

Efficient On-Board Processing of Oblique AAV Video for Rapid Flood Extent Mapping

Vishisht Sharma¹, Sam Leroux², Lisa Landuyt³, Nick Witvrouwen, and Pieter Simoons⁴

Abstract—Effective disaster response relies on rapid situational awareness, where oblique aerial video is the primary modality for initial scouting due to its ability to maximize spatial coverage and situational awareness in limited flight time. However, the on-board processing of high-resolution oblique streams is severely bottlenecked by the strict size, weight, and power constraints of autonomous aerial vehicles (AAVs). The computational density required to process these wide-field-of-view streams precludes low-latency inference on standard edge hardware. To address this, we propose temporal token reuse (TTR), an adaptive inference framework capable of accelerating video segmentation on embedded devices. TTR exploits the intrinsic spatiotemporal redundancy of aerial video by formulating image patches as tokens; it utilizes a lightweight similarity metric to dynamically identify static regions and propagate their precomputed deep features, thereby bypassing redundant backbone computations. We validate the framework on standard benchmarks and a newly curated Oblique Floodwater Dataset designed for hydrological monitoring. Experimental results on edge-grade hardware demonstrate that TTR achieves speedups ranging from $1.58\times$ to $1.69\times$ on video datasets (Floodwater, AAVid, Audi Autonomous Driving Dataset) with negligible degradation in segmentation accuracy ($<1\%$ mean intersection over union). On the image-based FloodNet benchmark, where temporal redundancy is inherently absent, TTR preserves full accuracy without degradation. These findings confirm that TTR effectively shifts the operational Pareto frontier for oblique video understanding, with the magnitude of efficiency gains scaling directly with the degree of temporal redundancy present in the input stream.

Index Terms—Adaptive computation, disaster management, edge computing, flood mapping, oblique aerial video, real-time semantic segmentation, remote sensing, spatiotemporal redundancy, unmanned aerial vehicles (AAVs).

Received 3 January 2026; revised 16 May 2026 and 15 August 2026; accepted 22 August 2026. Date of publication 31 August 2026; date of current version 25 September 2026. This research was funded by the Belgian Science Policy Office (BELSPO) via the Stereo IV Program, under Grant SR/00/415 (FLOWS). (Corresponding author: Vishisht Sharma.)

Vishisht Sharma is with the IDLab, Department of Information Technology, Ghent University—imec, 9052 Gent, Belgium, and also with the Environmental Intelligence Unit, Flemish Institute for Technological Research (VITO), 2400 Mol, Belgium (e-mail: vishisht.sharma@ugent.be).

Sam Leroux and Pieter Simoons are with the IDLab, Department of Information Technology, Ghent University—imec, 9052 Gent, Belgium (e-mail: sam.leroux@ugent.be; pieter.simoens@ugent.be).

Lisa Landuyt and Nick Witvrouwen are with the Environmental Intelligence Unit, Flemish Institute for Technological Research (VITO), 2400 Mol, Belgium (e-mail: lisa.landuyt@vito.be; nick.witvrouwen@vito.be).

Code is available online at: <https://github.com/decide-ugent/adaptive-segmentation>.

This article has supplementary downloadable material available at <https://doi.org/10.1109/JSTARS.2026.3729310>, provided by the authors.

Digital Object Identifier 10.1109/JSTARS.2026.3729310

I. INTRODUCTION

THE widespread adoption of unmanned aerial vehicles (AAVs) in applications ranging from disaster response to precision agriculture [1] has created an unprecedented demand for real-time, on-board data analysis. However, processing AAV-captured video presents a unique and often overlooked computational dilemma, born from a fundamental conflict between the data's characteristics and the operational nature of modern deep learning models. On the one hand, the video is frequently captured at high resolutions, e.g., 1920×1080 (Full HD) or even 4 K, to ensure that fine-grained details are preserved for accurate analysis. This high pixel count inherently demands significant processing power and memory bandwidth for every single frame. On the other hand, these data are also intrinsically characterized by vast temporal redundancy [2]. In a typical mission, a AAV may execute prolonged flight patterns, including hovering for surveillance, carrying out systematic pans for mapping, or orbiting specific points of interest, such as during disaster response scenarios. These maneuvers cause large portions of the observed scene (e.g. large agricultural fields, static water bodies, unchanging forests, or stationary buildings) to remain perceptually identical or nearly identical across tens, if not hundreds, of consecutive frames.

This redundancy is impacted by the brute-force nature of modern deep neural networks. While powerful, these architectures, by design, treat each incoming frame as an independent problem. They dictate that every pixel, or every patch in the case of a transformer-based network [3], must be processed through the entire computational graph, from the initial layers to the final prediction head. This exhaustive processing is ill-suited for video streams where the temporal correlation between frames is high, often for multiple consecutive frames.

Fig. 1, for instance, demonstrates this correlation by showing how patches of water and vegetation maintain high self-similarity even with significant camera motion and across a gap of over 200 frames. Consequently, processing these large, unchanging static regions with a computationally expensive model constitutes an inefficient use of computational cycles. This inefficiency is not a minor technical issue; it results directly into operational limitations: shorter flight times due to the high power draw of the on-board processing [4], [5], a data surge that slows down postmission analysis on the ground, and a bottleneck for applications that demand immediate situational awareness, such as search and rescue or flood mapping, where latency can render data obsolete the moment it is processed.

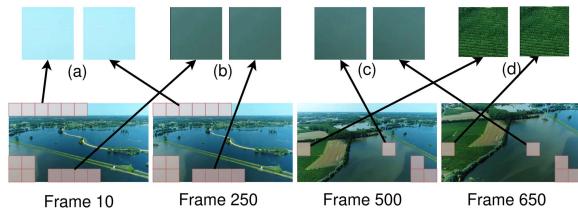


Fig. 1. Visualization of four pairs of patches across frame sequences, demonstrating the high self-similarity over time. For instance, the sky patch in (a) and the water patch in (b) remain almost identical between Frame 10 and Frame 250. Similarly, patches for water (c) and vegetation (d) show negligible change over hundreds of frames despite camera motion. Our TTR framework leverages a high cosine similarity score between such patches to bypass their recomputation.

This disparity, where some parts of a scene are highly dynamic (e.g., moving vehicles, flowing water, or the leading edge of an ongoing disaster) while the vast majority remains static, calls for a paradigm shift toward “adaptive computation” [6], [7]. This principle rejects the one-size-fits-all, uniform processing model and instead suggests a more intelligent, dynamic allocation of computational effort, precisely tailored to the spatiotemporal complexity and novelty of the input data. An adaptive system built on this paradigm operates with a nuanced understanding of the scene’s dynamics. For instance, it would allocate the maximum computational budget to track and segment high-priority, moving objects or rapidly changing land cover. These are the regions where accuracy is the most important and where new information resides. Conversely, for the large, static, and unaffected areas of the scene (e.g., the sky or a distant mountain range), it would expend only the minimal computational effort required to verify their static nature, before reusing previously computed results.

This processing strategy is not just an optimization; it is a requirement for enabling the potential of on-board AAV applications in real-world scenarios. By focusing finite computational resources where they are needed most, this approach makes it possible to achieve two critical goals simultaneously: maintaining high fidelity and accuracy in the dynamic regions of interest, while improving overall system efficiency. Our method bridges the gap between computationally demanding, high-accuracy models and the latency requirements of power-constrained edge devices, enabling their deployment in real-world missions.

The primary contribution of this article is the proposed temporal token reuse (TTR) framework, which directly targets the temporal redundancy. It introduces a computationally inexpensive mechanism to cache and reuse the deep feature representations of static scene regions. By identifying and reusing the precomputed features of tokens that have not visually changed since the previous frame, TTR reduces the computational load on the main segmentation backbone. This ensures that processing power is precisely focused only where it is needed most: on the dynamic and actively changing parts of the scene. This adaptive approach is crucial for optimizing the performance-to-power consumption ratio, enabling the potential of deploying high-accuracy, high-fidelity segmentation models on power- and compute-constrained aerial platforms.

Given our approach of treating image patches as tokens, a transformer-based architecture might seem like the obvious

choice. However, we deliberately diverge from this path due to the practical realities of on-board AAV processing. The very strength of transformers, their ability to model global relationships via self-attention, is also their primary drawback in this context, as its quadratic complexity is exceptionally demanding for high-resolution, real-time video on power-limited hardware. In contrast, convolutional neural networks (CNNs) offer a more pragmatic and tenable foundation, with a rich history of lightweight designs optimized for edge deployment. In addition, emerging CNN frameworks grant us the transformer-like granular control over distinct image regions, a necessity for our temporal reuse strategy while retaining the inherent efficiency of CNNs. By building upon a CNN backbone, our approach represents a deliberate tradeoff: we prioritize the practical feasibility of deploying a high-fidelity, adaptive system in a real-world operational environment over adherence to the latest, albeit computationally expensive, architectural trends.

II. RELATED WORK

Our work builds upon two domains of research: video segmentation and efficient deep learning. The challenge of deploying deep neural networks on resource-constrained platforms, such as AAVs, has catalyzed significant innovation in model optimization. As surveyed by [4], these efforts can be broadly divided into static model compression, efficient network design, and dynamic inference strategies.

Recent flood-focused studies provide complementary perspectives on this problem. Sundaresan and Solomon [8] investigated supervised postdisaster flood segmentation with DeepLabv3+ and unmanned aerial system imagery. Simantiris and Panagiotakis [9] proposed an unsupervised color-based segmentation method for AAV imagery, while Manaf et al. [10] studied robust aerial-image classification with explainable deep learning for postflood analysis. AIFloodSense provides a broader aerial flood dataset for semantic segmentation and scene understanding [11]; we additionally evaluate our models on this dataset in Section IV-C. Together, these studies provide relevant context for flood segmentation, aerial-image analysis, and dataset development.

A. Efficient Network Architectures

Static methods aim to create a single, efficient model before deployment. This includes techniques like network pruning, which removes redundant weights or channels [12], a field that continues to evolve with methods for structured pruning that can be reconfigured without retraining [13] and data-free pruning techniques for large-scale models [14]; quantization, which reduces the numerical precision of network parameters [15], with modern post-training methods now capable of accurately quantizing large vision models with minimal data [16]; and knowledge distillation, where a smaller student network is trained to mimic the output of a larger, more powerful teacher network [17], with recent research exploring how to distill knowledge from multiple diverse teachers effectively [18]. Furthermore, a significant body of work has focused on designing inherently efficient CNN architectures from the ground up, such as MobileNetV2 [19], ShuffleNetV2 [20], and ERFNet [21],

which utilize optimized operations like depth-wise separable convolutions and channel shuffling. This trend continues with modern designs that employ large kernel convolutions for efficiency at scale [22] and lightweight vision transformers tailored for mobile applications [23]. While these approaches successfully reduce the overall computational footprint (as measured in FLOPs) and memory requirements, they share a fundamental limitation: they still apply the same uniform computational graph to every pixel of every input frame. This makes them incapable of exploiting the massive spatio-temporal redundancy inherent in video data.

B. Frame-Level and Motion-Based Temporal Methods

To address temporal redundancy at a coarse frame level, a straightforward strategy is the Keyframe-based approach. These techniques perform full, computationally expensive segmentation on sparsely selected keyframes and then use a faster method for the intermediate delta frames. Early methods relied on simple interpolation, but more advanced techniques have been developed to propagate and aggregate deep features from keyframes to subsequent frames, improving accuracy [24]. While simple in principle, this global decision-making process presents a stark tradeoff between efficiency and temporal fidelity. A fixed, low keyframe rate drastically increases the risk of missing transient events, which has led to research on adaptive frame selection strategies that dynamically choose keyframes based on scene content [25]. More recent works have framed this as a policy-learning problem, where a lightweight network learns to dynamically skip frames to maximize both efficiency and accuracy for a given computational budget [26]. Further extending this, recent learning-based paradigms like cotraining have been proposed to leverage the information from unlabeled intermediate frames better, improving overall temporal consistency [27].

A more sophisticated and dominant paradigm is feature propagation with optical flow. This method calculates a dense optical flow field to warp feature maps from a keyframe to the current frame, thus maintaining temporal consistency and improving segmentation accuracy over time [28]. However, dense optical flow estimation is itself a computationally intensive task, which can offset the intended efficiency gains [29]. More critically, it is prone to errors in challenging but common scenarios, such as occlusions, textureless surfaces, and illumination changes, which are prevalent in aerial footage and can degrade segmentation quality [30]. Recognizing these limitations, recent research has focused on mitigating flow-induced errors, for example, by using attention mechanisms to blend features selectively and leveraging uncertainty estimation to identify and refine unreliable warped regions [31]. State-of-the-art approaches now seek to replace explicit flow entirely with more robust, learnable correspondence matching at the feature level, which can better handle large displacements and appearance changes [32]. Despite these advances, both keyframing and motion warping methods apply their temporal logic uniformly across the entire frame, lacking the fine-grained adaptability to handle scenes with spatially varying dynamics.

C. Spatially Dynamic Computation

Parallel to temporal efficiency methods, a highly relevant line of research has focused on exploiting spatial sparsity within a

single frame. The guiding principle, as articulated by [33], is that not all pixels are created equal. This has led to a variety of dynamic networks that allocate computation unevenly across the spatial domain [2]. One major approach involves the selective processing of image regions. For instance, SBNet [34], which inspired the SegBlocks framework, uses a lightweight policy network to identify a sparse subset of image blocks for processing. This concept has been modernized and extensively explored in vision transformers, which employ strategies like adaptive token selection to process a variable number of tokens based on input complexity [35], token merging to consolidate redundant features [36], and token routing to focus computation on the most salient image regions [37].

Other methods adapt the network's structure or depth dynamically. SkipNet [38] and BlockDrop [7] learn to conditionally bypass entire residual blocks for simpler image regions, creating adaptive inference graphs tailored to the input [39]. This principle is now central to dynamic vision transformers that can conditionally skip entire layers or self-attention blocks based on intermediate feature representations [40]. A more granular approach, dynamic convolutions [41], which was further refined to be more hardware-friendly [42], exploits spatial sparsity directly within the convolutional operation itself. Another variant, proposed by [6], adapts computation time per pixel, halting computation early for regions classified with high confidence, a concept now successfully applied to vision transformers through layer-wise early exiting mechanisms [43]. These methods have proven highly effective for efficient high-resolution image processing [44], [45]. However, their core limitation is that their focus is strictly intraframe; they re-evaluate the spatial complexity of the entire scene from scratch for every single video frame, thereby failing to capitalize on the most significant source of redundancy in aerial video surveys: temporal staticity.

In stark contrast, our TTR framework is conceived to address these shortcomings directly. It synthesizes the fine-grained, patch-level adaptability of spatial sparsity methods like SBNet [34] and extends this principle into the temporal dimension. Instead of the costly and error-prone estimation of optical flow, TTR employs a computationally cheap similarity metric to identify static regions over time precisely. Unlike coarse keyframe-based methods, TTR operates at a local level, allowing it to preserve high temporal fidelity for dynamic objects while aggressively saving computation in static areas. By caching and reusing features for unchanged regions, TTR elegantly avoids redundant processing, ensuring that finite computational resources are focused exclusively where they are needed most. This makes it a uniquely flexible, robust, and efficient solution for the demands of real-time segmentation on video sequences.

III. METHODOLOGY

A. SegBlocks: A Foundation for Adaptive CNN Processing

The SegBlocks [46] framework represents a critical step toward making CNNs more adaptive. It partitions an input image into a grid of blocks and dynamically adjusts the processing resolution based on spatial complexity. This innovation allows a CNN to focus its computational resources by processing detailed or complex regions at a high resolution while using a lower,

more computationally efficient resolution for simpler areas. We leverage this patch-based architecture as the fundamental building block for our temporal optimization, as it provides the necessary mechanism to process image regions independently.

A core challenge in any sparse, patch-based CNN framework is preserving spatial context and continuity. The effectiveness of CNNs relies on their receptive field, which allows neurons to gather information from a local neighborhood. In a standard dense convolution, this neighborhood is always available. However, in a sparse framework where only a subset of active patches are processed, the filters at the boundaries of these patches face a problem: their receptive fields extend into neighboring inactive patches whose features for the current network stage have not been computed. This lack of contextual information breaks the feature sharing across the frame, leading to artifacts visible as discontinuities or incorrect classifications along the seams of the processed blocks, thus degrading the final segmentation quality.

To resolve this dilemma, SegBlocks employs a specialized operation known as boundary feature copying, implemented within a BlockCopy [46] module. The primary function of this module is to reconstruct the necessary local neighborhood for each active patch before it undergoes convolution. After a sparsity mask indicates which patches are active, the BlockCopy module prepares the input for the convolutional layer by creating a temporary, larger patch padded with features from its immediate one-pixel-deep neighborhood, i.e., its 8 neighbors. The key innovation lies in how this padding is sourced as follows.

- 1) If a neighboring patch is processed at the same resolution (i.e., it is also “active”), its features are copied directly from the current stage’s input feature map.
- 2) If a neighboring patch is processed at a lower resolution, its smaller feature map is retrieved and then upsampled, e.g., using nearest-neighbor interpolation to match the resolution of the active patch before being copied into the padding region.

Once this contextually aware padded patch is constructed, the respective convolutional block is applied to it. The receptive field of the convolutions can now operate correctly, even at the edges. This mechanism seamlessly stitches the sparsely computed results together, preserving the integrity of the convolutional operation while enabling the computational savings of adaptive-resolution processing.

B. Temporal Token Reuse

While SegBlocks optimizes computation within a single frame by varying spatial resolution, our primary contribution, TTR, extends this concept into the temporal domain for video processing. We modify the core mechanism: instead of merely reducing the processing resolution, TTR skips computation entirely for patches that have not changed significantly from the previous frame. By identifying these temporally redundant tokens, we reuse the rich, deep feature maps computed for the previous frame, achieving substantial efficiency gains in scenes with static elements.

The architecture and operational flow of this process are detailed in Fig. 2. The TTR process for each frame is formalized into two key stages, detailed in Algorithms 1 and 2.

Algorithm 1: TTR Sparsity Mask Generation.

Require: Current Frame I_t
Require: Previous Frame’s Patch Cache:
 PrevPatchCache
Require: Cosine similarity threshold τ
Ensure: Sparsity Mask: SparsityMask
Ensure: Updated Cache: CurrPatchCache
 1: Let P_t be the set of patches $\{p_{t,i}\}$ from frame I_t
 2: Initialize empty SparsityMask
 3: **for** each patch $p_{t,i}$ in P_t **do**
 4: $p_{t-1,i} \leftarrow \text{PrevPatchCache}[i]$
 5: **if** $\text{CosSim}(p_{t,i}, p_{t-1,i}) \leq \tau$ **then**
 6: SparsityMask $[i] \leftarrow \text{ACTIVE}$
 7: **else**
 8: SparsityMask $[i] \leftarrow \text{REDUNDANT}$
 9: **end if**
 10: CurrPatchCache $[i] \leftarrow p_{t,i}$
 11: **end for**
 12: **return** SparsityMask, CurrPatchCache

First, a lightweight decision mechanism determines, which image regions require recomputation. For each incoming frame I_t , we compare its patches against those from the preceding frame I_{t-1} using the cosine similarity index. The input frame is divided into a regular grid of nonoverlapping patches of size 32×32 pixels; we denote the i th patch of the frame at time t as $p_i(t) \in \mathbb{R}^{3 \times 32 \times 32}$. Each patch is flattened into a vector, and its cosine similarity to the corresponding patch from the previous frame $p_i(t-1)$ is calculated as

$$\text{Similarity}(p_i) = \frac{p_i(t) \cdot p_i(t-1)}{|p_i(t)| |p_i(t-1)|}. \quad (1)$$

We define a high similarity threshold, τ . If $\text{Similarity}(p_i) > \tau$, the block is classified as static (REDUNDANT). Otherwise, it is considered dynamic (ACTIVE). In Algorithm 1, we use the shorthand $\text{CosSim}(p_{t,i}, p_{t-1,i})$ to denote this same cosine similarity operation between the i th patches of consecutive frames. We chose cosine similarity because it is inherently robust to uniform brightness changes. This logic for generating a SparsityMask based on interframe similarity is formally presented in Algorithm 1.

Regarding robustness to AAV vibrations: Modern consumer and professional AAVs employ active gimbal stabilization systems that effectively eliminate high-frequency mechanical vibrations from the video output. Across all datasets evaluated in this work, including our Floodwater dataset and public benchmarks, the captured footage exhibits smooth, stabilized motion consistent with current-generation drone hardware. As a result, interframe pixel displacements are dominated by intentional camera motion rather than mechanical vibrations, and the cosine similarity metric operates reliably without requiring any explicit motion compensation or preprocessing.

Once the SparsityMask is generated, it governs the data flow within the model’s forward pass. A critical component enabling this sparse processing is our BlockSkip module, a modification of the original BlockCopy mechanism tailored for temporal

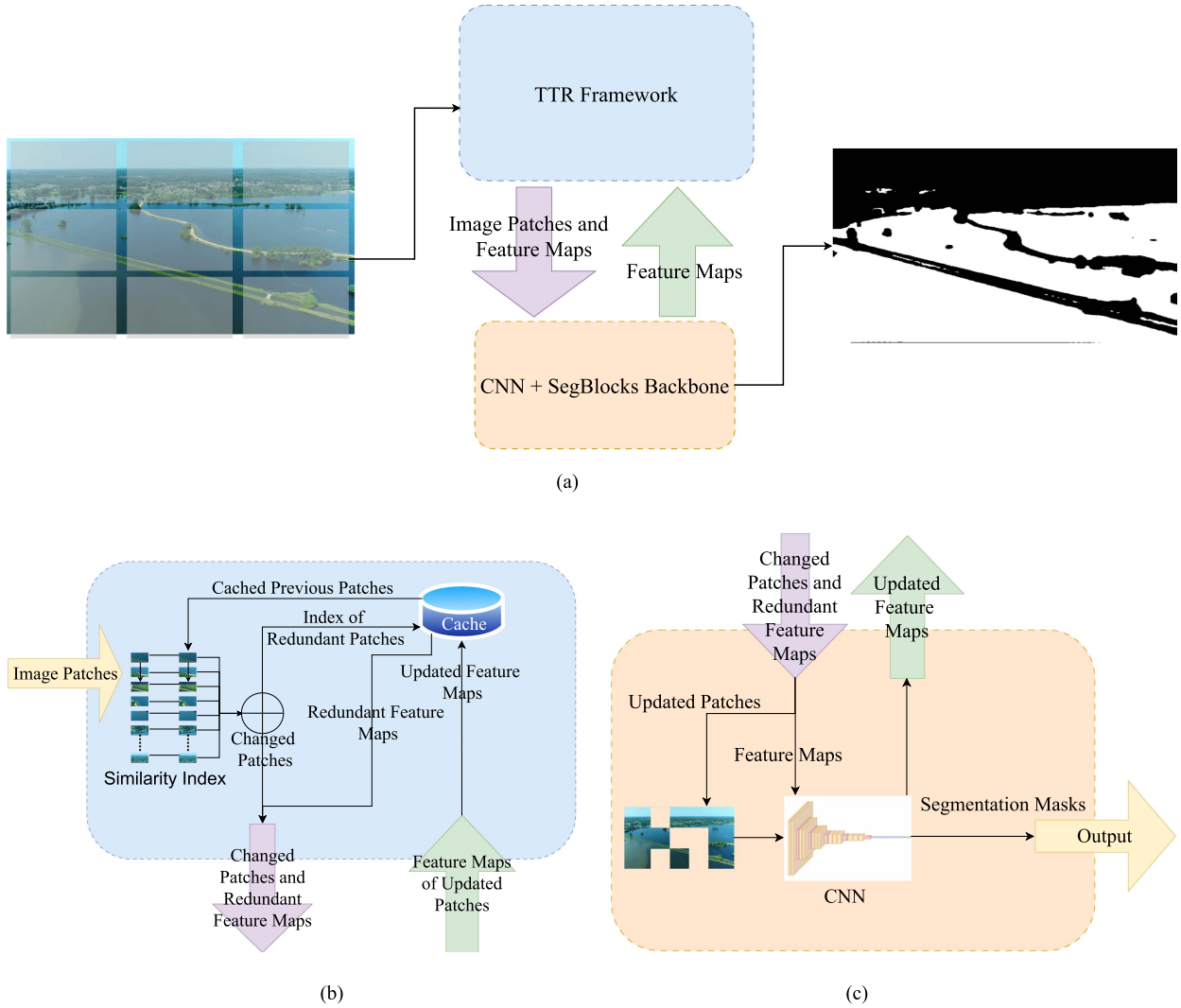


Fig. 2. Detailed operational diagram of the TTR framework. The data flow bifurcates based on patch similarity: 1) Changed patches undergo full CNN processing, with the new features used to update the cache. 2) For redundant patches, features are retrieved directly from the cache, bypassing the CNN. The final mask is assembled from both new and reused features. (a) High-level conceptual overview of the proposed framework. The input is processed collaboratively by the TTR Module and the main CNN + SegBlocks Backbone, which generates the final segmentation mask. (b) TTR Framework. (c) CNN + Segblocks.

reuse. Its purpose is to preserve spatial context across the sparsely computed feature maps. For each ACTIVE patch about to be processed by a convolutional layer, BlockSkip reconstructs its local neighborhood by padding it with features from its neighbors. If a neighboring patch is also ACTIVE, its features are sourced from the current computational graph. However, if a neighbor is REDUNDANT, its features are retrieved directly from the corresponding location in the previous frame's feature cache for that specific layer. By simulating a dense feature map locally, BlockSkip allows standard convolutional layers to operate correctly on sparse inputs, stitching together newly computed and reused features and preventing blocking artifacts that would otherwise degrade segmentation quality. This layer-wise process is detailed in Algorithm 2.

The feature cache aids the computational savings of TTR. This is not a single cache of final predictions, but a collection of deep feature tensors, one for each stage of the network. This process forms a continuous cycle: the fully assembled feature

maps (CurrFeatureCaches) generated during the forward pass of frame I_t become the PrevFeatureCaches for the subsequent frame I_{t+1} , ensuring the system is always ready for the next comparison. To prevent feature drift over extended sequences, a maximum reuse age of $D = 10$ frames is enforced. If a region remains classified as REDUNDANT for D consecutive frames, it is forcibly marked as ACTIVE, triggering a full recomputation and cache refresh regardless of its similarity score. This strategic coupling of a feature cache with sparse input processing results in an efficient and adaptive computational workflow. This design provides a mechanism for runtime control, allowing the computational budget to be modulated through the threshold parameter τ .

Crucially, the memory overhead of this cache is modest. Table I reports the computed cache size for each TTR-enhanced backbone at our default 512×512 input resolution. The feature cache stores one float32 tensor per backbone stage at that stage's spatial resolution and channel count; the patch cache retains the

Algorithm 2: TTR-Enhanced Forward Pass.

Require: Frame I_t , CNN Model with BlockSkip
Require: SparsityMask from Algorithm 1
Require: Prev. Feature Caches: PrevFeatCaches
Ensure: Mask M_t , Updated Caches: CurrFeatCaches

- 1: Initialize CurrFeatCaches
- 2: $x \leftarrow \text{FeatureMap}(I_t)$
- 3: **for** each stage l in Model.Stages **do**
- 4: $x_{\text{act}} \leftarrow \text{GetActive}(x, \text{SparsityMask})$
- 5: $x_{\text{pad}} \leftarrow \text{BlockSkip}(x_{\text{act}}, \text{PrevFeatCaches}[l], \text{SparsityMask})$
- 6: $f_{\text{new}} \leftarrow \text{Model.Stages}[l](x_{\text{pad}})$
- 7: $x_{\text{reuse}} \leftarrow \text{GetRedundant}(\text{PrevFeatCaches}[l], \text{SparsityMask})$
- 8: $x \leftarrow \text{Assemble}(f_{\text{new}}, x_{\text{reuse}}, \text{SparsityMask})$
- 9: CurrFeatCaches[l] $\leftarrow x$
- 10: **end for**
- 11: $M_t \leftarrow \text{Model.FinalLayer}(x)$
- 12: **return** M_t , CurrFeatCaches

TABLE I
TTR FEATURE CACHE MEMORY OVERHEAD PER BACKBONE (512×512 INPUT, FLOAT32, BATCH SIZE 1)

Backbone	St.	Output Channels	Feat.	Patch	Total
ResNet-50	4	256, 512, 1024, 2048	30 MB	3 MB	33 MB
EfficientNet-B4	5	24, 32, 56, 160, 448	10 MB	3 MB	13 MB

Feature cache sizes are computed from each stage's output dimensions ($C \times H \times W \times 4$ bytes).

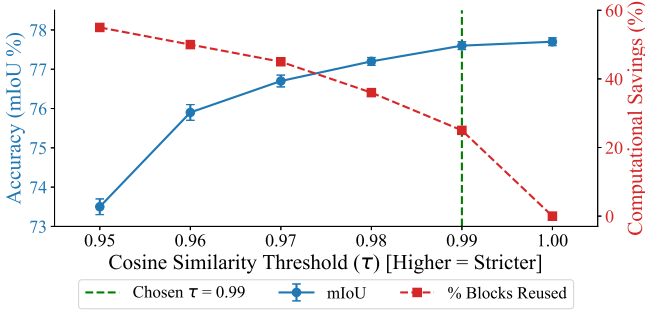


Fig. 3. Tradeoff between segmentation accuracy (mIoU, blue) and computational savings (% blocks reused, red) in function of the cosine similarity threshold. As the threshold becomes stricter (moving right), accuracy improves at the cost of reduced savings. We selected $\tau = 0.99$ as our operating point, offering a strong balance between high performance and significant computational reduction.

previous frame's raw image ($3 \times 512 \times 512$, float32) for cosine similarity comparison. Even in the worst case (ResNet-50), the total cache of 33 MB constitutes less than 0.4% of the Jetson Orin Nano's 8 GB shared memory and less than 10% of the model's own inference memory footprint.

The impact of the threshold hyperparameter τ is analyzed in Fig. 3. It acts as a control knob, allowing a user to balance the system between peak efficiency and maximum accuracy dynamically. For a rapid initial damage assessment, an operator could use a lower τ to maximize speed. Conversely, for a detailed survey mission, a higher τ would ensure higher fidelity. This

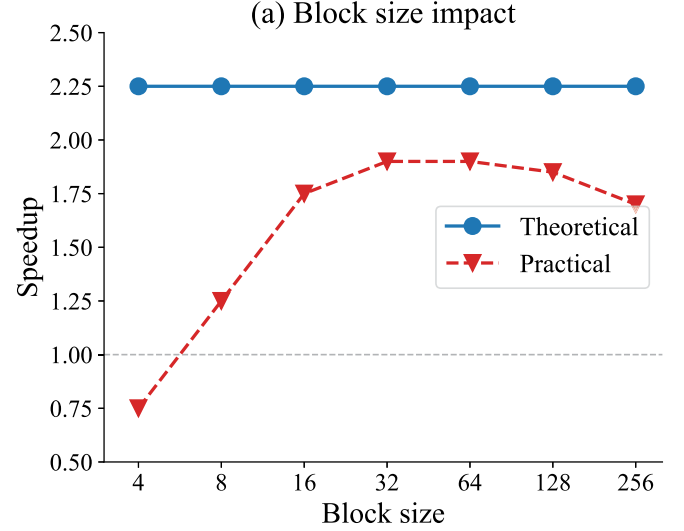


Fig. 4. Impact of block size on inference speedup. The theoretical speedup (blue) remains constant as it depends solely on the proportion of redundant patches. The practical speedup (red) peaks at a block size of 32×32 pixels, which offers the best balance between fine-grained spatial adaptivity and computational overhead. Smaller blocks incur excessive overhead from similarity computation and feature management, while larger blocks lose granularity as mixed static/dynamic patches must be fully recomputed.

highly favorable tradeoff, where a significant boost in frames per second (FPS) is achieved for a small drop in mean intersection over union (mIoU), makes TTR a compelling solution for resource-constrained devices.

The choice of block (patch) size is a critical design parameter, and its impact on practical inference speedup is analyzed in Fig. 4. While the theoretical speedup, which depends only on the fraction of redundant patches identified by the similarity check, remains constant across all block sizes, the practical speedup exhibits a clear peak at 32×32 pixels. This divergence is explained by two competing effects. At very small block sizes (e.g., 4×4 or 8×8), the number of patches per frame grows quadratically, inflating the overhead of per-patch similarity computation, sparsity mask construction, and the gather/scatter operations required to assemble sparse batches. This overhead dominates any savings from skipping small computation units. At the other extreme, large block sizes (e.g., 128×128 or 256×256) reduce the spatial granularity of the redundancy decision: a single moving object within an otherwise static block forces the entire block to be recomputed, wasting the savings that finer partitioning would capture. The 32×32 block size strikes an effective balance, providing patches large enough to amortize the management overhead while remaining small enough to tightly conform to spatial boundaries between static and dynamic regions. We adopt this as the default block size for all experiments reported in Section IV.

IV. RESULTS

A. Datasets

To align with the operational requirements of real-time Earth observation, our evaluation prioritizes domain-specific efficacy

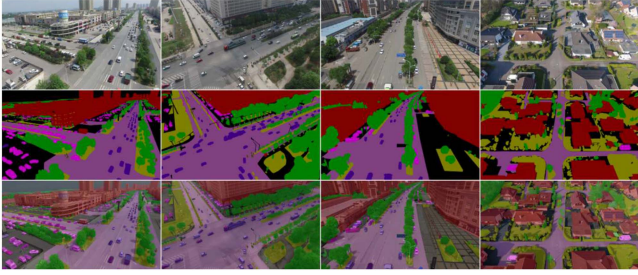


Fig. 5. Visualizations of the AAVid dataset. For each scene, the top row shows the original aerial RGB image. The middle row provides the corresponding pixel-level ground truth annotation provided by the dataset. The bottom row displays the segmentation masks laid on top of the RGB images.

in disaster monitoring before establishing broader architectural robustness. We commence with a targeted analysis using the FloodNet dataset, a standardized benchmark for postdisaster damage assessment, alongside our novel Floodwater dataset. While FloodNet validates the model’s fidelity in classifying complex disaster features against community standards, the Floodwater dataset curated for real-time aerial scouting demonstrates TTR’s efficiency in the precise operational oblique video scenarios for which it was designed. Subsequently, to confirm the framework’s generalizability across varying sensing conditions, we extend our evaluation to disparate public benchmarks: the aerial AAVid dataset, assessing performance in dense urban environments, and the ground-level Audi Autonomous Driving Dataset (A2D2) dataset, serving as a stress test for high-motion, low-redundancy limits.

1) *AAVid Dataset*: Our first benchmark is the AAVid dataset [47], a benchmark for aerial urban scene segmentation. This dataset comprises high-resolution video sequences (up to 4 K) captured by AAVs, featuring dense, pixel-level annotations across 8 distinct classes, such as buildings, roads, vegetation, and moving vehicles. The cluttered urban landscapes of AAVid, illustrated in Fig. 5, introduce significant semantic and visual complexity. The distinction between static and dynamic elements is often little and occurs at a fine spatial scale, providing a rigorous test for the core mechanism of our temporal reuse strategy.

For TTR to succeed here, its lightweight change detection must be sensitive enough to identify small dynamic objects (e.g., individual cars) while robustly classifying large static structures for caching, even under varying lighting. Demonstrating efficiency gains on AAVid validates that TTR’s fine-grained, patch-level operation is effective in complex, real-world aerial monitoring scenarios, confirming its applicability for tasks like urban traffic analysis and infrastructure monitoring.

2) *A2D2 Dataset*: To push our framework to test beyond the aerial domain, we extend our evaluation to the A2D2 [48]. This large-scale dataset provides video from a forward-facing vehicle camera, annotated with over 15 semantic classes. The inclusion of A2D2 represents a shift in our evaluation, moving from a top-down, relatively stable viewpoint to a ground-level perspective characterized by near-constant camera motion. This continuous forward movement induces a global optical flow where nearly

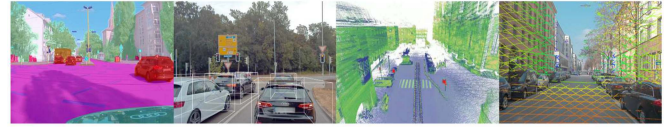


Fig. 6. Visualizations of the A2D2 dataset. From left to right: semantic segmentation, 3-D bounding boxes, dense point cloud from SLAM, single frame point cloud overlaid on corresponding camera image.

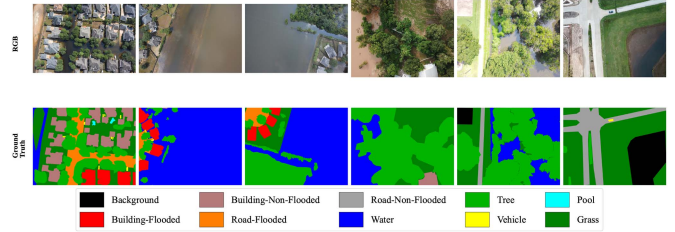


Fig. 7. Sample images from the FloodNet dataset [49]. Each row shows a AAV image captured after Hurricane Harvey alongside its dense semantic annotation. The dataset contains 10 classes, including fine-grained flood-related categories, such as flooded building, flooded road, and nonflooded building, whose visual similarity makes accurate segmentation challenging.

every pixel shifts between frames, creating a challenging low-redundancy environment that directly tests the limits of temporal caching.

This scenario serves as a crucial stress test for the sensitivity of our similarity metric. Demonstrating a significant performance boost on A2D2 (see Fig. 6) would prove that TTR is not just an optimization for static scenes but a robust, general-purpose technique. It validates that our approach can adaptively exploit whatever degree of temporal redundancy exists in a given video stream, making it a viable solution for a broader class of mobile robotics and vision applications where camera motion is persistent.

3) *FloodNet Dataset*: To validate our method against established community standards for postdisaster assessment, we incorporate the FloodNet dataset (see Fig. 7). FloodNet is a high-resolution aerial imagery benchmark captured after Hurricane Harvey, containing precise annotations for classes, such as “flooded building,” “flooded road,” and “nonflooded building.” Unlike the video-based datasets in our evaluation, FloodNet consists of independent images without temporal continuity between frames; it, therefore, serves as a control condition where no genuine temporal redundancy exists for TTR to exploit. While FloodNet is primarily an image-based benchmark, applying our video-based TTR framework to simulated sequences from this dataset allows us to verify that our method performs robustly on the specific semantic classes required for damage assessment, ensuring that the efficiency gains of TTR do not come at the cost of recognizing critical disaster features.

4) *Floodwater Dataset*: Floodwater segmentation is an ideal example of a real-life application for which TTR is relevant. The information needs to be available as fast as possible, while resource consumption should be minimized in order to maximize the flight time and, thus, the area covered in a single



Fig. 8. Visual diversity and challenges within the Floodwater dataset. This selection of frames highlights the complex scenarios for aerial semantic segmentation, ranging from flooded urban infrastructure (top-left, bottom-middle) to inundated agricultural fields (middle row). The dataset captures varying environmental conditions, from clear, sunny days with reflections on the water to overcast and stormy weather (top-right, bottom-left). This validates the dataset as a benchmark for developing and testing real-world flood monitoring systems.

flight. A significant challenge in developing and benchmarking robust flood mapping algorithms for AAVs is the scarcity of large-scale, publicly available video datasets with dense semantic annotations. While several existing AAV-based datasets are available [49], [50], [51], [52], they are often limited in scope for our specific use case. These limitations include low resolution, a lack of diversity in water conditions (e.g., varying turbidity, reflections), (nearly) perpendicular camera angles, though in practice, drones are mostly flown in oblique viewing mode, and critically, a scarcity of dense, frame-by-frame annotations for continuous video data. The process of manual, frame-by-frame semantic annotation for high-resolution video is notoriously labor-intensive, time-consuming, and cost-prohibitive, which explains the current data gap.

To address this gap and to provide a realistic benchmark for our TTR framework, we have curated a new dataset, which we term the floodwater dataset (see Fig. 8). We collected 30 *distinct video sequences* of flood-affected regions in Belgium from publicly available sources. These videos range from 2 to 20 min. in duration and were captured at high resolutions varying from 720p to 4 K, providing the fine-grained detail necessary for precise segmentation. The sequences were recorded at different flying speeds, resulting in a mix of slow-panning and fast-changing imagery, which allows for a robust evaluation of our model's ability to adapt to varying levels of temporal change. The dataset is partitioned into training, validation, and testing splits at the video level to prevent data leakage: 22 *videos for training* (68 chunks, 159 848 frames), 4 *for validation* (13 chunks, 30 017 frames), and 5 *for testing* (17 chunks, 41 159 frames), with all temporal chunks from the same source video assigned to the same split.

To overcome the annotation bottleneck, we leverage the segment anything model 2 (SAM2) [53], a state-of-the-art foundation model for interactive segmentation, within a semisupervised labeling pipeline. An annotator manually reviews the entire video sequence, identifying all visible water bodies by clicking positive and negative point prompts directly on the frames. SAM2 uses these prompts to generate initial segmentation masks and then propagates them temporally through the video using its

built-in video tracking capabilities, producing dense frame-level annotations from sparse human input. The annotator verifies the propagated masks across the full sequence; frames where tracking visibly fails, such as during abrupt scene transitions and rapid camera movements, are corrected with additional point prompts and repropagated. This iterative process yields dense, frame-level binary (flood/nonflood) pseudolabels for the complete video, turning our diverse but unlabeled video collection into a valuable, ready-to-use benchmark dataset.

The Floodwater dataset serves as the focus benchmark for our research. Its combination of high-resolution video and significant temporal redundancy (i.e., large static areas in slow-panning shots) creates the exact computational challenge that TTR is designed to solve. It allows us to directly measure performance gains in a realistic application and validate the suitability of our approach for real-time flood detection and damage assessment on resource-constrained AAVs. A key contribution of this dataset is the generation of dense, pixel-level pseudolabels for the critical binary classification of flooded versus nonflooded areas in video sequences. To foster further research in this critical domain, the dataset is made available at <https://github.com/decide-ugent/floodwater-dataset>.

B. Baseline and Experimental Architectures

All segmentation models were trained using the AdamW optimizer with an initial learning rate of 1×10^{-4} and a weight decay of 1×10^{-4} . The learning rate is adjusted using a ReduceLROnPlateau scheduler monitoring validation Dice score, with a reduction factor of 0.5 and a patience of 4 epochs. Training runs for a maximum of 30 epochs with early stopping triggered after 8 consecutive epochs without improvement in validation Dice score, with a minimum of 12 epochs before early stopping is enabled. The loss function is a composite of binary cross-entropy with logits and Dice loss, weighted equally. Input images are resized to 512×512 pixels and normalized using ImageNet statistics. Training augmentations include random horizontal flips, affine transformations (rotation up to $\pm 10^\circ$, translation up to $\pm 5\%$, scale $0.95\text{--}1.05\times$), and color jitter. No augmentations are applied during validation or testing. Batch sizes are set to 8 for ResNet-50 and 4 for deeper variants (ResNet-101, ResNet-152), with 4 data loading workers. The TTR mechanism is applied exclusively at inference time and does not alter the training procedure. The input image is divided into a grid of patches, where each patch is treated as an independent element in a batch. TTR selectively processes only the ACTIVE patches through the backbone, while REDUNDANT patches skip computation entirely and have their features retrieved from a layer-wise cache. The BlockSkip module enables local feature sharing across neighboring patches, ensuring spatial continuity at patch boundaries. As a result, the backbone always operates on fully contextualized dense inputs and is entirely unaware of the sparsity introduced by TTR.

To evaluate the performance and versatility of our proposed TTR framework, we establish a set of baselines representing diverse architectural paradigms and select standard backbones for integration. Our evaluation strategy is twofold: first, we compare our final TTR-enhanced models against existing segmentation

TABLE II
PERFORMANCE ON FLOODWATER, MANUALLY ANNOTATED FLOODWATER, FLOODNET, AND AIFLOODSENSE DATASETS

Dataset	Model	mIoU (%)	Pix. Acc. (%)	Inference Speed	
				FPS (1080 Ti)	FPS (Orin Nano)
Floodwater	SegFormer-B0	82.0	92.0	25±1	12±1
	EfficientNet-B4	78.0	88.0	30±1	15±1
	EfficientNet-B4 + TTR	77.6	87.8	50±4	25±3
	ResNet-50	74.0	83.0	22±1	11±1
	ResNet-50 + TTR	73.8	82.6	35±3	18±3
	BiSeNetV2	70.0	81.0	65±1	32±1
	MobileNetV3	68.0	76.0	80±1	40±1
Floodwater (Manual)	SegFormer-B0	76.3	85.6	25±1	12±1
	EfficientNet-B4	72.5	81.8	30±1	15±1
	EfficientNet-B4 + TTR	72.2	81.7	50±4	25±3
	ResNet-50	68.8	77.2	22±1	11±1
	ResNet-50 + TTR	68.6	76.8	35±3	18±3
	BiSeNetV2	65.1	75.3	65±1	32±1
	MobileNetV3	63.2	70.7	80±1	40±1
FloodNet	SegFormer-B0	66.5	87.2	25±1	12±1
	EfficientNet-B4	62.1	84.5	30±1	15±1
	EfficientNet-B4 + TTR	62.1	84.5	29±3	14±3
	ResNet-50	56.5	80.4	22±1	11±1
	ResNet-50 + TTR	56.5	80.4	21±3	10±3
	BiSeNetV2	54.2	79.0	65±1	32±1
	MobileNetV3	51.0	76.5	80±1	40±1
AIFloodSense	SegFormer-B0	85.66	92.48	23±1	13±1
	EfficientNet-B4	80.74	89.21	31±1	14±1
	EfficientNet-B4 + TTR	80.71	89.16	30±3	13±3
	ResNet-50	76.29	84.31	21±1	12±1
	ResNet-50 + TTR	76.24	84.27	20±3	11±3
	BiSeNetV2	75.26	85.18	64±1	33±1
	MobileNetV3	72.01	82.61	81±1	41±1

Models are ordered by descending baseline mIoU within each dataset. FPS is reported as mean ± std over 5 inference runs. Accuracy metrics are consistent across hardware. The Bold values put separate baseline and the proposed method in the paper. The paper contribution (TTR) is in bold.

networks; second, we integrate TTR into common backbones to demonstrate its direct impact and compatibility.

For our baseline comparisons, we select three architectures. First, we include SegFormer [3] as a representative of the current state-of-the-art transformer-based models. Given the dominance of transformers in many computer vision tasks, it is crucial to benchmark our CNN-based approach against a leading transformer to contextualize its performance within the broader landscape of modern segmentation networks. Second, to represent architecturally efficient designs, we select MobileNetV3 [54] and BiSeNetV2 [55]. These models serve as critical baselines as they are highly optimized for real-time performance on resource-constrained hardware and are widely adopted in edge applications, the primary target domain for our work.

To demonstrate the direct impact and architectural compatibility of our TTR mechanism, we integrate it into two widely adopted and foundational CNN backbones: ResNet [56] and EfficientNet [57]. The choice of these backbones is deliberate. Our patch-based caching and reuse strategy is designed to operate on feature maps produced by standard convolutional layers. Architectures that heavily rely on specialized operators like atrous (dilated) convolutions to expand their receptive field, such as DeepLab [58], can disrupt the direct spatial correspondence between image patches and their resulting feature representations, which would complicate our feature sharing between patches. ResNet and EfficientNet, with their conventional convolutional structures, provide a clean and direct testbed to isolate and measure the efficiency gains attributable solely to our temporal reuse strategy.

C. Quantitative Performance

In the following sections, we present a quantitative analysis of segmentation accuracy, measured by mIoU and pixel accuracy, and computational efficiency, measured in FPS. All models were benchmarked on two hardware platforms: a high-end NVIDIA GeForce GTX 1080 Ti for desktop performance and a power-constrained NVIDIA Jetson Orin Nano to simulate a real-world edge deployment scenario. The models enhanced with our method are denoted with “(TTR).” This allows for a clear evaluation of TTR’s effectiveness on standard, off-the-shelf architectures while comparing the final, optimized models against leading purpose-built, and transformer-based solutions.

1) *Performance on Floodwater, FloodNet, and AIFloodSense Datasets:* The performance of our TTR framework on the hydrological datasets, detailed in Table II, confirms both its operational efficiency and semantic robustness.

On the *Floodwater Dataset*, which represents our primary target application for real-time monitoring, TTR delivers substantial speed gains. This dataset is characterized by large, static regions of land and water, providing an ideal scenario for exploiting temporal redundancy. When applied to EfficientNet-B4, as seen in Table II TTR increases inference speed by a factor of $1.67\times$ on both the high-end GTX 1080 Ti (from 30 to 50 FPS) and the resource-constrained Jetson Orin Nano (from 15 to 25 FPS). This significant acceleration is achieved with a minimal fidelity tradeoff, incurring a negligible mIoU drop of only 0.4 percentage points. The impact on the Jetson platform is particularly critical; by elevating a high-accuracy model like EfficientNet-B4 from a nonreal-time 15 FPS to a deployable 25

FPS, TTR pushes it into a viable real-time operational window for edge devices.

To assess whether the segmentation results depend on the SAM2-generated pseudolabels, we additionally evaluated the trained models on a manually annotated subset of Floodwater. The 700 frames were held out from both the training and validation sets, and the models were not fine-tuned on these manual annotations. The frames were manually labeled independently, without using SAM2 masks, and were used exclusively for evaluation. The subset contains 700 frames extracted from six videos and covers representative floodwater conditions, including urban scenes, agricultural areas, water reflections, and varying illumination. This evaluation is intended primarily to assess segmentation accuracy against manually produced masks; it is not intended to provide an additional benchmark of temporal coherence, because the subset was selected for annotation verification rather than for characterizing long-term temporal behavior.

The results are reported in Table II under “Floodwater (Manual).” EfficientNet-B4 achieves an mIoU of 72.5% and a pixel accuracy of 81.8%, while EfficientNet-B4 with TTR obtains 72.2% mIoU and 81.7% pixel accuracy. ResNet-50 with TTR similarly decreases mIoU by only 0.2 percentage points relative to the dense model. These results provide an accuracy check against manually annotated masks rather than only against the SAM2-propagated pseudolabels. The small differences between the dense and TTR models indicate that feature reuse does not substantially degrade segmentation accuracy on this manually evaluated subset. The temporal efficiency results reported elsewhere in this article continue to be based on the full video evaluations.

Complementing this, the results on the *FloodNet* benchmark validate the method’s robustness against community standards for damage assessment. While *FloodNet* is a challenging dataset with fine-grained class distinctions (e.g., distinguishing “flooded road” from “nonflooded road”), TTR demonstrates remarkable stability. As shown in Table II, the EfficientNet-B4 + TTR model maintains the exact same accuracy as the dense baseline (62.1% mIoU). For reference, the original *FloodNet* paper [49] reports baseline mIoU scores of 42.61% (ENet), 61.53% (DeepLabV3+), and 79.69% (PSPNet with ResNet-101); our results fall within this published range, confirming the dataset’s inherent difficulty due to its fine-grained, class-imbalanced annotations. This result confirms that the sparsity introduced by our token reuse mechanism does not compromise the recognition of critical disaster features. Even when the efficiency gains are modest as *FloodNet* sequences always lack the temporal continuity of raw drone flight footage the framework ensures that segmentation quality remains preserved, validating TTR as a safe, reliable choice for mission-critical analysis.

2) *Evaluation on the AIFloodSense Dataset:* As an additional external evaluation, we tested the segmentation models on the AIFloodSense dataset [11]. This dataset provides diverse aerial flood imagery and, therefore, enables an assessment of model behavior under a broader range of geographic and environmental conditions. The results obtained with the same model configurations and hardware platforms used in the main experiments are reported in Table II.

The EfficientNet-B4 model achieves an mIoU of 80.74% and a pixel accuracy of 89.21%. When TTR is enabled, it retains near-identical segmentation accuracy, with an mIoU of 80.71% and a pixel accuracy of 89.16%, corresponding to decreases of 0.03 and 0.05 percentage points, respectively. ResNet-50 with TTR shows similarly small decreases, from 76.29% to 76.24% mIoU and from 84.31% to 84.27% pixel accuracy. Because AIFloodSense consists of independent aerial images rather than temporally continuous video sequences, it does not provide temporal redundancy for TTR to exploit. Accordingly, the TTR variants provide no throughput advantage in this setting: EfficientNet-B4 changes from 31 to 30 FPS on the GTX 1080 Ti and from 14 to 13 FPS on the Jetson Orin Nano, while ResNet-50 changes from 21 to 20 FPS and from 12 to 11 FPS, respectively. This result serves as an image-level control experiment: TTR retains nearly the same segmentation accuracy when temporal redundancy is absent, while its computational benefits are reserved for continuous video streams.

3) *Performance on AAVid and A2D2 Datasets:* To validate the robustness of our method, we evaluated on established benchmarks that have complex, real-world conditions. We selected the AAVid dataset, a prominent academic benchmark for aerial urban scene understanding. Featuring 8 distinct classes in diverse environments, this dataset challenges the framework to correctly distinguish between static infrastructure and dynamic objects like moving vehicles from an aerial perspective.

The results in Table III confirm that the benefits of TTR generalize effectively to this multiclass, real-world-inspired environment. On the 1080 Ti, EfficientNet-B4 with TTR jumps from 32 FPS to 54 FPS (a $1.69\times$ speedup), while the mIoU decreases by only 0.3%. A consistent speedup is also observed on the Jetson Orin Nano, where the frame rate is boosted from 16 to 27 FPS. This demonstrates the validity of our lightweight similarity metric, which effectively identifies static elements like buildings and roads for caching, thereby confirming TTR’s applicability to a wide range of aerial segmentation tasks.

Next, to stress-test our approach in a fundamentally different and more demanding context, we transitioned from an academic aerial benchmark to a dataset representing a direct, complex real-world application: autonomous driving. We utilized the A2D2 dataset, which captures the challenges of constant ego-motion and dense traffic from a ground-level view. This scenario presents a significant challenge, as the proportion of the scene that is truly static is much smaller than in typical aerial footage.

Even in this highly dynamic environment, TTR delivered substantial efficiency gains. The EfficientNet-B4 model was accelerated from 25 to 42 FPS on the 1080 Ti ($1.68\times$ speedup) and from 13 to 21 FPS on the Orin Nano ($1.62\times$ speedup), all with a minimal mIoU drop of only 0.3%. The fact that a significant speedup was still achieved proves that TTR is not confined to scenarios with vast static backgrounds. Instead, it robustly adapts to the level of dynamism in the scene, caching features for momentarily static elements (e.g., distant buildings, parked cars) while prioritizing computation for moving ones. This adaptability is critical for real-world deployment.

The overall impact of TTR on the accuracy-efficiency tradeoff across these datasets is visualized for the Nvidia 1080 Ti and the

TABLE III
PERFORMANCE ON AAVID AND A2D2 DATASETS

Dataset	Model	mIoU (%)	Pix. Acc. (%)	Inference Speed	
				FPS (1080 Ti)	FPS (Orin Nano)
UAVid	SegFormer-B0	62.3	86.0	25±1	12±1
	EfficientNet-B4	56.0	84.6	32±1	16±1
	EfficientNet-B4 + TTR	55.7	84.1	54±4	27±3
	ResNet-50	53.8	80.1	25±1	12±1
	ResNet-50 + TTR	53.2	79.8	40±3	20±3
	BiSeNetV2	52.0	81.0	65±1	32±1
	MobileNetV3	50.2	77.2	71±1	35±1
A2D2	SegFormer-B0	65.0	85.0	18±1	9±1
	EfficientNet-B4	60.0	80.0	25±1	13±1
	EfficientNet-B4 + TTR	59.7	79.4	42±4	21±3
	BiSeNetV2	58.0	80.0	55±1	28±1
	ResNet-50	54.0	74.0	24±1	12±1
	ResNet-50 + TTR	53.8	73.6	38±3	19±3
	MobileNetV3	52.0	81.0	70±1	35±1

Models ordered by descending baseline mIoU within each dataset. FPS is reported as mean ± std over 5 inference runs. The Bold values put separate baseline and the proposed method in the paper. The paper contribution (TTR) is in bold.

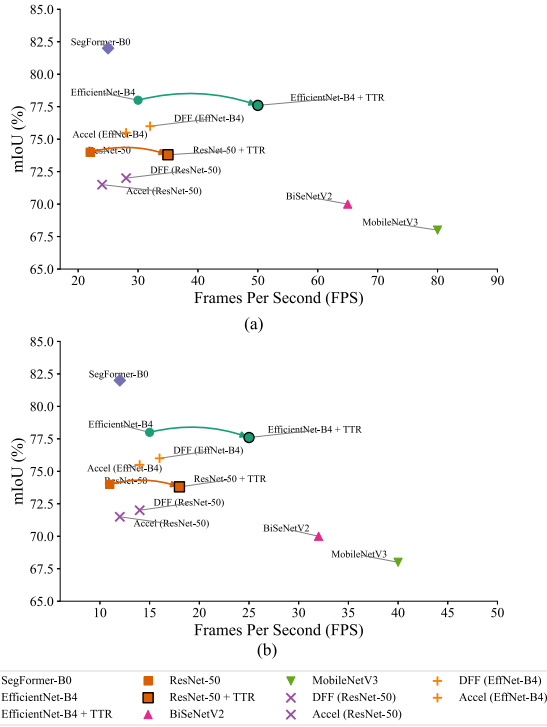


Fig. 9. Accuracy-efficiency tradeoff of different models on the Floodwater dataset, evaluated on (a) NVIDIA GTX 1080 Ti and (b) Jetson Orin Nano. Arrows illustrate the performance shift from baseline models to their TTR-enhanced counterparts, confirming a significant efficiency gain for a negligible accuracy cost.

Jetson Orin Nano in Fig. 9. The plots clearly illustrate how TTR shifts the performance of both ResNet-50 and EfficientNet-B4 toward the ideal top-right corner. Our method enables these more accurate, heavier backbones to achieve FPS rates competitive with much lighter models, creating a new and more favorable Pareto frontier for real-time segmentation on edge devices.

4) *Comparison With Temporal Video Segmentation Methods:* To contextualize TTR within the broader video segmentation literature, we compare against two established CNN-compatible temporal methods: deep feature flow (DFF) [24] and Accel [59].

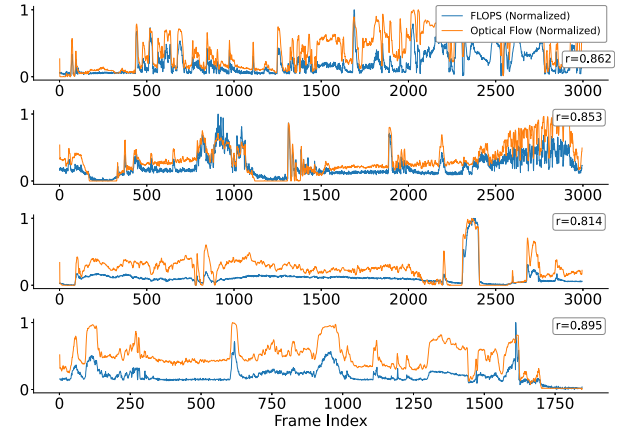


Fig. 10. Correlation between scene dynamism and computational load of our TTR framework. The plots compare the normalized computational load (FLOPS, blue) of our TTR-enhanced model against the normalized magnitude of interframe optical flow (orange) for four representative video sequences from the Floodwater dataset. The strong visual alignment and high Pearson correlation coefficients ($r > 0.8$) demonstrate that TTR effectively adapts its processing, allocating more computational resources during periods of high motion and significantly reducing computation when the scene is static.

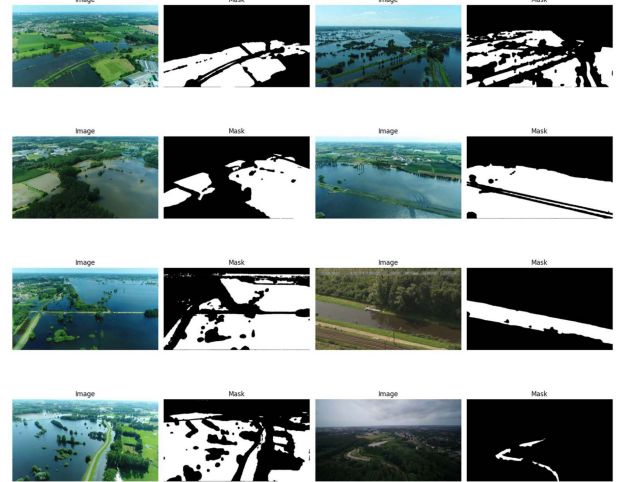


Fig. 11. Visualization of segmentation performance on representative flood-water scenes: left input images; right predicted masks.

TABLE IV
COMPARISON WITH TEMPORAL VIDEO SEGMENTATION METHODS ON THE FLOODWATER AND FLOODNET DATASETS (GTX 1080 Ti)

Method	Backbone	Extra Params	Floodwater		FloodNet	
			mIoU Drop	FPS	mIoU Drop	FPS
Baseline	ResNet-50	—	—	22±1	—	22±1
DFF	ResNet-50	~38M (FlowNet)	2.0%	28±1	3.5%	28±1
Accel	ResNet-50	~49M (FlowNet+R18)	2.5%	24±1	4.0%	24±1
TTR (Ours)	ResNet-50	0	0.2%	35±3	0.0%	21±3
Baseline	EffNet-B4	—	—	30±1	—	30±1
DFF	EffNet-B4	~38M (FlowNet)	2.0%	32±1	3.5%	32±1
Accel	EffNet-B4	~49M (FlowNet+R18)	2.5%	28±1	4.0%	28±1
TTR (Ours)	EffNet-B4	0	0.4%	50±4	0.0%	29±3

FPS is reported as mean ± std over 5 inference runs.

The Bold values put separate baseline and the proposed method in the paper. The paper contribution (TTR) is in bold.

DFF propagates deep features from sparse keyframes to intermediate frames via optical flow warping, requiring a dedicated FlowNet (~38 M parameters) at every nonkey frame. Accel extends this paradigm by adding a lightweight update branch (e.g., ResNet-18) to correct warping errors, further increasing per-frame overhead.

We evaluate DFF and Accel on the Floodwater and FloodNet datasets using the same backbone architectures as our baselines. Table IV summarizes the results on the GTX 1080 Ti.

The comparison reveals three key advantages of TTR. First, TTR introduces *zero additional parameters*, whereas DFF and Accel require dedicated neural networks (FlowNet, and in Accel's case an additional update branch) that increase the model footprint by 38–49 M parameters. On memory-constrained edge devices, this overhead alone can be prohibitive. Second, on the video-based Floodwater dataset, TTR achieves substantially lower accuracy degradation: 0.2%–0.4% mIoU drop compared to 2.0%–2.5% for the flow-based methods, because optical flow warping introduces systematic errors at occlusion boundaries and in textureless regions that compound across frames. Third, TTR delivers higher throughput because its change detection mechanism, a per-patch cosine similarity check, is orders of magnitude cheaper than running a full optical flow network at every non key frame.

The FloodNet results further highlight a critical robustness advantage. Because FloodNet consists of independent images arranged into simulated sequences, there is no genuine temporal coherence for flow-based methods to exploit. Consequently, DFF and Accel suffer amplified accuracy degradation (3.5%–4.0% mIoU drop) as the optical flow computed between unrelated frames introduces erroneous feature warping. In contrast, TTR gracefully degrades to baseline performance, preserving full accuracy with negligible overhead. This demonstrates that TTR is a *safe* inference-time addition: it accelerates processing when temporal redundancy is present and incurs no penalty when it is absent.

It is worth noting that while recent temporal video segmentation research, such as coarse-to-fine feature mining [60], mining relations among cross-frame affinities [61], and deep common feature mining [62], has shifted to transformer-based backbones, CNN backbones remain the practical choice for edge deployment due to their lower memory footprint and mature hardware support. DFF and Accel represent the state of the art for CNN-compatible temporal methods, and TTR advances this

line of work by replacing the expensive optical flow paradigm with a lightweight, training-free alternative.

D. Ablation Studies

To isolate the contribution of individual design choices, we ablate the two primary hyperparameters governing TTR: the similarity threshold τ and the block (patch) size.

1) *Threshold Sensitivity*: The threshold τ directly controls the tradeoff between segmentation fidelity and computational savings, as shown in Fig. 3. A lower threshold classifies more patches as redundant, maximizing reuse and FPS at the expense of accuracy, while a higher threshold preserves accuracy but reduces the proportion of skipped patches. At our selected operating point of $\tau = 0.99$, the EfficientNet-B4 backbone retains over 99% of its baseline mIoU while reusing approximately 60% of patches per frame on the Floodwater test set. The curve exhibits a favorable knee: relaxing the threshold below 0.99 yields rapidly diminishing accuracy returns for only marginal speed gains, confirming $\tau = 0.99$ as a robust default.

2) *Block Size Selection*: The granularity at which TTR partitions each frame into patches has a pronounced effect on practical speedup, as analyzed in Fig. 4. Although the theoretical speedup is invariant to block size, the measured wall-clock speedup peaks at 32×32 pixels on the Floodwater test set using EfficientNet-B4. Smaller blocks suffer from disproportionate management overhead relative to the computation they save, while larger blocks sacrifice spatial precision, forcing recomputation of predominantly static regions that contain even a small dynamic element. The 32×32 configuration maximizes practical throughput and is used throughout all experiments in this work.

V. DISCUSSION

The results presented consistently validate our central hypothesis: exploiting temporal redundancy via token reuse offers an improvement in computational efficiency for a minimal and often negligible cost in accuracy. Across all datasets and hardware platforms, we observed an increase in processing speed, a crucial factor for real-time deployment on resource-constrained platforms like AAVs.

The success of TTR is rooted in the inherent spatio-temporal redundancy of video data, particularly in aerial footage. The strong performance on the Floodwater and AAVid datasets



Fig. 12. Representative challenging inputs from the Floodwater data and the AIFloodSense dataset. The examples include strong water reflections and glare, complex flooded landscapes, vegetation and debris near water boundaries, and substantial variation in scene appearance. Such conditions can reduce patch-level temporal similarity and limit the computational benefit of TTR.

underscores how well our method is suited to these scenarios, where large portions of the frame are often static or slow-moving. By intelligently identifying and skipping redundant computations for these regions, TTR can allocate the AAV’s limited processing power where it is most needed. The choice of cosine similarity as the change detection metric further enhances this robustness by making the system less susceptible to global illumination shifts, a frequent challenge in outdoor environments that can confound simpler difference-based methods.

To quantitatively validate this adaptive behavior, we analyze the direct relationship between scene dynamism and the computational load of our TTR-enhanced model in Fig. 10. In this analysis, we use the magnitude of interframe optical flow as a proxy for scene dynamics; low values signify a static scene (e.g., a hovering AAV), while high values indicate significant motion from either the camera’s ego-motion or objects within the scene. The plots reveal a remarkably tight coupling between the normalized FLOPs executed by our model and the optical flow. The high Pearson correlation coefficients ($r > 0.8$) demonstrate a causal link that our lightweight similarity metric is an effective and efficient proxy for complex motion estimation, successfully guiding the model’s computational budget in real time.

The patterns depicted in the graph provide insights into the video behavior. The troughs, where the computational load drops significantly, correspond to periods of minimal change, such as when the AAV is hovering or panning slowly over a large, uniform body of water. In these moments, TTR achieves maximum efficiency by reusing the vast majority of feature patches from the previous frame. Conversely, the sharp peaks in both optical flow and FLOPs signify moments of high dynamism. These can be triggered by sudden AAV movements like sharp turns or rapid altitude changes, or by the appearance of dynamic objects. The fact that TTR immediately scales up its computation in response to these events is critical, as it confirms the framework’s ability to preserve high-fidelity processing when accuracy is important. This dynamic scaling ensures that TTR is not just an efficiency-saving mechanism but a truly adaptive one that aligns its effort with the temporal complexity of the scene.

Crucially, the metrics in Fig. 10 emerge from TTR’s fine-grained, patch-level decision-making. TTR excels in these low-motion scenarios by independently evaluating each patch, a capability that allows it to focus resources on dynamic regions

TABLE V
ENERGY PER FRAME ON THE JETSON ORIN NANO (15 W TDP, FLOODWATER DATASET)

Model	Base FPS	TTR FPS	Base J/fr	TTR J/fr
EfficientNet-B4	15	25	1.00	0.60
ResNet-50	11	18	1.36	0.83

while saving computation on static ones. This patch-wise adaptability is an advantage over coarse, frame-level methods that process the entire frame if any part of it changes. Importantly, these efficiency gains do not compromise performance. The qualitative analysis in Fig. 11 provides visual confirmation, showing that the predicted masks remain accurate even in complex scenes.

This adaptive resource allocation is important for real-world deployment on AAVs, where energy and onboard compute are strictly limited. By changing its computational footprint in direct proportion to scene complexity, TTR achieves substantial energy savings during common, low-motion periods of operation, extending mission duration. This behavior makes it a robust and reliable solution, ensuring that the system can handle bursts of activity without failure while remaining maximally efficient during low-motion phases, especially for AAVs or hardware-constrained applications.

To quantify these energy savings, Table V reports the energy consumed per frame on the Jetson Orin Nano, which operates under a fixed 15 W TDP power cap. Under sustained inference, both baseline and TTR-enhanced models draw near-identical power at the TDP ceiling, so the efficiency gain manifests directly as reduced energy per frame. For EfficientNet-B4 and ResNet-50, TTR reduces energy per frame by 39%–40%, meaning 67% more frames can be processed within the same energy budget.

VI. CONCLUSION

In this article, we introduced TTR, a framework designed to significantly improve the computational efficiency of video semantic segmentation, with a particular focus on AAV applications. By treating image patches as tokens and reusing the features of those tokens that remain static over time, TTR directly addresses the problem of temporal redundancy inherent in video streams. Our approach successfully marries the spatial adaptability of patch-based CNNs with a new layer of temporal intelligence.

Our primary contribution is the development of the TTR framework and the demonstration of its effectiveness in accelerating video semantic segmentation without substantially compromising accuracy. TTR introduces a principled mechanism for identifying and reusing temporally redundant token features through a lightweight cosine-similarity metric and a deep feature cache. By integrating this temporal reuse strategy directly into a patch-based CNN pipeline, the framework enables substantial gains in inference speed (speedups of $1.58\times$ to $1.69\times$ on video datasets) while incurring only a marginal loss in segmentation accuracy. This work validates that it is not always necessary to

reprocess all spatial content in every frame to maintain high-quality semantic understanding.

Ultimately, TTR represents a significant step toward enabling powerful, real-time perception on autonomous systems, making sophisticated computer vision models more accessible for a new generation of intelligent edge devices.

A. Limitations and Future Research

While TTR significantly reduces the computational latency of on-board segmentation, bridging the gap between pixel-level inference and actionable geospatial intelligence presents several opportunities for future exploration.

A primary limitation of the current framework is that it operates exclusively in the camera's image plane. While oblique imagery provides superior visibility of vertical infrastructure and water levels compared to nadir views, the resulting segmentation masks are inherently subject to perspective distortion and varying scale. For crisis responders, a segmented pixel mask provides qualitative awareness but lacks the direct geospatial context required to assess the precise impact on specific infrastructure.

A promising avenue for future research involves the integration of real-time *orthorectification* directly into the edge inference pipeline. By fusing the AAV's telemetry data (IMU and GPS) with a digital elevation model, future systems could project these oblique segmentation masks onto a georeferenced cartographic plane during flight. This projection would be a critical prerequisite for overlaying flood extents onto standard topographic maps, enabling the automated analysis of *real-time road network connectivity*. Such a workflow would allow the system to autonomously identify severed transit routes and calculate accessible paths for emergency vehicles, thereby directly linking computer vision outputs to the operational needs of crisis response.

In addition, further performance gains could be realized by exploring adaptive mechanisms for the similarity threshold τ . While a fixed threshold proved robust in our experiments, developing a learnable gating policy that adjusts to environmental factors (e.g., flight altitude or weather conditions) could dynamically optimize the tradeoff between sparsity and accuracy without manual calibration.

B. Failure Modes and Operational Boundary

The computational benefit of TTR is conditional on the presence of local temporal redundancy. During abrupt camera rotations, rapid altitude changes, or hard scene transitions, most patches become dissimilar to their cached counterparts. TTR then marks most regions as ACTIVE and approaches the computational cost of dense inference. In these cases, the primary failure mode is the loss of acceleration rather than the continued use of stale features.

Severe or spatially nonuniform illumination changes can also reduce patch similarity even when the underlying scene is unchanged. Although cosine similarity is relatively tolerant to uniform brightness scaling, shadows, exposure changes, glare, and localized lighting variations may trigger unnecessary re-computation. Water reflections, specular highlights, and rapidly changing ripples can similarly make a semantically unchanged water surface appear dynamic. This reduces the proportion of

reusable patches and can create localized discrepancies near water–land boundaries if a changed patch remains below the similarity threshold.

Highly dynamic scenes, including moving vehicles, strong vegetation motion, or rapidly evolving flood boundaries, contain limited temporal redundancy and, therefore, offer little computational benefit. The A2D2 evaluation represents this type of low-redundancy setting, while the independent-image FloodNet and AIFloodSense evaluations provide control conditions in which temporal reuse cannot provide a meaningful speedup. These cases clarify that TTR is intended primarily for continuous oblique-video monitoring with substantial interframe overlap. For abrupt or highly dynamic events, a dense inference pass, a stricter similarity threshold, or an adaptive cache-refresh policy may be preferable.

Future versions of TTR could incorporate an explicit scene-change detector or confidence-aware cache invalidation mechanism. Such a mechanism could trigger a dense refresh when global motion, illumination variation, or disagreement between neighboring patches indicates that cached features may be unreliable. Motion-compensated patch matching and uncertainty-aware reuse are additional directions for improving robustness under large displacements and severe appearance changes.

Fig. 12 illustrates representative input conditions that define the operational boundary of TTR. The panel includes challenging examples from the Floodwater data and the AIFloodSense dataset. Strong reflections and glare can make visually static water surfaces appear to change between frames, while complex vegetation, debris, and irregular water–land boundaries can reduce the number of patches that are safely reusable. These examples illustrate challenging input conditions rather than claiming that every displayed frame produces a segmentation error. Under such conditions, TTR is expected to activate more patches and consequently provide less acceleration than in scenes with stable appearance and high interframe overlap.

Finally, it is important to contextualize the operational scope of TTR. Traditional segmentation models apply the same full computational graph to every frame, operating at 100% capacity regardless of whether the scene content is trivial or complex. This uniform processing is inherently wasteful: during the long, routine phases of a AAV mission where the scene evolves slowly, such as hovering over a static water body, the model expends the same energy and compute as during a rapid camera sweep over a dynamic area. TTR directly addresses this inefficiency by scaling computation to match scene complexity. During “easy” scenes with high temporal redundancy, TTR aggressively reuses cached features, saving significant power and compute. When the scene becomes challenging with rapid changes from camera motion or evolving flood boundaries, TTR automatically returns to full computational capacity, processing all patches to maintain segmentation fidelity.

This characteristic means TTR is best suited for oblique AAV surveillance scenarios, where the camera's viewing geometry produces high interframe overlap and large portions of the scene persist unchanged across consecutive frames. In such stable monitoring contexts, TTR achieves speedups of up to $1.69\times$. Conversely, when temporal redundancy is inherently low, as in orthogonal or nadir acquisitions during active flood events, or on datasets of independent images such as FloodNet,

the proportion of reusable patches diminishes and efficiency gains become marginal. This adaptive behavior is validated by the strong correlation between TTR's computational load and optical flow magnitude (cfr. Fig. 10). TTR should, therefore, be understood as a method optimized for continuous oblique video monitoring, complementary to lightweight architectures or model compression techniques that target frame-independent scenarios.

DECLARATIONS

- 1) *Conflict of interest/Competing interests*: The authors declare no competing interests.
- 2) *Ethics approval and consent to participate*: Not applicable.
- 3) *Consent for publication*: Not applicable.
- 4) *Data availability*: The A2D2, AAVid, and FloodNet datasets are publicly available. The Floodwater dataset is shared.
- 5) *Materials availability*: Not applicable.

ACKNOWLEDGMENT

The authors would like to thank the Flanders Environment Agency (VMM) and Flanders Hydraulics Research for providing the extensive aerial video dataset that was instrumental to this study. Sam Leroux and Pieter Simoens would like to thank the Flanders AI Research Program.

REFERENCES

- [1] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "Can semantic labeling methods generalize to any city? The Inria aerial image labeling benchmark," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2017, pp. 3226–3229.
- [2] Z. Wu, C. Shen, and A. van den Hengel, "Real-time semantic image segmentation via spatial sparsity," 2017, Art. no. arXiv:1712.00213.
- [3] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12077–12090.
- [4] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
- [5] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, arXiv:1605.07678.
- [6] M. Figurnov et al., "Spatially adaptive computation time for residual networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1790–1799.
- [7] Z. Wu et al., "BlockDrop: Dynamic inference paths in residual networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8817–8826.
- [8] A. A. Sundaresan and A. A. Solomon, "Post-disaster flooded region segmentation using DeepLabv3 and aerial system imagery," *Natural Hazards Res.*, vol. 5, no. 2, pp. 363–371, 2025.
- [9] G. Simantiris and C. Panagiotakis, "Unsupervised color-based flood segmentation in UAV imagery," *Remote Sens.*, vol. 16, no. 12, 2024, Art. no. 2126.
- [10] A. Manaf, N. Mughal, K. R. Talpur, B. A. Talpur, G. Mujtaba, and S. R. Talpur, "Aerial image classification in post flood scenarios using robust deep learning and explainable artificial intelligence," *IEEE Access*, vol. 13, pp. 35973–35984, 2025.
- [11] G. Simantiris, K. Bacharidis, A. Papanikolaou, P. Giannakakis, and C. Panagiotakis, "Aifloodsense: A global aerial imagery dataset for semantic segmentation and understanding of flooded environments," *Remote Sens.*, vol. 18, no. 6, 2026, Art. no. 938.
- [12] Y. LeCun, J. S. Denker, and S. A.olla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.
- [13] J.-X. Liu et al., "Unstructured-to-structured pruning: A reverse-thinking paradigm," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 17377–17387.
- [14] D. Zhang, J. Liu, Y. Yang, and C. Zhu, "Data-free knowledge distillation for vision transformers," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2023, pp. 6586–6596.
- [15] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4820–4828.
- [16] W. Li et al., "OmniQuant: A general-purpose post-training quantization toolkit for vision and language models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 20231–20240.
- [17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Representation Learn. Workshop*, 2015, arXiv:1503.02531. [Online]. Available: <https://arxiv.org/abs/1503.02531>
- [18] L. Beyer, X. Zhai, A. Roy, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10924–10933.
- [19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
- [20] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.
- [21] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.
- [22] X. Ding, X. Zhang, Y. Zhou, J. Han, G. Ding, and J. Sun, "Scaling up your kernels to 31x31: Revisiting large kernel design in CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11963–11975.
- [23] S. Mehta and M. Rastegari, "Separable self-attention for mobile vision transformers," *Trans. Mach. Learn. Res.*, Jan. 2023. <https://openreview.net/forum?id=tB14yBEjKi>
- [24] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 424–432.
- [25] Z. Wu, C. Wang, Z. Lin, C. Chen, and X. Wang, "AdaFrame: Adaptive frame selection for fast video recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4899–4908.
- [26] N. u. Gundavarapu, A. Rao, and A. Mishra, "Let's hide behind the query: A new perspective on efficient video processing," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2023, pp. 1672–1682.
- [27] S. Paul, Y. Li, and D. Lin, "Co-training for video semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 15938–15947.
- [28] Y. Li, J. Shi, and D. Lin, "Video semantic segmentation with temporal super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 70–85.
- [29] A. Alfarano, L. Maiano, L. Papa, and I. Amerini, "Estimating optical flow: A comprehensive review of the state of the art," in *Proc. Comput. Vis. Image Understand.*, 2024, vol. 249, Art. no. 104160. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1077314224002418>
- [30] W. Gao, B. Fan, and Y. Fang, "Semantic segmentation algorithm for video from UAV based on adaptive keyframe scheduling via similarity measurement," *Meas. Sci. Technol.*, vol. 35, no. 10, Jul. 2024, Art. no. 106120. [Online]. Available: <https://dx.doi.org/10.1088/1361-6501/ad6026>
- [31] B. Wang, W. Huang, Z. Wang, and R. Li, "AFB-URR: Attention-guided feature blending and uncertainty-aware region refinement for video semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 273–289.
- [32] Z. Zhou et al., "Rethinking video segmentation from a temporal perspective," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 18090–18100.
- [33] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang, "Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6459–6468.
- [34] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNNet: Sparse blocks network for fast inference," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8711–8720.
- [35] J. Liang, Z. Wu, J. Liu, and L. Li, "EViT: An efficient vision transformer with dynamic token spatiofication," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2023, pp. 5556–5565.
- [36] D. Bolya et al., "Token merging: Your ViT but faster," in *Proc. Int. Conf. Learn. Representations*, 2023.
- [37] Y. Tang, K. Han, J. Guo, Y. Wang, and C. Han, "DynamicViT: Efficient vision transformers with dynamic token sparsification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 6144–6154.

- [38] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 420–436.
- [39] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–18.
- [40] Y. Cong, G. Yu, S. Zhang, B. Li, J. Wang, and Z. Li, "PoWER-BERT: Accelerating BERT inference via progressive word-vector elimination," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 10, pp. 12066–12074.
- [41] T. Verelst and T. Tuytelaars, "Dynamic convolutions: Exploiting spatial sparsity for faster inference," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2317–2326.
- [42] S.-C. Li, M. Chen, C.-S. Lai, and A. Lee, "Dynamic convolution for 3D point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1350–1359.
- [43] Z. Sun, Y. Li, Y. Wang, D.-C. Peng, Z.-J. Cheng, and X. Zhang, "Implicit early exiting for vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 5777–5787.
- [44] T. Wu, Z. Lei, B. Lin, C. Li, Y. Qu, and Y. Xie, "Patch proposal network for fast semantic segmentation of high-resolution images," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 07, pp. 12402–12409.
- [45] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 418–434.
- [46] T. Verelst and T. Tuytelaars, "Segblocks: Block-based dynamic resolution networks for real-time segmentation," *CoRR*, vol. abs/2011.12025, 2020. [Online]. Available: <https://arxiv.org/abs/2011.12025>
- [47] Y. Lyu, G. Vosselman, S. Oude Elberink, M. Y. Yang, and Q. Hu, "UAVid: A semantic segmentation dataset for UAV imagery," *ISPRS J. Photogrammetry Remote Sens.*, vol. 165, pp. 108–119, 2020.
- [48] J. Geyer et al., "A2D2: Audi autonomous driving dataset," 2020, *arXiv:2004.06320*.
- [49] M. Rahmemonfar, T. Chowdhury, A. Sarkar, D. Varshney, M. Yari, and R. R. Murphy, "Floodnet: A high resolution aerial imagery dataset for post flood scene understanding," *CoRR*, vol. abs/2012.02951, 2020. [Online]. Available: <https://arxiv.org/abs/2012.02951>
- [50] A. Itkyl, G. Joshi, R. Walambe, and K. Kotecha, "Aid-SAR: Aerial image dataset for search and rescue in the aftermath of disasters," in *Advanced Computing*, V. Garg, S. K. Pendyala Gupta, and M. Najafzadeh, Eds. Cham: Springer Nature Switzerland, 2025, pp. 197–209.
- [51] D. W. Nugraha, A. A. Ilham, A. Achmad, and A. Arief, "Aerial imagery of natural disaster-affected areas (AINDAA) dataset for semantic segmentation and natural disaster assessment," in *Proc. IEEE 7th Int. Conf. Inf. Technol., Inf. Syst. Electrical Engineering*, 2023, pp. 208–213.
- [52] V. Polushko et al., "Blessemflood21: A high-resolution georeferenced dataset for advanced analysis of river flood scenarios," *IEEE Access*, vol. 12, pp. 176389–176405, 2024.
- [53] N. Ravi et al., "Sam 2: Segment anything in images and videos," 2024. [Online]. Available: <https://arxiv.org/abs/2408.00714>
- [54] A. Howard et al., "Searching for MobileNetV3," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
- [55] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation," *Int. J. Comput. Vis.*, vol. 129, no. 11, pp. 3051–3068, 2021.
- [56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [57] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
- [58] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *CoRR*, vol. abs/1606.00915, 2016. [Online]. Available: <http://arxiv.org/abs/1606.00915>
- [59] S. Jain, X. Wang, and J. E. Gonzalez, "Accel: A corrective fusion network for efficient semantic segmentation on video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8866–8875.
- [60] G. Sun, Y. Liu, H. Ding, T. Probst, and L. V. Gool, "Coarse-to-fine feature mining for video semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3126–3137.
- [61] G. Sun, Y. Liu, H. Tang, A. Chhatkuli, L. Zhang, and L. V. Gool, "Mining relations among cross-frame affinities for video semantic segmentation," in *Proc. Computer Vision—ECCV 2022, Lecture Notes Comput. Sci.*, vol. 13694, 2022, pp. 522–539, doi: [10.1007/978-3-031-19830-4_30](https://doi.org/10.1007/978-3-031-19830-4_30).
- [62] Y. Zheng, H. Yang, and D. Huang, "Deep common feature mining for efficient video semantic segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 12, pp. 12991–13003, Dec. 2024.



Vishisht Sharma graduated with a master of science in artificial intelligence from Queen Mary University of London, London, United Kingdom, in 2023 and is currently working toward Ph.D. degree in information engineering technology with the IDLab, Department of Information Technology, Ghent University-imec, Ghent, Belgium, and Flemish Institute for Technological Research (VITO), Mol, Belgium.

His research focuses on hardware aware algorithm design and computer vision.



Sam Leroux graduated with a master in information engineering technology from Ghent University, Ghent, in 2014 and a Ph.D. degree in information engineering technology from Ghent University, Ghent, in 2019, and is currently an Assistant Professor with Ghent University, Ghent, Belgium and is also with imec, Leuven, Belgium. His main research interests include efficient neural network architectures and edge computing.



Lisa Landuyt received the M.S. degree in bioscience engineering from Ghent University, Ghent, Belgium, in 2016, and the Ph.D. degree in bioscience engineering from Ghent University, Ghent, Belgium, in 2021. She is currently a remote sensing researcher at VITO. Her research interests include deep learning applications for crisis management and environmental policy support.



Nick Witvrouwen received the M.S.E.E. degree in electrical engineering from KU Leuven, Leuven, Belgium, in 2001. He worked as an ASIC and image sensor design engineer. He is currently a researcher at VITO, Mol, Belgium, focusing on onboard remote-sensing technology.



Pieter Simoens graduated with a master degree in electrotechnical engineering from Ghent University, Ghent, Belgium, in 2005 and a Computer Science Engineering, Ghent University, Ghent, Belgium, in 2011 and is currently an Associate Professor with Ghent University, Ghent, Belgium, and is also with imec, Leuven, Belgium. His main research interests include the domain of intelligent decentralized and distributed systems, with a specific on resource efficient machine learning, representation learning and decision-making algorithms.