1-2024

Demonstrating canvas-based processing of multiple camera streams at the edge

Ila GOKARN
Singapore Management University

Hemanth SABBELLA
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; SABBELLA, Hemanth; HU, Yigong; ABDELZAHER, Tarek; and MISRA, Archan. Demonstrating canvas-based processing of multiple camera streams at the edge. (2024). *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS): Bengaluru, India January 3-7: Proceedings*. 297-299.

Available at: https://ink.library.smu.edu.sg/sis_research/9224

2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS) | 979-8-3503-8311-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/COMSNETS59351.2024.10427123

Demonstrating Canvas-based Processing of Multiple Camera Streams at the Edge

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

Hemanth Sabella
Singapore Management University
hemanthrs@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 demonstrate criticality-aware *canvas-based processing* of multiple concurrent camera streams at the resource constrained edge to show substantial improvement in the accuracy-throughput trade-off. The proposed system focuses the available computation resources on select Regions of Interest (RoI) across all the camera streams by (i) extracting RoI from the input camera stream (ii) 2D bin packing the RoI on a canvas frame and (iii) batching and inferring upon these constructed composite canvas frames with a YOLOv5 object detection model. Our experiments show that such canvas-based processing can (i) sustain real-time processing throughput of 23 FPS *per camera* across 6 concurrent input camera streams (cumulatively 138 FPS) on a *single* NVIDIA Jetson TX2 representing a 475% increase in throughput, with (ii) negligible loss in accuracy as compared to a First Come First Serve (FCFS) baseline running full frame detections on the input camera streams.**

***Index Terms*—Edge Computation, Canvas-based Processing, Multi-Camera Systems**

I. INTRODUCTION

Modern real-time applications for machine visual perception (such as traffic surveillance and autonomous navigation, Figure 1) rely on the *efficient and accurate* processing of multiple simultaneous sensor streams, often at the resource constrained edge, due to their mission-critical and $\{$latency, bandwidth, energy$\}$ sensitive nature. Such edge systems are challenged by a fundamental observation - modern high resolution sensor streams and deep learning vision models with their high computation resource demands and incurred latency are not optimised for real-time machine visual perception tasks [1]. This gap widens on resource constrained computation devices typically deployed in edge systems. One conventional approach to address this challenge is to deploy smaller/lighter models on resource constrained edge devices. This reduces computation latency and conversely increases processing throughput, while incurring an unfavourable reduction in detection accuracy. A second approach is input modification or down sampling of the sensor stream to reduce computation latency by sacrificing high sensing fidelity. The last approach is to *over-provision* the edge device by utilizing expensive GPUs with higher memory and greater number of cores. This reliance on expensive hardware is wasteful and arises, in part, because the conventional perception pipeline processes each frame, *in its entirety*, in a first-come first-served (FCFS) fashion, processing *all* bits/pixels of a single frame *at the same priority*. Such an approach fails to incorporate the reality that different regions of the sensing field may have different priority or criticality [1].

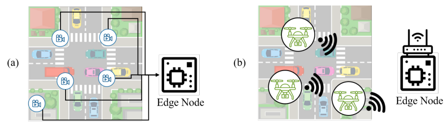

Fig. 1. Target applications: multi-sensor edge-based traffic surveillance in different deployments (a) stationary (left) and (b) drone-based (right).

In light of this accuracy-throughput trade-off, exemplar works have proposed two distinct approaches (i) *Criticality Aware Processing* where selected Regions of Interest (RoI) are processed with higher fidelity either by batching [2], patching or tiling [3], [4], or offloading to a more powerful device [5] for DNN inference, and (ii) *Selective DNN Computation* such as preemptive or anytime computation [6] that exit DNN computation when sufficient object detection confidence has been achieved. These approaches however, do not (i) exploit the concept of criticality holistically across *multiple* simultaneous sensor streams that must be concurrently processed, nor (ii) address the reality that RoI across all input camera streams must share the available computation resource despite the fact that perceived scene characteristics (i.e. number of objects and their dimensions) may vary dramatically across camera streams.

To address this gap, we present a prototype system for *Criticality-Aware Canvas-based Processing* that is centered on the concept of a canvas frame [7], [8] - a blank image unit of fixed dimensions $C \times C$. The canvas frame dimension $C \times C$ is evaluated as the largest image size that can inferred upon by a selected DNN model on a representative edge device while maintaining real-time inference or processing latency. To populate the blank canvas frames, the system processes N camera streams in parallel to decompose each input frame into a set of tiles. These tiles are extracted at camera-specific scales or "crop" dimensions and jointly represent RoI of different sizes across all N cameras. Selected tiles that completely encompass the RoI(s) are evaluated for criticality and Inverse 2D

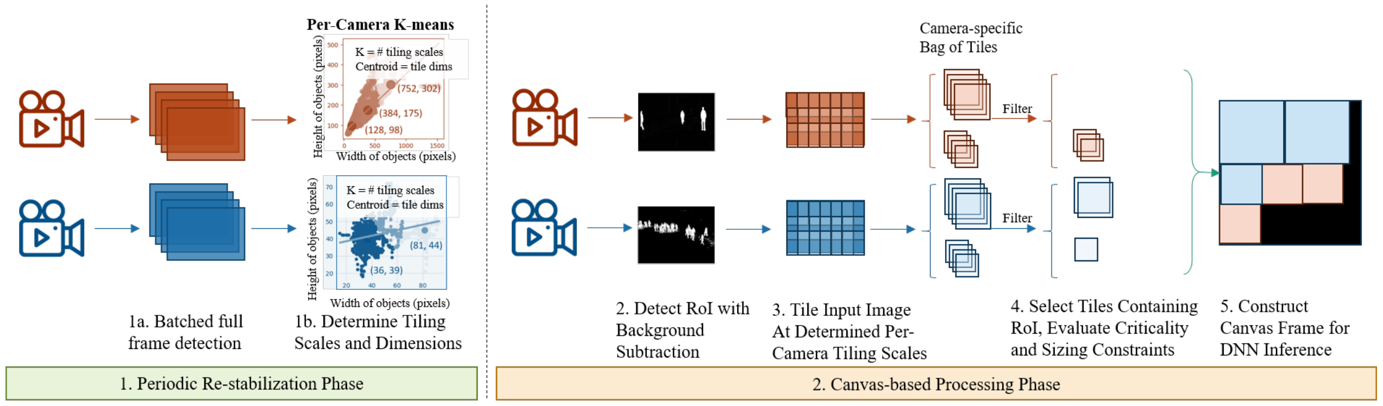
Per-Camera K-means
K = # tiling scales
Centroid = tile dims
Height of objects (pixels)
Width of objects (pixels)
(752, 302)
(384, 175)
(128, 98)
(81, 44)
(36, 39)
1a. Batched full frame detection
1b. Determine Tiling Scales and Dimensions
Camera-specific Bag of Tiles
Filter
2. Detect RoI with Background Subtraction
3. Tile Input Image At Determined Per-Camera Tiling Scales
4. Select Tiles Containing RoI, Evaluate Criticality and Sizing Constraints
5. Construct Canvas Frame for DNN Inference
1. Periodic Re-stabilization Phase
2. Canvas-based Processing Phase

Fig. 2. System Prototype Design of Sub-Components Operating at a Single Jetson TX2 Edge Device (Note: best viewed in colour)

bin packed onto the canvas frame for inference. The system's bin packing or canvas construction mechanism is designed to ensure that RoI tiles are proportionately squeezed onto the canvas frame based on RoI-specific criticality and spatial constraints, thus providing equitable sharing of canvas space for all RoI. The system additionally leverages batched inference of canvas frames to best optimize resource utilization. Our prototype evaluations on a NVIDIA Jetson TX2 [9] edge device show that compared to a full-frame FCFS inference baseline, canvas based processing with a batch size of 4 yields a $4.75\times$ (or 475%) increase in processing throughput to 23 FPS *per-camera* across $N = 6$ camera streams (cumulatively 138 FPS) for a pedestrian detection application with marginal $\leq 1\%$ loss of accuracy.

II. Empirical Results

A. System Design

We first describe the prototype system design which comprises of five fundamental components working in tandem on an edge device, illustrated in Figure 2.

1) A periodic re-stabilization phase initialises the system by running batched full-frame detections across all N camera streams for a user-defined T number of frames every M minutes. This operation has a dual purpose of (i) detecting stationary objects to be maintained in memory and (ii) determining tiling scales and dimensions for subsequent system operations. The number of tiling scales and their dimensions are determined per-camera using K-means clustering of the $\{$height, width$\}$ distributions of detected objects in each camera stream. An elbow detection method yields the k value or the number of clusters/tiling scales, and the centroid of each cluster yields the tiling scale dimensions. Camera-specific tile scales allow the system to adapt to varying object or scene characteristics and ensure that each RoI will be enclosed or captured by at least one tile at its required/appropriate tiling dimension.
2) The system processes each camera input stream in parallel and on a per-frame basis to identify potential RoI using background subtraction. This mechanism prioritizes objects in motion as stationary objects are already accounted for in the previous step.
3) The input frame is then tiled at the determined camera-specific scales to yield a "bag of tiles" that jointly represent the RoI at different scales or "crop" dimensions.
4) Tiles that fully enclose the RoI are evaluated for criticality and spatial sizing constraints based on their tile/object dimensions and cost to canvas (i.e. its canvas utilization)
5) The selected tiles representing all RoI across N camera streams are then 2D Inverse Bin Packed proportional to the determined RoI/tile criticality and squeezed within its (re)sizing constraints onto the canvas frame for inference.

B. Prototype Evaluation Design

Edge Device: We implement the prototype on an NVIDIA Jetson TX2 [9] edge device which features a 256 CUDA-core PASCAL GPU and an ARMv8 multi-CPU architecture.
DNN Model: We use a pre-trained YOLOv5s [10] model that is optimised to run on a half-precision TensorRT engine and fine-tuned for drone-based overhead views of pedestrians. We map selected RoI tiles from i^{th} input frame from N cameras to the i^{th} canvas frame for inference. We empirically determine that a canvas size of 640×640 and a batch size of $b = 4$ best balances optimizes GPU utilization while also maintaining a sufficiently low inference latency of $\sim 170ms$ or conversely high inference throughput 24 canvas frames per second. With RoI tiles from N cameras mapped to each canvas frame, this yields 24 FPS inference throughput *per camera*. The periodic re-stabilization phase that runs for $T = 10$ frames every $M = 30$ seconds performs batched full frame detections incurring a latency of $2.5s$ every cycle, thus tempering the achievable throughput to 23 FPS for each of the N camera frames that are mapped to a single canvas frame.
Dataset: We employ Okutama-Action [11], a drone-based pedestrian detection dataset to evaluate our prototype functionality. This dataset features 43 video sequences captured by drones flying overhead a group of pedestrians, recorded at 30 FPS and 4K resolution.
Choosing N cameras for Canvas-Based Processing: We evaluate canvas-based processing of $\binom{43}{N}$ cameras at a time, with N chosen as a function of (i) the maximum number of camera streams that can be decomposed into tiles in parallel during the asynchronous inference of the previous canvas frame i.e. the next batch of canvas frames must be constructed

For batch size $b = 4$	**mAP@0.5**	**FPS per Camera**	**Cumulative FPS for** $N = 6$
FCFS Baseline	0.798	4	24
Naive Uniform Packing of N input frames	0.704	24	144
Ours	0.785	23	138

TABLE I. Comparison of canvas-based processing against baselines

before the previous batch completes processing, and (ii) the maximum number of dataset-specific chosen tiles from N cameras that can be squeezed within their spatial constraints on a single canvas frame. We empirically choose $N = 6$ for the Okutama-Action dataset.

Metric: We observe the impact of canvas-based processing on both throughput and accuracy to validate our hypothesis that tiling RoI from N camera frames onto a composite canvas frame can increase camera processing throughput with no loss to accuracy. To this end, we discuss the mean average precision (mAP) of the YOLOv5s model at an IoU threshold of 0.5 i.e. mAP@0.5 and the achievable inference throughput, both per-camera and cumulatively across N camera streams for a batch size of $b = 4$.

Baselines: We evaluate the system performance against two opposing baselines:

1. **FCFS baseline** which assumes that a single GPU performs full frame inference on input frames from a single camera stream. For fairness, we evaluate resize and box pad each input frame to fit into a canvas frame of size 640×640. This baseline describes the maximum achievable object detection accuracy for the chosen task and dataset.

2. **Naive bin-packing baseline** where input images from N camera streams are uniformly resized *in its entirety* (i.e. without decomposing into RoI tiles) and packed onto a canvas frame. This baseline describes the maximum achievable throughput for processing N camera streams concurrently and allows us to evaluate the cost or overhead of canvas-based processing.

C. Prototype Evaluation

We now evaluate the achievable throughput and accuracy using canvas-based processing on our system prototype and compare it against the FCFS baseline. As seen in Table I, our system prototype for canvas-based processing is able to substantially moderate the throughput-accuracy trade-off by ensuring the *efficient **and** accurate* processing of $N = 6$ camera input streams on a single Jetson TX2 device. When compared to the FCFS baseline, our system prototype reports significant gains of $4.75\times$ or 475% in throughput with a negligible $\leq 1\%$ loss in accuracy. On the other hand, when compared to the naive uniform packing baseline, our system prototype reports a gain of 8% in object detection accuracy with a minor loss in throughput (i.e. 24 FPS down to 23 FPS per camera). This is due to the overhead of the periodic re-stabilization phase which determines the most-recent RoI dimensions and tiling scales, while also detecting and maintaining stationary objects in memory.

III. Conclusion

We demonstrate the components of criticality-aware canvas-based processing of multiple concurrent camera streams on a single edge device. Our system prototype implemented on a single NVIDIA Jetson TX2 shows dramatic improvement of the throughput-accuracy trade-off against evaluation baselines with the concurrent processing of $N = 6$ camera streams for a pedestrian detection task. We leave workload and object $\{$appearance, motion$\}$ adaptive canvas-based processing for future work to further improve the achievable camera processing capacity and throughput.

Acknowledgment

This work was supported by National Research Foundation, Singapore under its NRF Investigatorship grant (NRF-NRFI05-2019-0007), and in part by The Boeing Company, IBM (IIDAI), ARL W911NF-17-2-0196, and NSF CNS 20-38817. 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.

References

[1] S. Liu, S. Yao, X. Fu, R. Tabish, S. Yu, H. Yun, L. Sha, and T. Abdelzaher, "On removing algorithmic priority inversion from mission-critical machine inference pipelines," December 2020.

[2] 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.

[3] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "Flexible high-resolution object detection on edge devices with tunable latency," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 559–572.

[4] Z. Yang, X. Wang, J. Wu, Y. Zhao, Q. Ma, X. Miao, L. Zhang, and Z. Zhou, "Edgeduet: Tiling small object detection for edge assisted autonomous mobile vision," *IEEE/ACM Transactions on Networking*, 2022.

[5] 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 *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 201–214.

[6] J.-E. Kim, R. Bradford, and Z. Shao, "Anytimenet: Controlling time-quality tradeoffs in deep neural network architectures," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 945–950.

[7] 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.

[8] Y. Hu, I. Gokarn, S. Liu, A. Misra, and T. Abdelzaher, "Under-provisioned gpus: On sufficient capacity for real-time mission-critical perception," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

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

[10] Ultralytics, "Ultralytics/yolov5: Yolov5 in pytorch, onnx, coreml, and tflite," *GitHub*, 2022. [Online]. Available: https://github.com/ultralytics/yolov5

[11] M. Barekatain, M. Martí, H.-F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger, "Okutama-action: An aerial view video dataset for concurrent human action detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 28–35.