such tiles, eventually deriving, for each unique OoI, a collection of tiles (across multiple camera views) containing the object.

Tile Prioritization: For every unique OoI, *JIGSAW* classifies the tile providing the highest utility (out of all the OoI-tile mappings collected across N cameras) as the *mandatory* tile, with the other tiles being labeled as *optional*. The resulting elements of the set of mandatory tiles (with cardinality equal to the set of unique objects) are then prioritized in decreasing order of the "object age" metric maintained by the object tracker. This age is computed as the difference between the timestamp of the last DNN inference performed on *any* instance of the OoI and the timestamp at which the object's location was last approximated using background subtraction.

Canvas Frame Construction: *JIGSAW* then uses a meta-heuristic approximation of an Inverse Bin Packing algorithm [16] to squeeze as many tiles, across the N cameras, onto the canvas frame of fixed volume. When bin-packing tiles, *JIGSAW* operates by prioritizing *mandatory* tiles with higher age, thereby assuring that the tracking latency does not degrade dramatically for any object. *JIGSAW* attempts to pack all *mandatory* tiles, and as many additional *optional* tiles as feasible, into a canvas frame. This reflects the balance between the desire to improve object accuracy (likely to improve when multiple tiles/perspectives are incorporated) while avoiding excessively squeezing individual tile dimensions. Additionally, if *JIGSAW* is unable to pack all mandatory tiles (e.g., when the total number of objects is very high), the age metric will be updated for all the 'starved' objects, thereby prioritizing those objects in the subsequent round of canvas construction and DNN inference.

DNN Inference and Post Processing: The constructed canvas frame is then submitted to the edge device's GPU for DNN-based inference. *JIGSAW* then asynchronously translates the detections (output of the DNN) to their original camera frame coordinates and performs per-camera Non Maximum Suppression on the bounding boxes to handle cases where a single OoI was included in multiple tiles from the *same* camera frame. The cross-camera database is also used to additionally create 'virtual bounding boxes' on the frames of other overlapping cameras that may not have been included in the canvas frames with the maximum detection confidence available per-object across the included tiles. Finally, *JIGSAW* also updates all associated per-camera trackers with these post-processed detections and the DNN inference timestamp (thus, updating their 'age' metric).

IV. Evaluation Methodology

We now describe the prototype *JIGSAW* implementation and the process for evaluating its effectiveness for a multi-camera streaming perception task.

Evaluation Platform and Model: We implemented *JIGSAW* on the NVIDIA Jetson TX2 device, an edge computing platform that features a 256-core NVIDIA Pascal GPU and a dual CPU unit. The two TX2 devices (the primary edge platform and a secondary backup device to handle transient demand spikes) are connected to a desktop transmitting the camera streams using both wired and wireless connections, illustrated in Figure 5. *JIGSAW's* default DNN is a half-precision TensorRT optimised YOLOv5s model, containing 7.2M parameters operating at 16.5 FLOPs for an image size of 640x640. We size all canvas frames at 640×640 for DNN inference in a batch size $b = 1$ (unless specified otherwise); on the TX2, this yields an inference latency of $\sim 52ms$ per canvas frame (i.e., a throughput of 19 canvas FPS).

Benchmark Dataset: We evaluate *JIGSAW* using the CityflowV2 dataset [6], where camera clusters exhibit spatial overlap as illustrated in Figure 2. It features 3 hours of synchronised video, comprising a mix of residential and highway traffic conditions, at 10 FPS from 40 cameras installed across 10 intersections. We principally use four scenarios that exhibit diversity in object density and kinematic properties. Scenario 'S01' and 'S02' observe higher density and velocity in the "highway" setting with an observed object density per second of 10.8 and 10.96 respectively, and new object arrival per second of 0.6 and 1.24 across $N = 5$ and $N = 4$ cameras respectively. On the other hand, scenario 'S03' and 'S04' observe lower density and velocity in the "residential" setting with an observed object density per second of 3.65 and 5.58 respectively, and new object arrival per second of 0.15 and 0.39 across $N = 6$ and $N = 25$ cameras respectively.

Evaluation Metrics: We principally utilize the metrics of streaming average precision (sAP) and streaming object recall (sAR), introduced in [2]. Both these metrics differ from the standard object detection evaluation metrics of Average

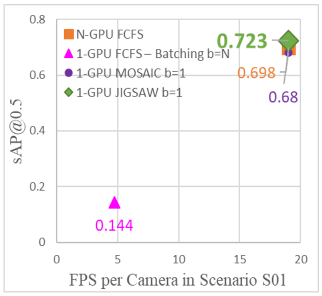

Fig. 6. sAP@0.5 vs. FPS per Camera in 'S01'; $N = 5$; Density: 10.8 obj/sec; Arrival: 0.58 obj/sec

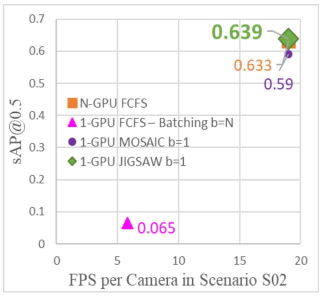

Fig. 7. sAP@0.5 vs. FPS per Camera in 'S02'; $N = 4$; Density: 10.9 obj/sec; Arrival: 1.25 obj/sec

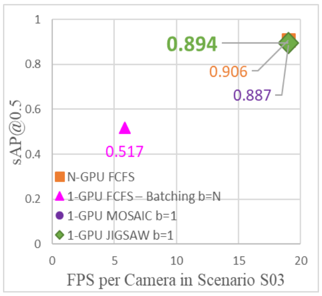

Fig. 8. sAP@0.5 vs. FPS per Camera in 'S03'; $N = 6$; Density: 3.65 obj/sec; Arrival: 0.15 obj/sec

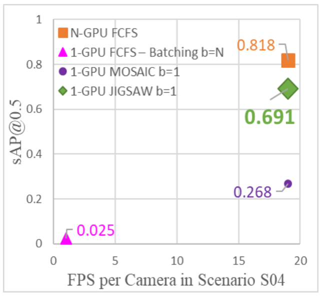

Fig. 9. sAP@0.5 vs. FPS per Camera in 'S04'; $N = 25$; Density: 5.58 obj/sec; Arrival: 0.39 obj/sec

Precision (AP) and Average Recall (AR) in that the streaming detections are aligned for evaluation by the *time of DNN inference* rather than the input frame index. In other words, streaming evaluation compares $(y_t, \hat{y_{\phi(t)}})$ where $\phi(t)$ is the prediction timestamp of the most recent prediction before time t–i.e., $\underset{j}{\text{argmax}}\ s_j < t$, where s_j are the time-stamps of preceding predictions. We set the sAP@0.5 threshold as 0.3 specifically for *JIGSAW* operation, below which the system offloads dense camera streams to the secondary edge device.

Evaluation Baselines: We compare *JIGSAW's* performance against the following baselines:

1. N-GPU Streaming FCFS $b = 1$**:** The evaluation baseline represents the maximum achievable streaming metric value when N cameras are mapped to N GPUs (an 'infinite' GPU setting) to process fresh frames available from the camera C_i in a First Come First Served (FCFS) fashion.

2. 1-GPU Streaming FCFS – Batching $b = N$**:** Fresh frames received from N cameras are batched together for DNN inference by a single GPU task and evaluated for streaming accuracy and recall. The batch size b equals the number of cameras i.e. $b = N$, and represents a strategy where all N frams are processed for inference in parallel.

3. 1-GPU MOSAIC $b = 1$**:** We compare *JIGSAW* against MOSAIC [4] which packs all available tiles from fresh frames across N cameras onto a single canvas frame for processing. For this baseline, we choose the streaming canvas-based evaluation setting with batch size $b = 1$.

4. 1-GPU JIGSAW $b = 1$**:** *JIGSAW* (with batch size $b = 1$ by default) attempts to pack onto the canvas frame all mandatory and as many optional tiles across N cameras, but prioritized by object-specific cross-camera spatiotemporal utility values tuned to the streaming perception task.

V. Evaluation

We first evaluate the streaing accuracy vs. throughput trade-off achieved for N cameras on a single GPU, before additionally exploring the impact of wireless network artifacts on *JIGSAW's* performance.

A. Streaming Accuracy vs. Throughput

JIGSAW's system design jointly leverages (i) the canvas-based processing paradigm to maximise the object detection accuracy across N cameras, and (ii) the streaming perception paradigm to maximise the perception throughput to consistently sustain 19 FPS per camera (i.e. the expected canvas processing throughput) across N cameras on the Jetson TX2. Figures 6, 7, 8, and 9 show that for all evaluated scenarios 'S01', 'S02', 'S03', and 'S04', each with distinct object density and arrival patterns, *JIGSAW* consistently outperforms the 1-GPU FCFS Batching baseline by $57.9\%, 57.4\%, 37.7\%$, and 66.62% respectively in streaming AP (sAP) metric within each scenario. The results show that batching frames from N cameras to support their simultaneous streaming perception on a single GPU provides *almost no benefit*. For the 1-GPU FCFS Batching baseline, a larger N implies higher resource contention due to larger batch sizes (i.e. $b = N$) and proportionately higher latency, which in turn causes the perception pipeline to *lag behind* the real world dynamics. It is important to note here that there is *no* linearity in the relationship between the number of cameras N and sAP (which evaluates how the physical world has changed *during* DNN inference), and is instead a function of $\{N$, new object arrivals rate/kinematics$\}$ which is distinct across the evaluated scenarios. In general, this is reflected in sAP for 'S01' and 'S02' (faster object arrival rate/kinematics) and 'S04' (higher N), when compared to sAP for 'S03' which has a lower object arrival rate across a smaller N.

We note that MOSAIC and *JIGSAW* have comparable performance when the number of cameras/objects packed onto the canvas frame is low. Figures 6, 7, and 8 all show that for $N = (5, 4, 6)$ for Scenario 'S01', 'S02', and 'S03' respectively, *JIGSAW* and MOSAIC achieve comparable results due to the fact that *JIGSAW* is able to squeeze all mandatory and most optional tiles onto the canvas frame. Qualitatively, we observe that *JIGSAW* packs $\sim 32\%$ less tiles than MOSAIC due to the cross-camera per-object matching and utility evaluation, thus allowing each packed tile to acquire slightly larger dimensions, which in turn results in a $\sim 1 - 4\%$ accuracy gain over MOSAIC across Scenarios 'S01', 'S02', and 'S03'. On the other hand, when the number of cameras/objects mapped to a canvas frame is high (Scenario 'S04', where $N = 25$), Figure 9 shows that *JIGSAW* outperforms MOSAIC significantly, achieving 42.3% higher streaming AP (even though both achieve 19 FPS per camera throughput). As anticipated,

	‘S01’ $N = 5$		‘S02’ $N = 4$		‘S03’ $N = 6$		‘S04’ $N = 25$	
	sAP@0.5	FPS	sAP@0.5	FPS	sAP@0.5	FPS	sAP@0.5	FPS
FCFS N-GPU	0.548 ($\downarrow$ 0.15)	18 ($\downarrow$1)	0.434 ($\downarrow$ 0.199)	18 ($\downarrow$1)	0.861 ($\downarrow$ 0.045)	18 ($\downarrow$1)	0.708 ($\downarrow$ 0.11)	18 ($\downarrow$1)
FCFS Batching 1 GPU	0.144	5	0.065	6	0.517	4	0.025	1
MOSAIC 1 GPU	0.532 ($\downarrow$ 0.148)	16 ($\downarrow$3)	0.421 ($\downarrow$ 0.169)	16 ($\downarrow$3)	0.849 ($\downarrow$ 0.038)	16 ($\downarrow$3)	0.197 ($\downarrow$ 0.071)	15 ($\downarrow$4)
JIGSAW 1 GPU	0.557 ($\downarrow$ 0.166)	16 ($\downarrow$3)	0.436 ($\downarrow$ 0.203)	16 ($\downarrow$3)	0.857 ($\downarrow$ 0.037)	16 ($\downarrow$3)	0.676 ($\downarrow$ 0.015)	15 ($\downarrow$ 4)

TABLE I
JIGSAW's WIRELESS SYSTEM DESIGN: ACHIEVABLE STREAMING ACCURACY AND THROUGHPUT - VALUES IN BRACKETS INDICATE DIFFERENCES WITH WIRED SYSTEM RESULTS

MOSAIC's inability to discriminate and differentially schedule/discard tiles causes it to unfairly squeeze each tile beyond its spatial size bounds, thereby leading to a significant loss in object detection confidence. In contrast, *JIGSAW* strikes a fine balance between the need to achieve high detection accuracy and maintain streaming throughput, taking advantage of cross-camera overlap to often avoid redundant processing. In effect, *JIGSAW* is able to achieve accurate detection/tracking with a **far higher camera capacity** (at least 4-fold) than any prior technique employing either FCFS scheduling or spatial multiplexing on a **single edge device**.

Finally, we observe that *JIGSAW's* performance (where $N = 25$ camera streams are multiplexed on a single GPU) is roughly comparable to that achieved by the 25-GPU FCFS baseline, with only a < 0.127 loss in sAP.

B. JIGSAW's *Performance in Wireless Networks*

We evaluate *JIGSAW's* performance over a wireless network, which is subject to artifacts such as variable wireless transmission (empirically varying between $\sim 47 - 52ms$) and video decoding latency (avg=∼4.05 ms)). In suh an environment, the edge device requests frames from the cameras when the DS desires to construct a new canvas frame.

For such wireless operations, we observe (see that Table I) that *JIGSAW* continues to outperform baselines, although the overheads from (i) wireless RTT and (ii) wait time at the camera for a fresher frame effectively reduce the achievable *JIGSAW* throughput to 15-16 FPS per camera with minor $\leq 1\%$ reduction in achievable streaming AP. We also note that the impact of wireless overheads on sAP are felt more keenly for faster-moving dense highway-based traffic in Scenarios ‘S01’ and ‘S02’, with less dramatic impact for the less dense and stationary residential traffic scenarios ‘S03’ and ‘S04’.

Acknowledgements

This work was supported by National Research Foundation, Prime Minister's Office, Singapore under both its NRF Investigatorship grant (NRF-NRFI05-2019-0007), and its Campus for Research Excellence and Technological Enterprise (CREATE) program. The Mens, Manus, and Machina (M3S) is an interdisciplinary research group (IRG) of the Singapore MIT Alliance for Research and Technology (SMART) centre. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. This work was also funded in part by DEVCOM ARL under Cooperative Agreement W911NF-17-2-0196 (ARL IoBT CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S.Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Y. Hu, S. Liu, T. Abdelzaher, M. Wigness, and P. David, “On exploring image resizing for optimizing criticality-based machine perception,” in *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2021, pp. 169–178.

[2] M. Li, Y. Wang, and D. Ramanan, “Towards streaming perception,” *ECCV*, 2020.

[3] T. Stone, N. Stone, P. Jain, Y. Jiang, K.-H. Kim, and S. Nelakuditi, “Towards scalable video analytics at the edge,” in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2019, pp. 1–9.

[4] I. Gokarn, H. Sabbella, Y. Hu, T. Abdelzaher, and A. Misra, “Mosaic: Spatially-multiplexed edge ai optimization over multiple concurrent video sensing streams,” in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023, pp. 278–288.

[5] 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,” in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 186–199.

[6] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, “Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8797–8806.

[7] NVIDIA, “Jetson tx2 developer kit,” *NVIDIA*, 2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2

[8] S. Bateni and C. Liu, “Apnet: Approximation-aware real-time neural network,” in *2018 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2018, pp. 67–79.

[9] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, “Adaptive neural networks for efficient inference,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[10] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, “Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency,” in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 476–488.

[11] M. Li, Y.-X. Wang, and D. Ramanan, “Towards streaming perception,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 473–488.

[12] S. Liu, T. Wang, J. Li, D. Sun, M. Srivastava, and T. Abdelzaher, “Adamask: Enabling machine-centric video streaming with adaptive frame masking for dnn inference offloading,” in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3035–3044.

[13] T. Y.-H. Chen, L. Ravindranath, S. Deng, P. Bahl, and H. Balakrishnan, “Glimpse: Continuous, real-time object recognition on mobile devices,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, 2015, pp. 155–168.

[14] J.-Y. Wu, V. Subasharan, T. Tran, and A. Misra, “Mrim: Enabling mixed-resolution imaging for low-power pervasive vision tasks,” in *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2022, pp. 44–53.

[15] S. Liu, T. Wang, H. Guo, X. Fu, P. David, M. Wigness, A. Misra, and T. Abdelzaher, “Multi-view scheduling of onboard live video analytics to minimize frame processing latency,” in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 503–514.

[16] Y. Chung and M.-J. Park, “Notes on inverse bin-packing problems,” *Information Processing Letters*, vol. 115, no. 1, pp. 60–68, 2015.