Dual Sparsity Transformer with Contour Loss for Real-Time UAV Image Segmentation

Wen Lu, Minh Nguyen*

***Abstract*—Integrating a semantic segmentation network into an Unmanned Aerial Vehicle (UAV) improves situational awareness and facilitates autonomous operations in dynamic environments. However, designing such a network for onboard deployment is challenging, as it must achieve high performance while maintaining low computational and memory requirements and ensuring real-time processing capabilities. UAVs typically operate at high altitudes, providing broad ground coverage; however, this results in key objects—such as humans, vehicles, and obstacles—appearing smaller in the imagery, thereby complicating their accurate identification. To address these challenges, we propose a lightweight semantic segmentation network and a network-agnostic loss specifically designed for UAV imagery. The Dual Sparsity Transformer (DST) incorporates two forms of sparsity: data-based sparsity, which reduces computational complexity; and content-based sparsity, which filters out irrelevant information to generate more refined aggregated features. The novel loss leverages predicted contours to capture complex patterns, boundaries, and small objects, imposing a higher penalty for misclassifications in these areas. This encourages the network to prioritize the accurate detection of challenging-to-distinguish objects. Our approach exhibits remarkable accuracy and real-time throughput for 4K resolution images on a mobile GPU, highlighting its effectiveness for onboard deployment in UAV systems.**

***Index Terms*—semantic segmentation, lightweight neural network, unmanned aerial vehicle, remote sensing.**

I. Introduction

UAVS, commonly known as drones, are aircraft systems that operate without a human pilot on board, typically controlled remotely or autonomously. Initially developed for military applications, UAVs have rapidly evolved over the past few decades and now serve a wide range of industries, from agriculture and environmental monitoring to infrastructure inspection and disaster management. UAVs are also revolutionizing logistics and delivery services by offering faster, more reliable, and cost-effective solutions. For instance, UAVs are increasingly employed to transport medical supplies—such as vaccines, blood samples, and medications—especially to remote or disaster-affected areas where conventional transportation is slow or inaccessible. Moreover, by bypassing traffic congestion, UAVs enable faster last-mile delivery, improving customer satisfaction; as a result, companies like Amazon and Uber Eats are exploring drone-based systems for parcel and meal delivery.

Integrating semantic understanding directly into a UAV's processing pipeline enhances its ability to perform complex tasks in dynamic and unpredictable environments while reducing dependence on external infrastructure. For example, deploying a real-time semantic segmentation network aboard allows the UAV to differentiate between various terrains (e.g., roads, grass, water bodies) and adjust its flight strategy accordingly, facilitating tasks such as terrain following and landing zone identification. Additionally, onboard semantic segmentation enables the UAV to detect vehicles, pedestrians, buildings, trees, and other low-altitude obstacles in real time, allowing it to dynamically alter its flight path to avoid collisions—particularly important in urban or densely forested areas. By processing semantic information locally, the UAV eliminates the need for constant communication with a ground station or reliance on preloaded maps, enabling more reliable flight in areas with weak or no radio/GPS signal.

Semantic segmentation is a computationally intensive task, as it requires pixel-wise classification. Due to their compact size and light weight, most UAVs cannot accommodate large-capacity batteries or high-power processors. As a result, they typically rely on edge devices such as embedded GPUs, FPGAs, or specialized accelerators, which have limited computational resources and memory. However, general semantic segmentation models typically demand substantial memory and computational power, which are unavailable in embedded environments. Although networks such as BiSeNetV1 [1], BiSeNetV2 [2], DANet [3], and STDC [4] prioritize efficiency and utilize lightweight architectures, they achieve only marginal gains in acceleration and resource usage at the cost of significant accuracy loss—particularly on small objects and boundaries. Consequently, achieving high-quality segmentation while maintaining low inference time is difficult, especially in real-time video processing scenarios, where high frame rates are crucial.

Recent years have witnessed Vision Transformers (ViTs) achieving remarkable results and surpassing Convolutional Neural Networks (CNNs) in various vision tasks [5], [6], [7]. The success of ViTs can be attributed to their ability to capture long-range dependencies through self-attention. However, the vanilla self-attention mechanism models dependencies between all image patches, resulting in a quadratic computational cost relative to the image size. This makes it impractical for onboard processors, particularly when handling high-resolution images. Furthermore, recent studies [8], [9], through visualizing attention map activations, have shown that patches with high attention scores are typically sparse and clustered around the query patches. This suggests that distant patches are often semantically irrelevant, indicating substantial redundancy in global attention and the need for data-based

Wen Lu and Minh Nguyen are with School of Engineering, Computer & Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand (e-mail: wen.lu@autuni.ac.nz, minh.nguyen@aut.ac.nz).

*, Minh Nguyen is the corresponding author.

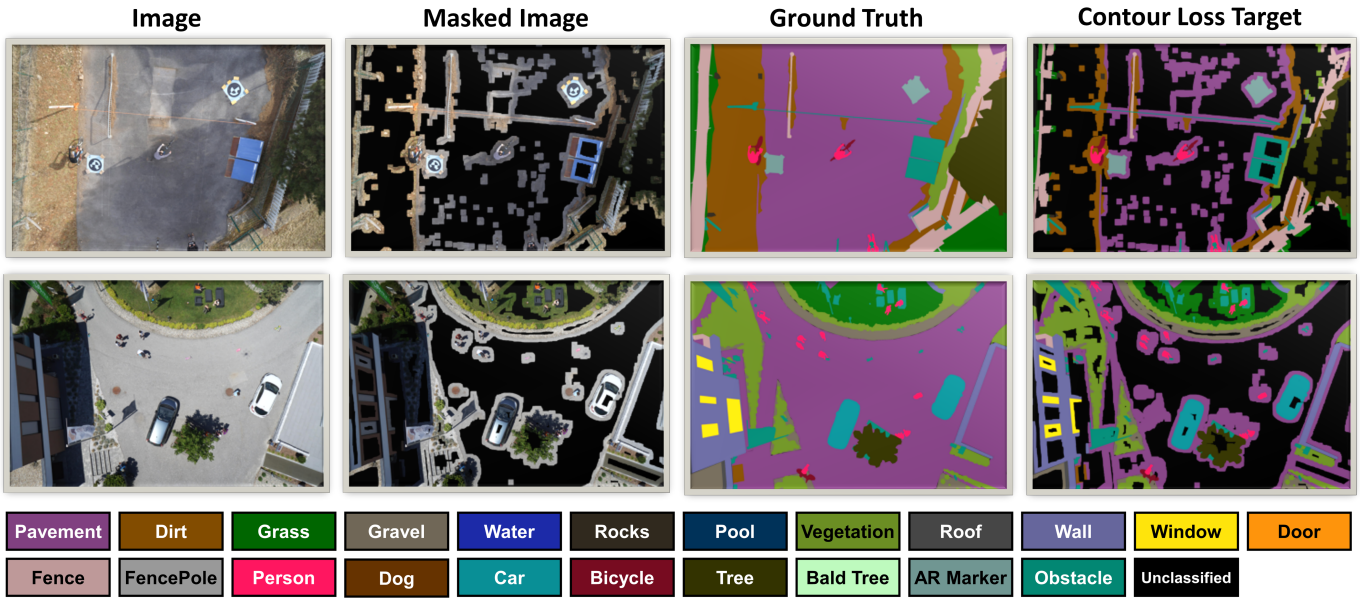

Fig. 1. Images, Masked Images, Ground Truths, and Contour Loss Targets of the images from the Semantic Drone Dataset (SDD) [11].

sparsity to improve efficiency. Additionally, conventional self-attention mechanisms rely on the similarity scores of all query-key pairs for feature aggregation, but not all keys are semantically relevant to a given query. Therefore, removing irrelevant keys and values can reduce the susceptibility of feature aggregation to noise, implying the need for content-based sparsity to enhance representation quality.

When UAVs survey large areas, such as landscapes, cities, or infrastructure, smaller objects—such as humans, vehicles, or poles—occupy only a small portion of the image. As a result, these objects appear significantly smaller compared to larger features like roads, buildings, or fields. This diminutive appearance of key objects presents challenges for semantic segmentation networks in accurately detecting and recognizing them. Fourier analysis of the feature maps shows that multi-head self-attention (MHSA) attenuates high-frequency signals, acting as a low-pass filter [10]. This behavior can exacerbate errors in detecting small objects and boundaries. Therefore, a method to highlight high-frequency components is required to complement MHSA.

To address the three challenges mentioned above, we propose a lightweight semantic segmentation network and a network-agnostic loss specifically designed for UAV imagery. The DST incorporates both data-based sparsity, which reduces computational complexity and increases inference speed, and content-based sparsity, which filters out irrelevant information to produce more refined aggregated features. The novel loss leverages predicted contours to capture complex patterns, boundaries, and small objects, imposing an additional penalty for misclassifications in these areas. This encourages the network to prioritize objects that are difficult to distinguish. As shown in Figure 1, small objects (e.g., persons, bicycles, cars, and obstacles) and boundaries are accurately predicted in our pipeline and subsequently used to generate the targets for the Contour Loss. As presented in Figure 2, our DST

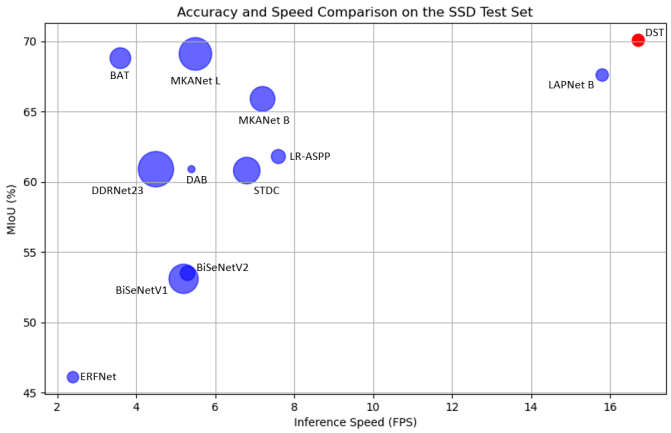

Fig. 2. A comparison of accuracy, speed, and the number of network parameters on the SSD dataset, with the size of the circle indicating the number of network parameters. Inference speeds were evaluated using an image size of 3072×2048 pixels on an NVIDIA RTX 2060 Max-Q 6G Mobile GPU.

(denoted in red) exhibits an exceptional speed-accuracy trade-off, offering key advantages such as low computational cost, minimal memory usage, and low-latency processing, thereby effectively meeting the stringent constraints of UAV platforms. The contributions of this study can be summarized in three aspects:

1) To reduce redundancy and eliminate irrelevant information, we have developed a dual sparsity attention mechanism that performs top-K attention within dilated sliding windows centered on each query token, effectively integrating both data-based and content-based sparsity.
2) To enable real-time image segmentation on UAV platforms with strict hardware constraints, we have created

a lightweight semantic segmentation network that incorporates kernel re-parameterization for acceleration and adaptive weighting for effective multi-scale feature fusion. Our DST attains impressive accuracy in processing 4K resolution images in real-time on a mobile GPU, demonstrating its suitability for onboard deployment.

3) To improve the precision of boundary detection and segmentation of small objects, we propose a novel, network-agnostic loss, termed Contour Loss. This loss focuses on capturing contours and small objects, applying a higher penalty for misclassifications in these regions. Thereby, it encourages the network to prioritize the accurate detection of challenging-to-discriminate objects.

II. Related Work

This section begins with an overview of different efficient transformer attention mechanisms, followed by a discussion of various boundary loss functions.

A. Efficient Transformer Attention Mechanisms

In a ViT, an image is divided into patches that are flattened and embedded into a sequence of tokens. These tokens are then passed through a series of transformer layers where self-attention is applied. The vanilla self-attention mechanism computes the relationships between all pairs of tokens in the image, which scales quadratically with the number of tokens. For high-resolution images, this results in a large computational burden due to the quadratic complexity (i.e., $\mathcal{O}(N^2)$, where N is the number of image patches).

Local Attention aims to reduce the quadratic complexity of self-attention by restricting the attention scope to a smaller, local subset of tokens. For instance, Windowed Attention limits each token to attending only to a fixed window of neighboring patches. A prominent example is the Swin Transformer, which partitions the image into non-overlapping local windows and applies a window-shifting strategy in subsequent layers to facilitate information exchange across windows [12]. Axial Stripe Attention, on the other hand, divides the image encoding into vertical and horizontal stripes, performing self-attention within cross-shaped windows [13]. However, these studies primarily incorporate the inductive bias of locality, as explored in CNNs, to enhance the efficiency of transformer attention but fail to address sparsity and redundancy.

Sparse Attention seeks to reduce computational complexity and memory consumption by introducing a sparsity pattern in the attention matrix. Instead of each token attending to all other tokens, sparse attention limits the connections to only a subset of tokens. For instance, some methods use random sampling for the attention matrix, where each token attends to a random subset of tokens rather than all others [14]. In Strided Attention, each token only attends to tokens that are spaced by a fixed stride or interval [15]. However, these works either employ a dropout strategy that randomly discards attention scores or adopt fixed-pattern sparsity, both of which fail to eliminate noisy interactions between irrelevant tokens.

Both Local Attention and Sparse Attention fall under data-based sparsity, where the attention pattern is either predefined or fixed. In contrast, our Dual Sparsity Attention (DSA) extends beyond locality by incorporating content-based sparsity, adaptively selecting the top-K tokens for each query token. This approach enables the model to focus on the most relevant patches while ignoring less important ones, enhancing both the efficiency and flexibility of the attention process.

B. Boundary Loss Functions

Boundary loss functions are specialized loss functions that are designed to handle tasks where the accurate delineation of object boundaries is crucial. These types of loss functions are often used in image segmentation, especially for tasks like semantic segmentation and instance segmentation, where it is important not only to classify pixels correctly but also to accurately delineate the contours or boundaries of objects.

In typical segmentation tasks, traditional loss functions may fail to capture fine-grained details of object boundaries. Boundary loss functions are therefore introduced to explicitly penalize deviations from the ground truth boundaries. These losses are particularly useful when objects are very thin or have complex shapes, where the boundaries are crucial for accurate segmentation. However, manually annotating boundary/contour labels for a specific dataset is both costly and time-consuming. Therefore, an automatic and accurate boundary/contour label generation method is crucial for boundary loss functions.

In recent years, several studies have proposed various boundary/contour label generation methods. For instance, STDC employs the Laplacian operator on ground-truth images to generate binary detail labels [4]. However, these labels lack class-specific information and suffer from label scarcity due to the absence of dilation operations, resulting in limited effectiveness. MKANet uses the Sobel operator on ground-truth images to create boundary labels [16] , while MGTT applies the Connected Component Algorithm for the same purpose [17]. However, these approaches are most effective in the context of land-cover classification for satellite imagery, where boundaries are often unclear due to low spatial resolution. Consequently, they struggle to capture complex patterns within ground objects and fail to adequately address challenges related to high intra-class variation.

The Laplacian operator is a traditional image processing technique used to detect edges and contours. It is based on the concept of the second derivative of an image, highlighting areas of rapid intensity change. LAPNet applies the Laplacian operator to images to generate contour labels for the boundary loss function [18]. However, the Laplacian operator lacks the ability to capture the broader context of an image, making it less effective at detecting contours that are not sharply defined or where the boundary does not exhibit a clear intensity transition. This limits its ability to identify complex or nuanced contours.

Rather than relying on a single predefined operator, our approach utilizes a transformer network to learn hierarchical feature representations from data, developing multiple layers of filters that capture not only basic edges but also more complex patterns and contours. By incorporating dilated transformer attention and multi-scale feature fusion, the network

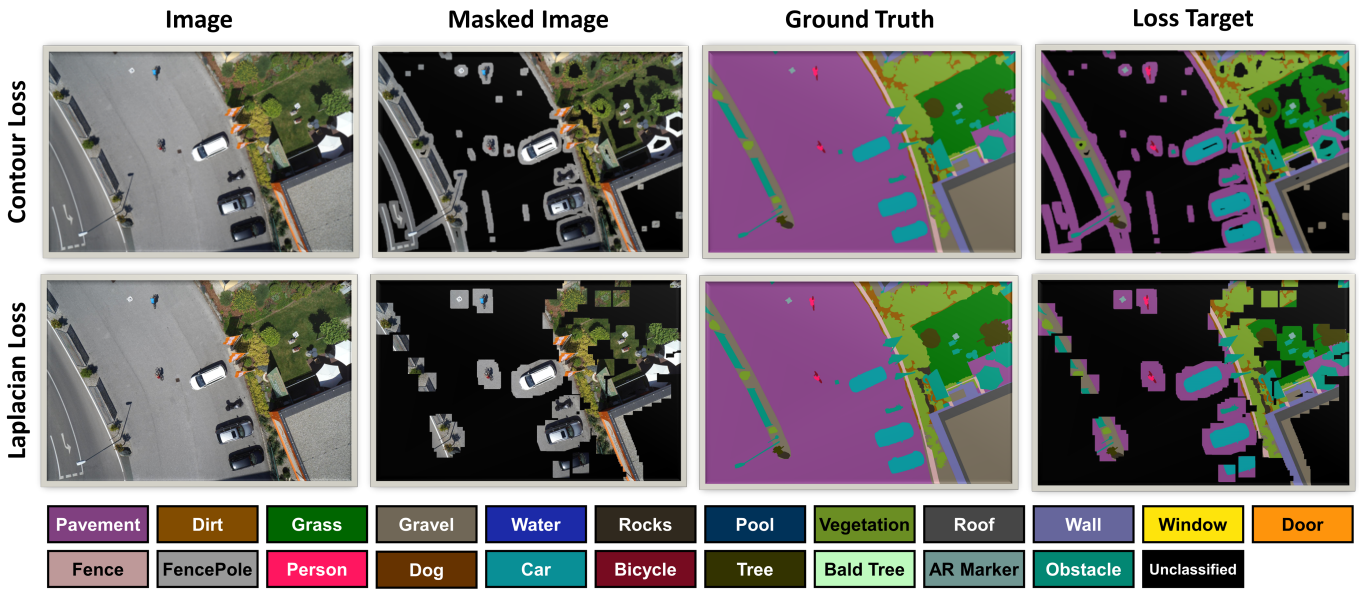

Fig. 3. Comparison of ground truth labels for boundary loss calculation between our approach and the Laplacian operator.

integrates local details with global context, thus generalizing more effectively to new, unseen images. Specifically, we first train the network on a contour dataset of general images and then generate contour labels for UAV datasets through transfer learning.

As seen in Figure 3, our approach, Contour Loss, produces smoother object boundaries and more precise highlighting of small objects, such as vegetation, fences, and poles. In contrast, the Laplacian operator tends to generate more fragmented or imprecise boundaries, particularly for small objects and thin structures. The comparison indicates that Contour Loss yields more accurate ground truth labels for boundary loss calculation, reducing boundary artifacts and preserving object shapes more effectively. Its ability to capture fine details makes it superior to Laplacian Loss, particularly in applications requiring precise object contours.

III. Proposed Method

This section begins by detailing the DSA mechanism, followed by an overview of the network architecture of our lightweight model for onboard semantic segmentation. Finally, we introduce the innovative Contour Loss.

A. Dual Sparsity Attention

Visualizing attention map activations in the standard transformer reveals that patches with high attention scores are sparse and clustered around query patches. This observation motivates us to develop a DSA mechanism based on its three key characteristics: locality, sparsity, and non-redundancy.

To address these characteristics, DSA employs the following strategies:

- For locality, sliding windows centered on each query token are used to constrain the attention scope.
- For sparsity, only the top-K attention scores are utilized for feature aggregation.
- For non-redundancy, the concept of depthwise dilated convolution is incorporated, with varying dilation rates applied across multiple heads to introduce different token spacings in each head.

As illustrated in Figure 4, the N heads divide the tokens along the channel dimension into N slices. For a query token located at (i, j) in the n-th slice ($1 \leq n \leq N$), DSA first selects the corresponding keys and values within a dilated window of size $w \times w$, centered at (i, j), with a dilation rate of n. It then retains only the largest K similarity scores and their associated value tokens for feature aggregation. The output token at position (i, j) in the n-th slice, denoted as y_{nij}, is given by:

$$y_{nij} = \sigma\left(\Uparrow_{\mathcal{K}}\left(\frac{q_{nij} \times \widehat{\mathbf{k}}_n^{\top}}{\sqrt{d}}\right)\right) \times \widehat{\mathbf{v}}_n \tag{1}$$

where $\sigma(\cdot)$ denotes the softmax operator, and $\Uparrow_{\mathcal{K}}(\cdot)$ represents the top-K selection operator, defined as:

$$\Uparrow_{\mathcal{K}}(\mathbf{s})_m = \begin{cases} \mathbf{s}_m & \text{for} \quad \mathbf{s}_m \geq t \\ -\infty & \text{for} \quad \mathbf{s}_m < t \end{cases} \tag{2}$$

here, $\mathbf{s}_m$ is an arbitrary element of the vector $\mathbf{s}$, and t is the K-th largest value in $\mathbf{s}$.

For the n-th slice of the Keys or Values, the notation $\widehat{\mathbf{x}}_n \in \mathbb{R}^{w^2 \times d}$ represents a matrix formed by stacking tokens within a dilated window of size $w \times w$, centered at (i, j), with a dilation rate of n. It is defined as:

$$\widehat{\mathbf{x}}_n = \{x_{ni'j'} | i' = i + p \times n, j' = j + q \times n\}, \tag{3}$$

with $-w/2 \leq p, q \leq w/2$.

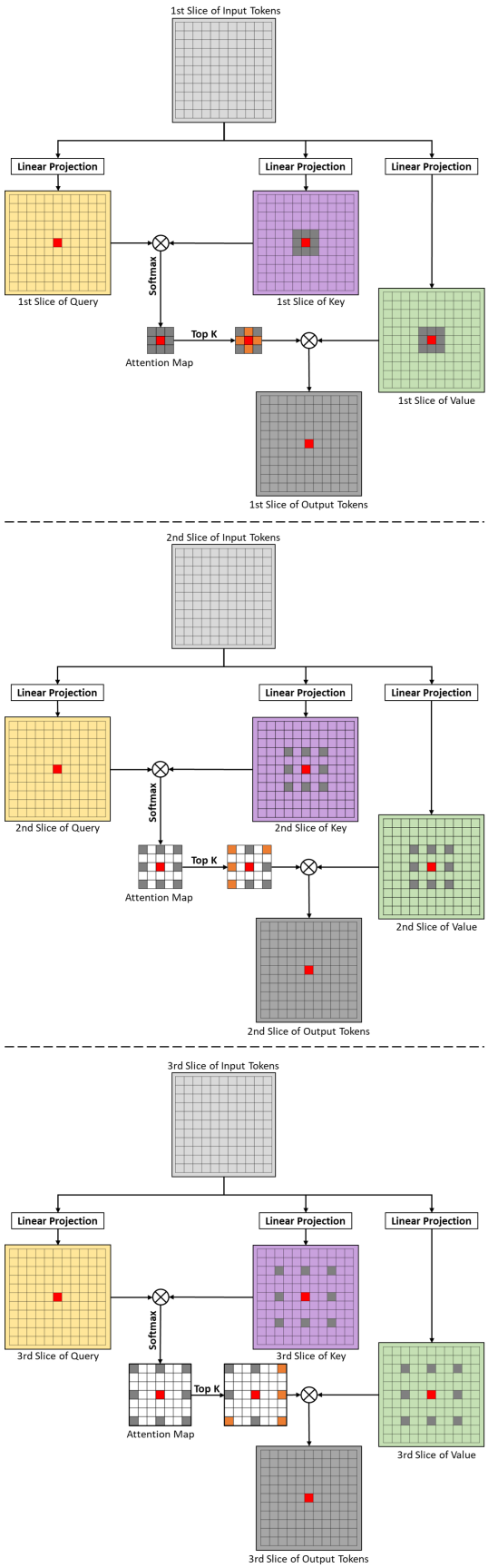

Fig. 4. Structure of the DSA.

Finally, the output tokens of all N attention heads are concatenated and passed through a linear projection to obtain the final output features.

To enable sparse attention, we apply a hard top-K operator that retains only the top scoring keys for each query and masks the rest with $-\infty$ before applying the softmax. This operation is non-differentiable with respect to the top-K indices; however, gradient flow is preserved through the retained attention weights after the softmax. Specifically, the top-K selection acts as a routing mask during backpropagation, and gradients are propagated only through the selected entries. In our experiments, we observed no degradation in convergence or performance due to the non-differentiability of the top-K selection, and the model effectively learns to adapt its attention patterns within this sparse structure.

In DSA, N serves as the hyperparameter that controls data-based sparsity, whereas K regulates content-based sparsity. Larger values of N or smaller values of K increase sparsity, while smaller values of N or larger values of K reduce sparsity. We set $N = 3$ and $K = w^2 \times 0.5$ in subsequent experiments and analyzes the effects of different settings in the ablation study.

B. Network Architecture

As shown in Table I and the top section of Figure 5, the encoder of the DST comprises six stages. Each stage begins with a 3×3 convolution with a stride of 2, followed by batch normalization and ReLU activation. Starting from the third stage, we employ three depthwise convolutions with kernel sizes of 3×3, 7×7, and 11×11, arranged in parallel during training. These kernels are designed to capture features at different spatial scales: small (3×3), medium (7×7), and large (11×11). The use of larger kernels (7×7 and 11×11) enables the network to achieve a comparable receptive field to deeper architectures. During inference, the smaller 3×3 and 7×7 kernels are re-parameterized into the larger 11×11 kernel by fusing batch normalization layers and combining kernel parameters. This transformation reduces the computational cost of the original three parallel branches to that of a single 11×11 kernel.

Let $K_{3\times3}$, $K_{7\times7}$, and $K_{11\times11}$ denote the trained kernels, each zero-padded to match the largest kernel size (11×11). The fused kernel K_{fused} is computed as:

$$K_{\text{fused}} = K_{3\times3} \oplus K_{7\times7} \oplus K_{11\times11}, \tag{4}$$

where $\oplus$ denotes element-wise summation after alignment.

This transformation is exact, as convolution is a linear operator, ensuring no loss of precision in the output feature maps.

Shallow layers primarily capture low-level details (textures, edges), while deeper layers capture high-level semantics (objects, context). Given this distinction, we utilize Stage 3 output features specifically for detail recovery and boundary refinement. Meanwhile, the output features from Stages 4 to 6 are fed into the DSA module to model middle-range dependencies and contextual relationships.

This article has been accepted for publication in IEEE Transactions on Geoscience and Remote Sensing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGRS.2025.3603946

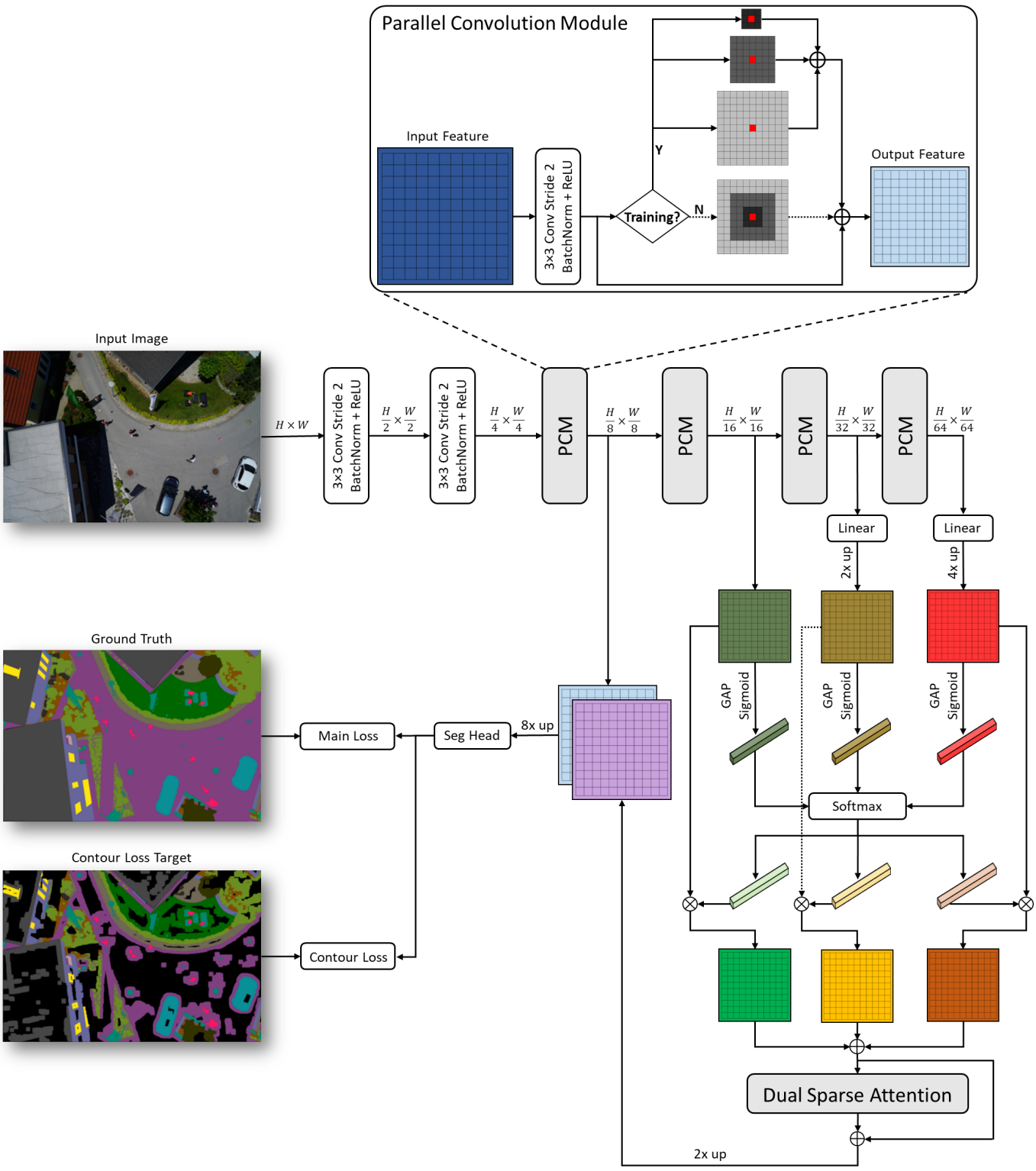

Fig. 5. The Architecture of DST.

Multi-scale features generated by Stages 4 to 6 of the encoder cannot be directly added together due to fundamental differences in their resolution, channel dimension, and the type of information they carry. Directly adding these features without proper consideration can lead to a ”mixing” of apples and oranges, where finer details may be overwhelmed by coarser representations or vice versa. This can result in a loss of important information or create inconsistent and ineffective combined representations. To overcome this challenge, a feature fusion module is implemented to integrate multi-scale features from Stages 4 to 6 before inputting them into the DSA module. This module integrates features from different stages

TABLE I
THE ENCODER OF DST.

Stage	Output Size	Operation	Output Channels
Input Image	1024×1024		3
Stage 1	512×512	ConvS2	$c/2$
Stage 2	256×256	ConvS2	c
Stage 3	128×128	ConvS2	$c \times 2$
	128×128	PCM	$c \times 2$
Stage 4	64×64	ConvS2	$c \times 4$
	64×64	PCM	$c \times 4$
Stage 5	32×32	ConvS2	$c \times 6$
	32×32	PCM	$c \times 6$
Stage 6	16×16	ConvS2	$c \times 8$
	16×16	PCM	$c \times 8$

ConvS2: 3×3 convolution with stride 2, followed by batch normalization and ReLU activation.
PCM: Parallel Convolution Module.
model width (c): the number of channels in the output feature map from the stem stage.

by considering their relative importance and scale. It effectively leverages the strengths of each scale and ensures that the combined features are coherent. The fusion module acts as a mediator, enhancing relevant information while suppressing noise or less important details. As illustrated on the right side of Figure 5, the output features from Stages 5 and 6 are first aligned with those from Stage 4 using linear projection and bilinear interpolation. Next, Global Average Pooling, channel squeezing and restoring via linear projection are applied to the multi-scale features to compute their average channel-wise weights. These weights are then mapped to the [0,1] range using the Sigmoid function, followed by a Softmax operation to normalize them across channels. Finally, the features are scaled by their corresponding normalized weights and summed to produce the fused features.

The features aggregated by the DSA module are interpolated bilinearly with a factor of 2 before being combined with Stage 3 features via concatenation. The merged features are then passed through a 1×1 convolutional layer, which serves as the segmentation head for predicting class logits. Finally, the predicted class logits are interpolated bilinearly with an upscaling factor of 8 to match the original image resolution and are subsequently utilized to compute the primary and contour loss functions.

Two network parameters influence the computational complexity of DST: (1) model width (c, the channel count of the output feature map from the stem stage), and (2) the sliding window size in DSA (w). Our objective is to design a lightweight semantic segmentation network capable of real-time inference at more than 15 FPS on a resource-constrained device with computational capacity below 5 TFLOPS when processing 4K images. Given these constraints, a configuration of $c = 48$ and $w = 3$ meets the required performance criteria.

C. Contour Loss

Empirical observations suggest that prediction errors are more likely to occur along object boundaries and in small regions [19]. Furthermore, Fourier analysis of feature maps reveals that MHSA attenuates high-frequency signals, functioning as a low-pass filter [10]. This effect can further amplify errors in detecting small objects and boundaries. To mitigate these challenges, it is essential to incorporate an auxiliary loss function alongside an automated boundary or contour label generation method. This approach assigns higher penalizing weight to high-frequency components, such as boundaries, small objects, and intricate patterns, during loss calculation, thereby encouraging the model to prioritize the accurate detection of difficult-to-distinguish objects.

Algorithm 1 Generate Contour Loss Target

Input: Image $\mathbf{X}$, Ground Truth $\mathbf{Y}$, Contour Predicting Model $f_\theta(\cdot)$, Dilating Operator $g_d(\cdot)$
Output: Contour Loss Target $\hat{\mathbf{Y}}$

$\mathbf{X_b} \leftarrow f_\theta(\mathbf{X})$
$\mathbf{X_d} \leftarrow g_d(\mathbf{X_b})$
$\hat{\mathbf{Y}} \leftarrow \mathbf{Y} \otimes \mathbf{X_d}$
return $\hat{\mathbf{Y}}$

$\otimes$ represents element-wise multiplication.

Our approach involves training DST on a contour dataset to develop a model capable of predicting contours in UAV images. We note that no existing UAV image dataset contains pre-annotated contour information, therefore, we utilize the Berkeley Segmentation Dataset 500 (BSD500) [20], a well-established contour dataset of general images, for training our network. We anticipate achieving accurate UAV image contours through transfer learning from the BSD500 dataset.

One of our objectives is to delineate strips along boundaries that include surrounding pixels, allowing the network to focus on learning the distinct characteristics of different classes. Another goal is to identify the entire bodies of small objects, such as humans and obstacles. However, since the model is trained exclusively to detect boundaries, the resulting contours are too thin to meet these objectives. To address this, a dilation operation is applied to expand the detected contours. Finally, these refined areas are used to generate targets for the Contour Loss by aligning them with the ground truth annotations.

The procedure is illustrated in Figure 6 and detailed in Algorithm 1. As shown in the top-middle subfigure of Figure 6, the detected contours are initially thin and sparse. To address this, a dilation operation is applied to expand these contours into broader regions. The dilation rate, d, is a hyperparameter that determines the extent to which surrounding pixels contribute to the Contour Loss calculation. For optimal performance, d should be large enough to encompass small objects and their immediate surroundings while excluding distant, irrelevant areas. While fine-tuning this parameter often leads to improved performance, we opted not to fine-tune it during our experiments in order to ensure broader applicability and generalizability. Instead, we empirically set d to 50 pixels for the figures and subsequent experiments to evaluate the effect of the Contour Loss on prediction accuracy. As depicted in Figure 7, this approach effectively highlights boundaries, small objects (such as persons, gravel, and obstacles), and their immediate surroundings.

As illustrated on the left side of Figure 5, the upsampled

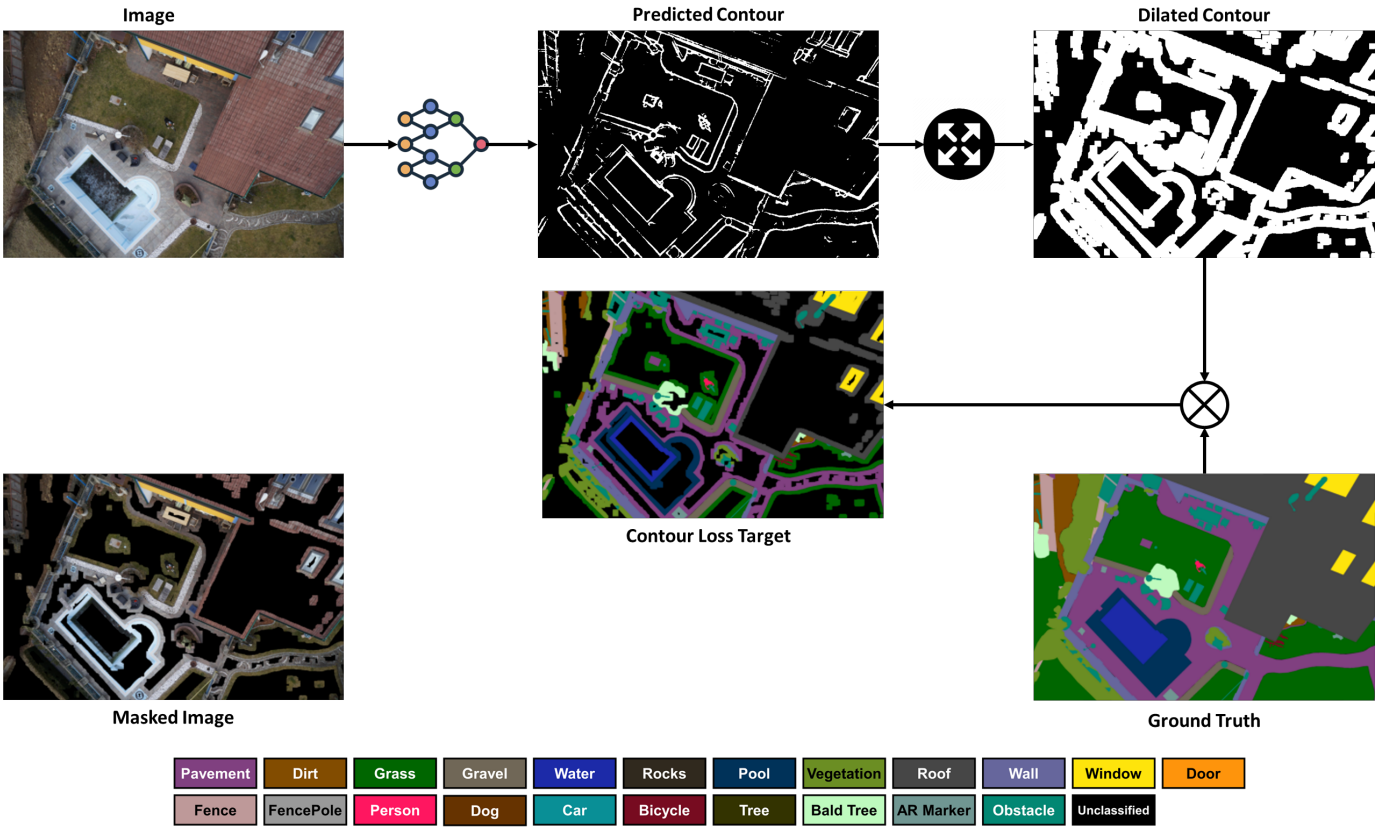

Fig. 6. The procedure for generating the Contour Loss target is described, where $\otimes$ represents element-wise multiplication.

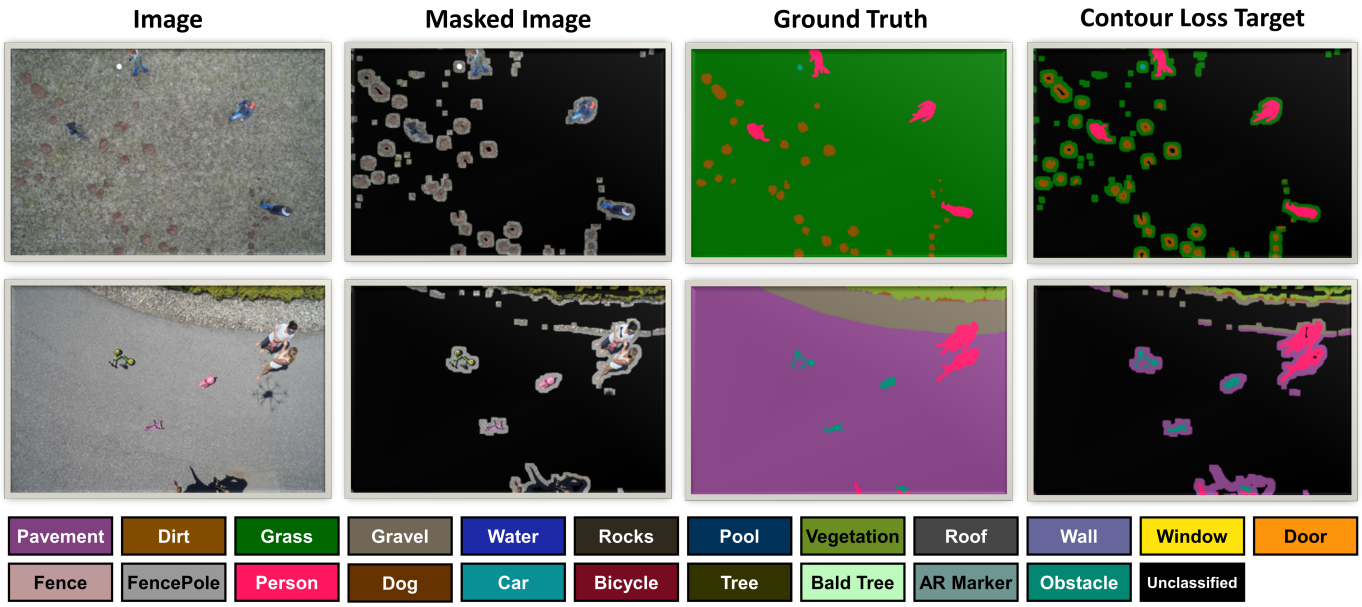

Fig. 7. Images, Masked Images, Ground Truths, and Contour Loss Targets from the SDD dataset.

class logits and Contour Loss targets are fed into the loss function to compute auxiliary loss. By emphasizing intricate patterns, boundary regions, and small objects, the Contour Loss helps the network better distinguish these elements from their surroundings. As a network-agnostic loss function, the Contour Loss can be applied to any semantic segmentation model, with common loss functions such as Cross Entropy or Dice Loss used for its computation.

The total loss, L_t, is computed as a weighted sum of the primary loss, L_p, and the auxiliary Contour Loss, L_a:

$$L_t = w_1 \times L_p + w_2 \times L_a \tag{5}$$

In the following experiments, we used Cross Entropy as the loss function for both the primary and auxiliary losses. Therefore, the total loss, L_t, is given by:

$$L_t = -w_1 \times \sum_{i=1}^{N}\sum_{j=1}^{n} y_{ij} \cdot log(p_{ij}) - w_2 \times \sum_{i=1}^{N'}\sum_{j=1}^{n} y_{ij} \cdot log(p_{ij}) \tag{6}$$

Here, N represents the total number of pixels in a batch, while N' denotes the subset of pixels considered for the Contour Loss—specifically, those with a value of 1 in the Dilated Contour map, as shown in the upper-right corner of Figure 6. n indicates the number of classes, y_{ij} corresponds to the true class probability distribution (typically one-hot encoded), and p_{ij} represents the predicted class probability distribution.

There is a potential risk of overfitting to boundary detection at the expense of overall object recognition accuracy. To mitigate this, we set $w_1 = 1$ and $w_2 = 2$ in our experiments, applying a moderate threefold penalty to misclassifications of critical targets identified by our method.

We acknowledge that there are domain differences between BSD500 (comprising natural and object-centric scenes) and UAV imagery (characterized by top-down views, varying scales). However, our rationale for using BSD500 for pre-training stems from the following observations and design considerations:

- Contour Features are Domain-Independent:
  Contour detection is a low-level task that relies on edge, texture, and gradient features rather than high-level semantics. These features are domain-agnostic and transferable across natural and aerial images. While global image semantics differ between domains, the low-level features used to detect object boundaries (e.g., edges, contours, junctions) are often shared across domains. BSD500 provides high-quality human-annotated contours that enable the model to learn strong edge-detection priors, which are foundational for downstream boundary detection tasks, regardless of image domain.
- Absence of UAV-Specific Contour Annotations:
  Currently, there is no publicly available UAV dataset with explicit boundary or contour annotations. BSD500 thus serves as the most suitable alternative for learning generic contour representations in a supervised manner. Our use of BSD500 is not for final segmentation prediction but as a means to pretrain a contour predictor that contributes auxiliary supervision to improve segmentation accuracy on UAV images.
- Domain Gap Mitigation via Dilation Operation:
  To account for the domain gap, we apply a dilated version of the contours predicted by the pretrained model. This operation helps adapt the contour representation to UAV-specific spatial structures and scales. The dilation operation also broadens the contour regions to better align with the characteristics of UAV-target objects, which are often small and densely clustered.

IV. Experiments

To validate the effectiveness of DST and Contour Loss, we first trained DST on a contour dataset to develop a model for contour prediction. We then conducted experiments on two UAV datasets, comparing our approach against multiple existing methods.

A. Experimental Setting

Network training was performed on an NVIDIA RTX 3090 24GB GPU, while inference speed was measured on an NVIDIA RTX 2060 Max-Q 6GB Mobile GPU (FP32: 4.55 TFLOPS), which closely matches the computational capacity of an NVIDIA Jetson AGX Orin embedded system (FP32: 5.33 TFLOPS). Network performance was evaluated using the Mean Intersection over Union (MIoU) metric, a standard evaluation measure for segmentation tasks defined as follows:

$$\text{MIoU} = \frac{1}{N}\sum_{c=1}^{N}\frac{TP_c}{TP_c + FP_c + FN_c}. \tag{7}$$

where N denotes the number of classes and TP_c, FP_c and FN_c represent the counts of true positive, false positive, and false negative pixels, respectively, for Class c.

For the binary-class contour detection task, the Binary Cross Entropy loss function was used as the sole objective for training, while for the multi-class semantic segmentation task, the Cross Entropy loss function was applied to both the primary loss and the Contour Loss to ensure comprehensive optimization. In both tasks, the AdamW optimizer [21] was employed with a base learning rate of 0.001 and cosine decay scheduling. The network for contour detection was trained for 2,000 epochs with a batch size of 32, incorporating a warmup strategy during the first 200 epochs to stabilize the learning process, whereas the networks for semantic segmentation were trained for 1,200 epochs with a batch size of 12, with warmup applied during the first 80 epochs. To enhance model generalization, data augmentation techniques were implemented during training. For contour detection, these included random flipping, rotation, scaling (factors: 0.8, 0.9, 1.0, 1.1, and 1.25), cropping to 256×256 pixels, and color jittering, while for semantic segmentation, augmentation involved random flipping, rotation, scaling (factors: 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, and 1.75), cropping to 1280×1280 pixels, and color jittering.

B. Experimental Results on the BSD500 Dataset

BSD500 is commonly used to assess both classical and deep learning-based contour detection methods, offering a standard benchmark for boundary detection performance [20]. It consists of 500 natural images, each annotated with multiple human-provided segmentations, making it a valuable resource for evaluating algorithms that aim to replicate human perception of object boundaries. The dataset is divided into 200 training, 100 validation, and 200 test images, covering diverse natural scenes.

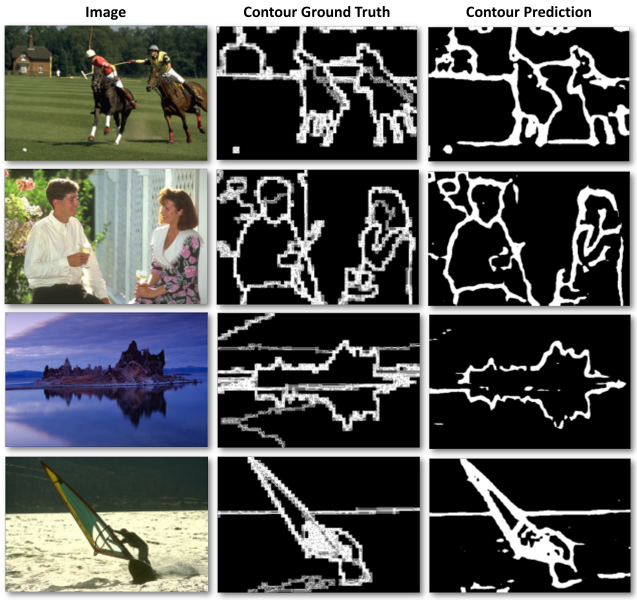

Fig. 8. Predicted contours of images from the BSDS500 dataset.

Figure 8 presents qualitative results of our contour prediction model, demonstrating its ability to accurately capture context-aware contours across diverse scenes. The model effectively preserves salient edges while suppressing noise, ensuring that key structures such as human figures, objects, and natural elements are well-defined. Notably, it performs well in detecting complex and fine details, such as the contours of the polo players, the seated individuals, and the sailboarder, aligning closely with the ground truth annotations. Additionally, the model maintains robustness across varying lighting conditions and backgrounds, successfully identifying edges in both high-contrast and low-contrast regions. Compared to traditional edge detection techniques (e.g., the Laplacian operator applied by [18]), our approach generates smoother, more coherent contours while minimizing fragmentation, providing a solid foundation for the subsequent Contour Loss calculation.

The quantitative results presented in Table II demonstrate the superior performance of DST compared to state-of-the-art edge detectors on the BSD500 benchmark.

C. Experimental Results on the UDD Dataset

The Urban Drone Dataset (UDD) [47] comprises diverse urban scenes from four Chinese cities, captured by UAVs in both oblique and nadir views at altitudes ranging from 60 to 100 meters. It includes six classes: facade, road, vegetation, vehicle, roof, and other. The images have resolutions of either 4096×2160 or 4000×3000 pixels. The dataset is officially divided into training and validation sets, and following previous studies, we report experimental results on the validation set.

As presented in Table III, DST stands out among the methods listed in the table for highest accuracy and fastest inference speed. It achieves the highest MIoU of 74.9%, outperforming all other lightweight models and even matching the best heavyweight model, Segformer. All the Heavyweight models suffer from out-of-memory errors, making them impractical for deployment on resource-limited devices. In the lightweight category, LAPNet Base and MKANet Base come close in accuracy but fall short in speed, with DST running 6.6% faster than LAPNet Base and nearly three times faster than MKANet Base.

TABLE II
COMPARISON OF RESULTS ON THE BSD500 DATASET.

Method	ODS	OIS
DeepEdge [22]	0.753	0.772
DeepContour [23]	0.757	0.776
HED [24]	0.788	0.808
DeepBoundary [25]	0.789	0.811
CEDN [26]	0.788	0.804
RDS [27]	0.792	0.810
AMH-Net [28]	0.798	0.829
RCF [29]	0.811	0.830
CED [30]	0.815	0.833
LPCB [31]	0.815	0.834
EDTER [32]	0.824	0.841
EdgeNAT-S0[33]	0.816	0.834
EdgeNAT-S1[33]	0.824	0.840
EdgeNAT-S2[33]	0.826	0.841
DST	**0.829**	**0.845**

The DST model demonstrates superior segmentation accuracy compared to other networks in the Figure 9. Notably, in the highlighted yellow regions, DST more accurately captures fine-grained details, particularly for facades, vegetation, and road boundaries, closely matching the ground truth. DST maintains sharper object boundaries and clearer distinctions between different classes. This is particularly evident in the top yellow-highlighted structures, DST better identifies the facade and vegetation elements, where other models either misclassify or fail to properly segment the structure. This visual analysis, combined with DST's superior MIoU and FPS performance, solidifies its advantage as a fast yet highly accurate model for real-time semantic segmentation.

D. Experimental Results on the SDD Dataset

The Semantic Drone Dataset (SDD) [11] is designed to enhance the safety of autonomous drone flight and landing procedures through the semantic understanding of urban scenes. It captures a residential community with over 20 houses in a small European town. UAV images were collected from a nadir view at altitudes ranging from 5 to 30 meters, using a high-resolution camera with a resolution of 4000×6000 pixels. The dataset includes 22 classes: paved area, dirt, grass, gravel, water, rocks, pool, vegetation, roof, wall, window, door, fence, fence pole, person, dog, car, bicycle, tree, bald tree, AR marker, and obstacle. The class distribution is shown in Table IV. Unlike other UAV imagery datasets (Chen *et al.*, 2018; Lyu *et al.*, 2020), which contain fewer than 10 classes, this dataset provides a highly detailed classification. Some classes, such as fence poles and bicycles, are small in scale, while others, like persons, dogs, and obstacles, show substantial intra-class variability. Furthermore, the low representation of these classes (approximately 1%) poses additional challenges for semantic segmentation tasks. However, these categories are crucial for ensuring flight safety and enabling autonomous landings. The

TABLE III
COMPARISON OF RESULTS ON THE UDD6 VALIDATION SET.

Method	Class IoU (%) Other	Facade	Road	Vegetation	Vehicle	Roof	MIoU (%)	FPS 2060M
Heavyweight								
U-Net [34]	58.1	67.3	65.2	89.3	62.6	82.5	70.8	OOM
FCN-8s [35]	58.7	67.7	66.1	88.4	66.2	83.5	71.8	OOM
SETR [7]	69.6	70.9	69.8	79.6	61.8	79.9	71.9	OOM
Swin Transformer [12]	70.2	71.8	69.9	82.1	62.3	80.0	72.7	OOM
DeepLabV3+ [36]	70.8	72.3	71.2	81.8	62.9	80.2	73.2	OOM
Sparse Swin [37]	62.4	72.6	71.1	89.7	63.4	80.5	73.3	OOM
OCRNet [38]	71.8	73.4	71.0	81.9	63.6	81.6	73.9	OOM
SegNet [39]	60.7	71.9	68.7	89.0	67.6	86.5	74.1	OOM
ACNet [40]	71.2	**73.2**	71.5	82.8	64.2	81.8	74.1	-
BiFormer-S [8]	61.5	71.8	69.1	90.2	66.9	86.5	74.3	OOM
Dilate-S [9]	61.1	71.7	70.5	89.6	66.7	86.8	74.4	OOM
UAVSNet [41]	60.9	71.7	68.4	89.4	**70.5**	86.9	74.6	-
Segformer [42]	**71.8**	73.1	72.3	82.5	68.6	80.8	**74.9**	OOM
Lightweight								
BiSeNetV1 [1]	57.8	64.4	65.6	90.0	59.0	84.2	70.2	3.9
BiSeNetV2 [2]	59.5	68.0	66.8	89.8	58.7	84.4	71.2	4.1
DDRNet23 [43]	59.6	66.9	66.2	90.4	62.6	85.1	71.8	3.3
LR-ASPP [44]	57.7	68.2	67.1	89.9	63.0	86.3	72.0	5.9
DABNet [45]	59.3	69.4	69.4	89.8	59.3	86.9	72.4	4.1
ERFNet [46]	59.2	72.3	67.4	89.1	65.4	86.7	73.4	1.8
DANet [3]	71.6	72.6	71.3	82.1	63.1	81.3	73.7	5.8
STDC [4]	60.5	68.8	69.2	90.3	66.6	**87.5**	73.8	5.2
MKANet Base [16]	61.9	70.3	69.5	**90.6**	67.1	87.2	74.4	5.7
LAPNet Base [18]	61.8	70.6	70.7	**90.6**	65.6	87.4	74.5	15.1
DST	60.8	72.1	**72.7**	90.3	66.4	86.9	**74.9**	**16.1**

The inference speeds were measured using the original image resolution of 4096×2160 pixels on an NVIDIA RTX 2060 Max-Q 6G Mobile GPU.
OOM refers to an out-of-memory error.
A dash ('-') signifies that FPS was not reported by the authors and could not be measured due to the unavailability of the code on GitHub.

TABLE IV
COMPARISON OF RESULTS ON THE SDD TEST SET.

Method	Class IoU(%) Pave.	Dirt	Gras.	Grav.	Wat.	Rock	Pool	Vege.	Roof	Wall	Win.	Door	Fen.	Pole	Per.	Dog	Car	Bic.	Tree	Bald	Mark.	Obst.	MIoU (%)	Param (M)	Speed (FPS)
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5			
Heavyweight:																									
DeepLabV3 [48]	90.1	49.1	93.1	68.4	90.3	36.8	95.3	66.0	79.3	49.9	34.0	0.0	25.3	0.0	53.7	21.1	88.8	55.8	50.8	53.2	79.0	54.9	56.1	42.0	OOM
DeepLabV3+ [36]	90.5	48.7	92.6	68.9	90.7	37.5	93.6	66.3	81.4	53.0	33.6	3.0	28.8	2.2	52.7	26.3	90.4	55.2	55.0	51.1	77.3	56.7	57.1	54.7	OOM
HRNet48 [49]	89.3	51.2	91.9	68.2	89.9	39.5	92.8	67.3	77.3	53.7	32.3	0.8	29.1	4.4	55.0	46.7	83.3	59.8	64.7	50.1	76.6	57.4	58.2	63.6	OOM
Sparse Swin [37]	94.4	52.7	94.2	81.0	92.7	52.1	97.2	71.1	84.9	60.1	44.0	4.0	44.6	4.2	63.3	44.1	90.2	66.9	70.6	60.4	81.2	63.2	64.4	17.6	OOM
BiFormer-S [8]	92.1	53.6	94.1	72.2	91.7	57.2	97.8	71.1	85.6	62.2	52.8	7.3	48.1	10.6	66.9	55.5	92.9	69.4	70.3	62.6	83.6	65.5	66.5	26.0	OOM
Dilate-S [9]	92.5	53.6	94.0	73.6	91.9	57.8	97.9	71.1	85.8	63.4	53.6	8.8	47.6	8.9	66.8	57.6	93.3	69.9	70.6	62.6	84.5	65.6	66.9	54.0	OOM
Lightweight:																									
ERFNet [46]	91.2	48.3	92.1	77.9	87.3	27.7	90.5	59.2	75.0	28.3	13.0	0.0	2.6	0.0	37.8	0.0	76.0	12.8	52.0	47.3	55.1	39.4	46.1	2.1	2.4
BiSeNetV1 [1]	88.2	48.3	92.7	72.8	87.8	38.9	89.0	65.6	69.3	40.6	23.2	0.0	20.1	0.5	49.1	24.9	71.8	57.0	55.6	47.0	75.8	50.0	53.1	13.8	5.2
BiSeNetV2 [2]	91.6	47.0	91.2	77.9	85.1	37.9	81.2	61.9	77.6	38.0	17.8	0.0	27.6	0.0	48.5	21.2	87.4	57.2	49.3	51.2	75.5	51.5	53.5	3.4	5.3
STDC [4]	93.9	52.0	93.5	81.2	91.4	45.4	94.8	70.6	80.8	50.8	34.6	0.0	37.7	0.7	59.9	37.9	91.1	62.8	67.7	54.5	77.6	59.8	60.8	11.4	6.8
DDRNet23 [43]	91.6	54.4	93.7	72.8	92.5	47.5	95.9	69.7	80.1	55.8	38.2	0.0	37.5	1.9	57.3	30.5	93.1	67.1	63.2	61.5	76.3	59.9	60.9	20.2	4.5
DABNet [45]	93.4	54.6	93.1	78.5	92.1	48.9	92.3	70.7	82.1	50.6	39.9	0.0	38.6	2.6	59.6	29.1	90.4	63.3	69.6	51.1	82.1	58.0	60.9	**0.8**	5.4
LR-ASPP [44]	91.8	52.3	93.9	72.6	90.9	49.3	96.0	70.4	83.1	54.7	38.2	1.3	38.6	2.4	63.7	39.2	92.3	62.5	72.9	56.3	77.6	60.7	61.8	3.2	7.6
MKANet B [16]	93.5	52.9	94.3	78.3	93.7	57.2	97.6	**73.7**	83.1	57.6	52.4	0.6	42.9	7.6	65.7	51.7	93.3	68.8	74.0	64.1	82.5	64.3	65.9	9.9	7.2
LAPNet B [18]	94.9	53.2	94.4	**83.6**	**94.4**	55.8	97.0	71.9	83.6	58.9	49.3	4.6	48.1	8.4	69.2	**65.9**	**94.6**	68.6	72.3	**67.4**	84.7	66.6	67.6	2.5	15.8
BAT [50]	95.3	53.7	**94.8**	83.3	91.4	59.7	97.6	72.7	86.1	63.0	54.2	1.3	52.1	12.2	**72.1**	64.0	93.5	73.7	74.1	65.8	83.2	69.3	68.8	6.9	3.6
MKANet L [16]	94.5	**55.3**	**94.8**	79.5	92.0	**62.0**	97.9	73.5	85.6	67.4	56.1	10.1	48.6	9.8	70.3	59.4	**94.6**	**75.5**	**74.5**	64.7	**86.3**	68.6	69.1	17.1	5.5
DST	**95.4**	53.8	**94.8**	82.6	91.1	60.2	**98.0**	72.6	**87.6**	**68.2**	**61.9**	**20.1**	**54.7**	**13.1**	69.2	58.4	94.2	71.6	**74.5**	66.9	83.9	**69.4**	**70.1**	2.4	**16.7**

Inference speed was evaluated on an NVIDIA RTX 2060 Max-Q 6G Mobile GPU. Experiments utilized images scaled down to 3072×2048 pixels to prevent GPU memory overflow. Predictions from these downscaled images were subsequently upscaled to their original resolution of 6000×4000 pixels.
OOM refers to an out-of-memory error.

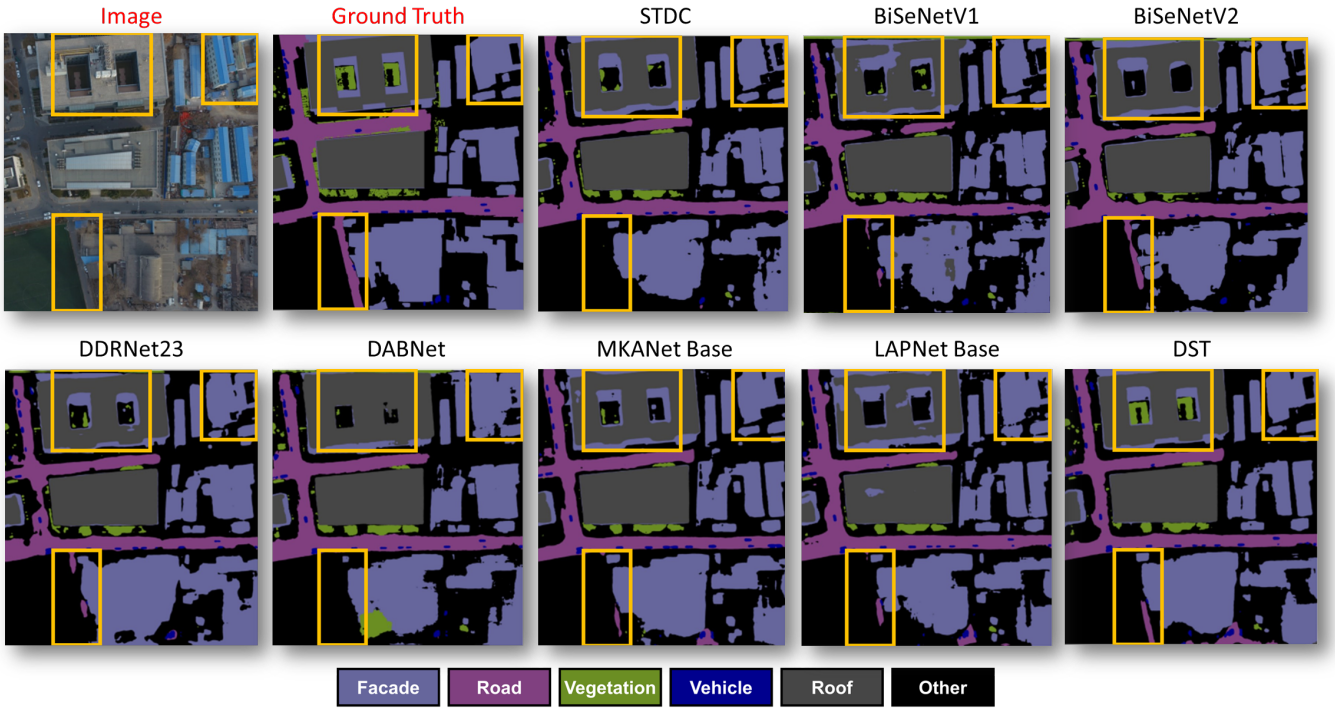

Fig. 9. A comparative analysis of predictions generated by different methods on the UDD dataset.

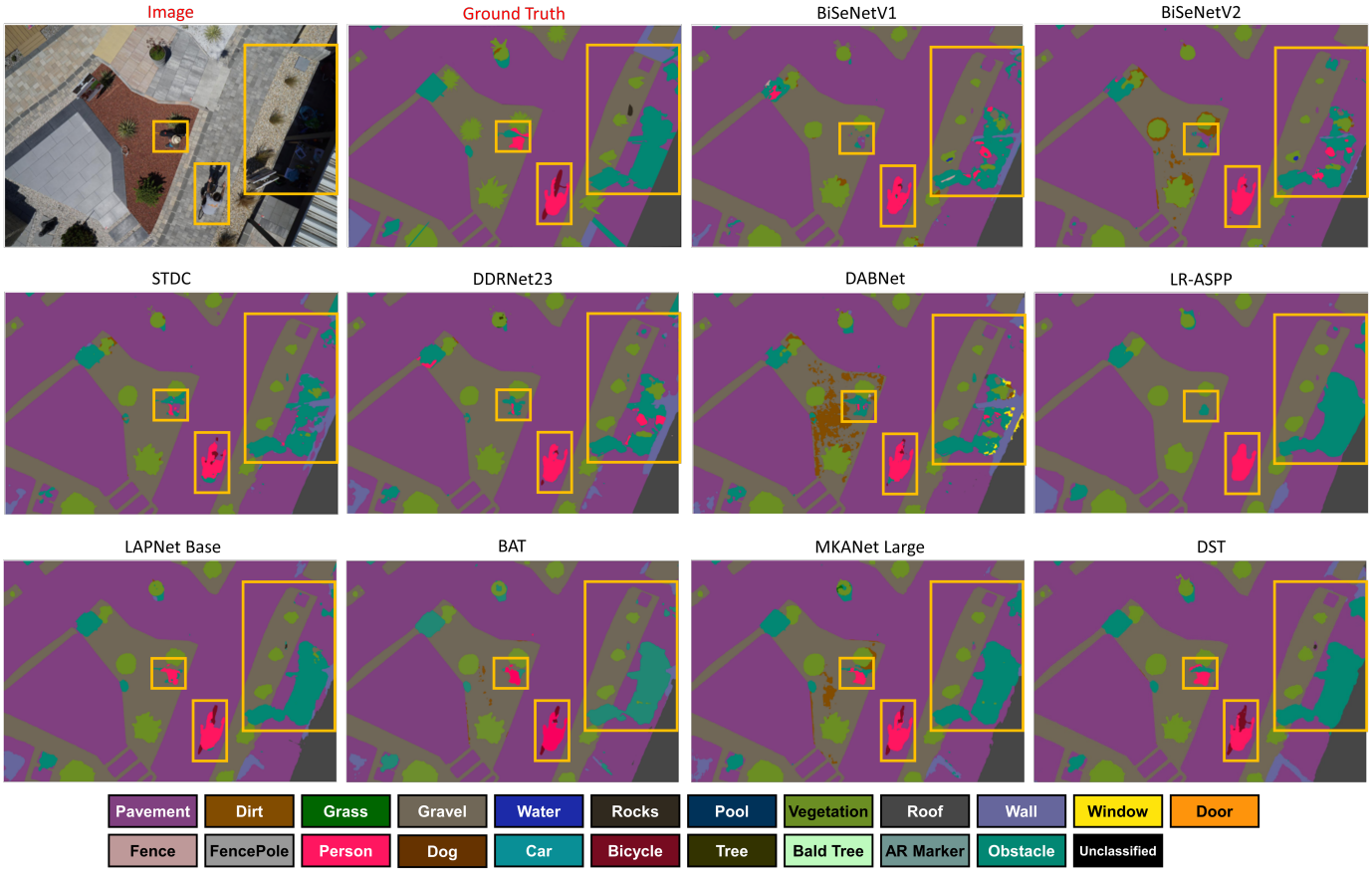

Fig. 10. A comparative analysis of predictions generated by different methods on the SSD dataset.

images and their corresponding ground truths are shown in the first and third columns of Figure 7, with the class legend provided at the bottom. The dataset, consisting of 400 publicly accessible images, is split into training, validation, and test sets in a 7:1:2 ratio.

To prevent GPU memory overflow due to the original image size of 4000×6000 pixels, images were downscaled to 2048×3072 pixels before being fed into the networks. The predicted class logits were then upscaled to restore the original resolution.

As shown in Table IV, DST achieves superior segmentation accuracy while maintaining a compact architecture with only 2.4M parameters—a dramatic outperformance compared to MKANet L (17.1M) and BAT (6.9M). This efficiency ensures low memory consumption and efficient deployment on UAVs. Notably, DST excels at segmenting small-scale objects such as fences, poles, and obstacles, which frequently appear in Contour Loss targets, demonstrating the effectiveness of Contour Loss in refining boundary precision and improving clarity for small objects. Furthermore, DST achieves the highest per-class IoU across various terrain types (e.g., pavement, grass, pool, and tree) and all building components (e.g., roof, wall, window, and door). This capability is particularly crucial for applications requiring precise environmental understanding, such as low-altitude navigation and autonomous landing, where accurately detecting these key elements significantly enhances flight safety and collision avoidance.

In addition to its high accuracy, DST stands out by achieving the fastest inference speed at 16.7 FPS—a significant lead over other lightweight models. The combination of exceptional accuracy, minimal computational overhead, and rapid inference makes DST an ideal choice for real-time semantic segmentation tasks in resource-constrained environments.

In the left yellow box of Figure 10, DST provides the most precise segmentation of both the person and the object he is carrying. The person is clearly defined with accurate shape, while other models struggle to distinguish between the person and the object. In the bottom yellow box, DST preserves the fine details of the cyclist's limbs with high fidelity, closely matching the ground truth, whereas most competing models distort the cyclist's form. In the right yellow box, DST maintains a clean and well-structured segmentation, outperforming other models that exhibit noticeable errors along object boundaries (e.g., fragmented or incorrect labels).

V. Ablation Analysis

Unless otherwise stated, all ablation experiments were conducted using the same default settings as the main model configuration described in Section IV. In each ablation study, only the parameter under investigation was varied, while all other hyperparameters remained fixed. This controlled setup ensures that the effect of each individual component can be independently assessed.

A. The Effect of DSA

To evaluate the contribution of the proposed DSA module, we conducted an ablation study by removing DSA from the full DST model while keeping all other components unchanged. As shown in Table V, the model with DSA achieves a mean IoU (mIoU) of 70.1%, outperforming the version without DSA (67.5%) by 2.6 percentage points. The performance gains are especially prominent in challenging categories such as Wall (68.2% vs. 64.4%), Window (61.9% vs. 54.5%), Door (20.1% vs. 6.2%), Fence (54.7% vs. 49.9%), Pole (13.1% vs. 9.1%), Bald Tree (66.9% vs. 63.5%), and Obstacle (69.4% vs. 66.8%), demonstrating that DSA effectively enhances the model's capacity to capture spatially sparse and structurally complex features. These results validate the importance of incorporating both data-based and content-based sparsity attention for improved segmentation accuracy.

B. The Effect of Sliding Window Size and Top-K Value

The ablation study in Table VI analyzes the effect of different sliding window sizes (w) and top-K values (k) on the performance of DST. The results indicate that increasing the sliding window size from $w = 3$ to $w = 5$ generally improves segmentation accuracy, these gains suggest that a larger receptive field allows the model to capture contextual dependencies more effectively.

K controls content-based sparsity, where smaller values increase sparsity and larger values decrease it. Notably, when $k = 1$, the mechanism reduces to standard sliding window attention with no content-based sparsity. The impact of the top-K value is evident, as lower k values ($k = 0.5$) consistently yield better accuracy compared to $k = 0.75$ and $k = 1$ across both window sizes. This finding supports our hypothesis that reducing redundancy and eliminating irrelevant or noisy information can lead to the generation of more refined aggregated features.

C. The Effect of Head Number

To analyze the sensitivity of the dual-sparsity mechanism to the dilation rates, we note that the dilation rates are implicitly governed by the number of attention heads N. As explained in Section III-A and illustrated in Figure 4, each head processes a distinct slice of the feature map using a dilation rate equal to its slice index n, where $1 \leq n \leq N$. For instance, when $N = 2$, dilation rates of 1 and 2 are applied; when $N = 3$, rates of 1, 2, and 3 are used, and so on. Therefore, varying N inherently varies the range of dilation rates applied across the network.

The ablation study in Table VII analyzes the effect of head number (N) on the performance of DST. N serves as the hyperparameter that controls data-based sparsity, where larger values increase sparsity and smaller values decrease it. As head number increases, the performance generally improves up to a certain point, after which gains diminish or even slightly degrade. This trend suggests that while a larger N provides more contextual information for the model, an excessive value may introduce noise or insufficient information, leading to suboptimal results. The highest segmentation accuracy is observed at $N = 3$, indicating an optimal balance between capturing sufficient context and reducing redundancy.

TABLE V
COMPARISON OF DSTS WITH AND WITHOUT THE DSA MODULE ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
w/ DSA	**95.4**	**53.8**	**94.8**	**82.6**	91.1	**60.2**	**98.0**	**72.6**	**87.6**	**68.2**	**61.9**	**20.1**	**54.7**	**13.1**	**69.2**	**58.4**	**94.2**	**71.6**	**74.5**	**66.9**	**83.9**	**69.4**	**70.1**
w/o DSA	93.8	53.4	94.3	77.3	**91.4**	59.5	97.9	71.9	86.1	64.4	54.5	6.2	49.9	9.1	68.2	57.1	93.4	70.4	71.6	63.5	83.2	66.8	67.5

TABLE VI
COMPARISON OF DSTS WITH DIFFERENT SLIDING WINDOW SIZES AND TOP K VALUES ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
w=3																							
k=0.5	**95.4**	**53.8**	**94.8**	**82.6**	**91.1**	**60.2**	**98.0**	**72.6**	87.6	**68.2**	**61.9**	**20.1**	**54.7**	**13.1**	**69.2**	**58.4**	**94.2**	**71.6**	**74.5**	**66.9**	**83.9**	**69.4**	**70.1**
k=0.75	95.3	**53.8**	94.5	82.4	90.7	59.8	97.8	72.5	**87.9**	67.9	61.8	18.4	53.9	12.4	68.7	58.2	93.6	71.3	74.4	66.5	83.6	69.1	69.8
k=1	95.3	53.7	94.4	82.3	90.5	59.9	97.8	72.3	87.8	68.0	61.5	17.9	53.6	12.1	68.5	58.1	93.6	71.4	74.3	66.2	83.5	69.0	69.6
w=5																							
k=0.5	**94.4**	**53.9**	**94.9**	**78.7**	**92.1**	**61.3**	**98.1**	**72.4**	**89.4**	**70.6**	**60.8**	**34.3**	**56.2**	**13.8**	**71.1**	**63.0**	**95.4**	**73.0**	**71.8**	**63.6**	**85.8**	**70.8**	**71.1**
k=0.75	**94.4**	**53.9**	94.7	78.6	92.0	60.7	**98.1**	72.2	89.1	70.3	60.5	33.4	55.8	13.5	70.9	62.3	95.2	72.3	71.1	62.8	85.1	70.2	70.8
k=1	94.3	53.8	94.5	78.5	91.7	59.9	98.0	71.8	88.8	69.6	60.3	32.8	55.2	13.1	70.4	61.2	95.2	71.8	70.4	62.2	84.7	69.8	70.4

K = Size × Size × k

TABLE VII
COMPARISON OF DSTS WITH VARIOUS HEAD NUMBERS ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
$N = 2$	95.3	**54.0**	94.6	**82.7**	90.2	59.5	97.9	72.0	**87.7**	**68.5**	61.8	17.0	52.4	10.6	66.7	**59.0**	92.6	71.0	74.5	66.5	84.0	68.5	69.4
$N = 3$	**95.4**	53.8	**94.8**	82.6	**91.1**	**60.2**	98.0	**72.6**	87.6	68.2	61.9	**20.1**	**54.7**	13.1	**69.2**	58.4	**94.2**	**71.6**	74.5	**66.9**	83.9	69.4	**70.1**
$N = 4$	**95.4**	53.0	94.7	82.6	90.5	**60.2**	**98.2**	72.3	87.1	67.5	**63.0**	17.0	52.0	**13.3**	67.6	55.7	92.7	71.4	**76.1**	66.3	**84.1**	**69.5**	69.6

TABLE VIII
COMPARISON OF DSTS WITH AND WITHOUT THE CONTOUR LOSS ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
DST w/ Contour (DST)	95.4	53.8	**94.8**	82.6	**91.1**	**60.2**	98.0	**72.6**	87.6	**68.2**	61.9	20.1	**54.7**	**13.1**	**69.2**	58.4	**94.2**	**71.6**	74.5	**66.9**	83.9	69.4	**70.1**
DST w/ Contour (EdgeNAT-S2)	**95.5**	**54.3**	94.5	**83.0**	90.9	**60.2**	**98.2**	72.5	87.2	67.3	62.8	19.9	52.9	11.9	67.1	56.6	93.0	**71.6**	74.7	66.8	**84.6**	69.2	69.8
DST w/ Contour (EDTER)	95.4	53.1	94.7	82.6	90.5	60.1	**98.2**	72.4	87.2	67.5	**63.1**	16.9	52.2	12.7	67.5	55.6	92.7	71.3	**76.1**	66.3	84.2	**69.5**	69.5
DST w/o Contour	95.0	53.2	94.6	81.4	90.2	55.1	97.6	71.7	**88.1**	65.1	55.5	**20.8**	49.4	8.5	67.5	**65.2**	94.0	68.5	71.2	63.6	82.1	66.9	68.4

D. The Effect of Contour Loss

The comparison of segmentation accuracy with and without Contour Loss, as presented in Table VIII, demonstrates the effectiveness of Contour Loss in improving object boundary delineation and overall segmentation performance. The mean MIoU increases from 68.4% to 70.1% when Contour Loss is applied, indicating a notable enhancement in segmentation quality. This improvement is particularly evident in small-scale classes, such as rock (60.2% vs. 55.1%), fence (54.7% vs. 49.4%), and pole (13.1% vs. 8.5%). These results highlight Contour Loss's ability to focus on small objects, ensuring better representation of thin and small structures.

Examining individual class IoU values further reveals the impact of Contour Loss in preserving object shapes and reducing misclassification. For instance, objects with complex boundaries, such as tree (74.5% vs. 71.2%), bald tree (66.9% vs. 63.6%), bicycle (71.6% vs. 68.5%), and obstacle (69.4% vs. 66.9%), exhibit significant gains in accuracy when Contour Loss is employed. These improvements suggest that Contour Loss effectively reduces boundary artifacts and segmentation inconsistencies. Additionally, the increase in IoU for wall (68.2% vs. 65.1%) and window (61.9% vs. 55.5%) indicates that Contour Loss enhances the model's ability to distinguish between adjacent objects with similar textures, reducing over-smoothing along edges.

Contour Loss proves to be superior to traditional segmentation approaches by addressing the limitations of standard loss functions in boundary refinement. While conventional methods struggle with fragmented boundaries and small object segmentation, Contour Loss provides a more precise representation, particularly for thin structures and objects with irregular contours.

To address concerns regarding the effectiveness of our BSD500-trained contour network compared to existing

TABLE IX
COMPARISON OF DSTS WITH DIFFERENT WIDTHS ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
$c = 48$	95.4	53.8	94.8	82.6	**91.1**	60.2	98.0	**72.6**	87.6	68.2	61.9	20.1	54.7	13.1	69.2	**58.4**	94.2	71.6	**74.5**	66.9	83.9	69.4	70.1
$c = 96$	**96.0**	**54.6**	**95.1**	**84.4**	90.7	**64.6**	**98.4**	72.2	**88.9**	**71.4**	**64.2**	**35.1**	**55.6**	**17.8**	**73.1**	56.9	**95.0**	**75.4**	72.5	**67.4**	**86.3**	**71.3**	**72.1**

TABLE X
COMPARISON OF DSTS WITH VARIOUS DILATION RATES ON THE SDD DATASET.

	Class IoU(%)																					MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	**(%)**
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
$d = 20$	95.2	53.4	94.6	82.2	90.8	59.7	97.9	72.3	87.3	67.5	60.4	20.7	53.1	12.4	68.6	**61.4**	94.1	71.0	74.7	66.4	83.3	68.5	69.8
$d = 50$	**95.4**	**53.8**	**94.8**	**82.6**	**91.1**	**60.2**	**98.0**	**72.6**	**87.6**	**68.2**	**61.9**	20.1	**54.7**	**13.1**	**69.2**	58.4	**94.2**	**71.6**	74.5	**66.9**	83.9	**69.4**	**70.1**
$d = 80$	95.3	53.4	94.7	82.3	90.7	60.1	97.9	72.4	87.1	67.2	60.5	**22.0**	54.3	13.0	68.7	54.1	94.0	70.9	**74.8**	66.5	**84.7**	68.9	69.7

transformer-based edge detectors, we conducted additional experiments incorporating pretrained EDTER and EdgeNAT-S2 models into our DST framework. The results, presented in Table VIII, show that our method (DST w/ Contour (DST)) achieves the highest mean IoU (70.1%), outperforming both EdgeNAT-S2 (69.8%) and EDTER (69.5%) under the same experimental setting. These improvements are particularly evident in fine-grained and boundary-sensitive classes such as Fence, Pole, and Person, confirming that the proposed pretraining approach contributes to more accurate boundary modeling than existing alternatives. This demonstrates the effectiveness of our contour loss and BSD500-based pretraining in enhancing scene parsing performance.

E. The Effect of Model Width

The ablation study in Table IX examines the effect of increasing the model width (c, the channel count of the output feature map from the stem stage) from 48 to 96 on DST performance. Expanding the model to $c = 96$ improves the MIoU from 70.1% to 72.1%, enhancing accuracy across most classes. These results indicate that while a narrower model $c = 48$ achieves competitive performance, a wider model captures more detailed representations, leading to refined segmentation. However, the trade-off between computational cost and performance gain must be considered, as wider models typically demand greater memory and processing power.

F. The Effect of Dilation Rate

To assess the sensitivity of the proposed Contour Loss to the dilation radius d, we conducted a controlled study by varying $d \in \{20, 50, 80\}$, as UAV image resolutions and object scales vary significantly across scenes. The results, presented in Table X, show that a moderate dilation radius of $d = 50$ achieves the best performance, yielding a mean IoU of 70.1%. Both smaller ($d = 20$) and larger ($d = 80$) values result in marginally reduced performance, suggesting that while the method is somewhat robust, it benefits from an appropriately tuned dilation rate. Based on these findings, we recommend selecting d proportional to the scale of the smallest object in the scene.

G. The Effect of Loss Weighting

To address the concern regarding the fixed weighting between the contour loss and segmentation loss, we conducted an ablation study by varying the loss-balancing coefficient w_2, while keeping $w_1$constant. Table XI presents the per-class IoU and mean IoU (mIoU) results on the SDD dataset under different settings of $w_2 \in \{0, 1, 2, 3, 4\}$, where $w_2 = 0$ corresponds to removing the contour loss entirely. The results demonstrate that performance improves as w_2 increases from 0 to 2, with the best mIoU achieved at $w_2 = 2$. Further increasing w_2 beyond 2 leads to a slight drop in accuracy, likely due to the contour loss overpowering the primary segmentation objective. These findings confirm that a balanced contribution from both losses is essential for optimal training stability and accuracy. The results also justify our choice of weighting in the main experiments.

VI. CONCLUSIONS

In this paper, we presents a novel approach to addressing the challenges of real-time semantic segmentation on UAV platforms. By integrating both data-based and content-based sparsity, DST effectively balances computational efficiency with segmentation accuracy, making it particularly suitable for resource-constrained environments like UAVs. Data-based sparsity via multi-head dilated sliding window attention reduces unnecessary computations, crucial for embedded systems. Content-based sparsity through top-K attention enhances feature relevance by filtering out irrelevant information, leading to improved segmentation quality. Additionally, the introduction of Contour Loss has proven to be instrumental in refining the segmentation of small objects and intricate boundaries. Furthermore, with its ability to process high-resolution images on mobile GPUs at low latency, our approach meets the stringent demands of onboard UAV deployment.

A promising direction for future research on DST is adaptive loss balancing, where dynamic weighting mechanisms automatically adjust penalties based on object size and boundary complexity. This approach could mitigate overfitting to contours while preserving overall object recognition accuracy.

TABLE XI
COMPARISON OF DSTS WITH VARYING WEIGHTING COEFFICIENT w_2 ON THE SDD DATASET.

	Class IoU(%)																						MIoU
Method	**Pave.**	**Dirt**	**Gras.**	**Grav.**	**Wat.**	**Rock**	**Pool**	**Vege.**	**Roof**	**Wall**	**Win.**	**Door**	**Fen.**	**Pole**	**Per.**	**Dog**	**Car**	**Bic.**	**Tree**	**Bald**	**Mark.**	**Obst.**	(%)
Prop. (%)	37.6	3.3	20.0	7.2	2.3	0.7	0.7	6.9	7.5	2.7	0.6	0.1	1.0	0.1	1.1	0.1	0.8	0.2	2.1	1.4	0.2	3.5	
$w_1 = 1, w_2 = 0$	95.0	53.2	94.6	81.4	90.2	55.1	97.6	71.7	**88.1**	65.1	55.5	20.8	49.4	8.5	67.5	**65.2**	94.0	68.5	71.2	63.6	82.1	66.9	68.4
$w_1 = 1, w_2 = 1$	**95.4**	**54.3**	94.5	**82.9**	90.8	60.1	**98.2**	**72.6**	87.0	67.0	**62.5**	20.0	52.0	11.5	67.3	58.3	93.4	**71.6**	**75.1**	66.4	**85.4**	68.9	69.8
$w_1 = 1, w_2 = 2$	**95.4**	53.8	**94.8**	82.6	**91.1**	**60.2**	98.0	**72.6**	87.6	**68.2**	61.9	20.1	**54.7**	**13.1**	**69.2**	58.4	94.2	**71.6**	74.5	**66.9**	83.9	**69.4**	**70.1**
$w_1 = 1, w_2 = 3$	95.2	53.6	94.7	82.3	90.9	59.6	98.0	72.5	87.4	67.5	60.5	20.9	53.3	12.0	68.7	61.1	**94.3**	71.1	74.7	66.6	83.6	68.6	69.9
$w_1 = 1, w_2 = 4$	95.2	54.2	94.4	82.5	90.7	59.9	98.1	72.5	86.8	67.0	61.7	**21.3**	51.2	10.8	66.9	61.8	93.5	71.5	**75.1**	66.0	84.6	68.3	69.7

Another important avenue is the integration of multimodal data, such as LiDAR or thermal imaging, to complement RGB inputs. This enhancement could improve segmentation performance in challenging conditions, including low-light environments and occluded scenes.

AUTHOR CONTRIBUTIONS

Conceptualization, Wen Lu; Data curation, Wen Lu; Formal analysis, Wen Lu; Funding acquisition, Minh Nguyen; Investigation, Wen Lu; Methodology, Wen Lu; Project administration, Minh Nguyen; Resources, Minh Nguyen; Software, Wen Lu; Supervision, Minh Nguyen; Validation, Minh Nguyen; Visualization, Wen Lu; Writing – original draft, Wen Lu; Writing – review & editing, Minh Nguyen. All authors have read and agreed to the published version of the manuscript.

Wen Lu received the B.Eng. degree in Materials Physics from Wuhan University of Technology in 2007, the M.Eng. degree in Computer Science and Technology from Hubei University of Technology in 2023. He is currently working toward the Ph.D. degree in computer and information sciences with the School of Engineering, Computer & Mathematical Sciences, Auckland University of Technology.

His research interests include computer vision, remote sensing, machine learning, and deep learning.

Minh Nguyen received the B.Sc. degree in computer science, and the M.Sc. and Ph.D. degrees in computer vision from The University of Auckland, Auckland, New Zealand, in 2007, 2010, and 2014 respectively.

Since 2017, he has codirected the Centre for Robotics and Vision with Auckland University of Technology (AUT). Currently, he is the Head of the Department of Computer & Information Sciences with AUT, leading a team of 40 faculty members. His research interests include computer vision, AI, virtual/augmented reality, computer–human interaction, knowledge representation, and machine learning.

REFERENCES

[1] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 325–341.

[2] 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," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3051–3068, 2021.

[3] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146–3154.

[4] M. Fan, S. Lai, J. Huang, X. Wei, Z. Chai, J. Luo, and X. Wei, "Rethinking bisenet for real-time semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 9716–9725.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[7] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[8] L. Zhu, X. Wang, Z. Ke, W. Zhang, and R. W. Lau, "Biformer: Vision transformer with bi-level routing attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 10 323–10 333.

[9] J. Jiao, Y.-M. Tang, K.-Y. Lin, Y. Gao, A. J. Ma, Y. Wang, and W.-S. Zheng, "Dilateformer: Multi-scale dilated transformer for visual recognition," *IEEE Transactions on Multimedia*, vol. 25, pp. 8906–8919, 2023.

[10] N. Park and S. Kim, "How do vision transformers work?" in *10th International Conference on Learning Representations, ICLR 2022*, 2022.

[11] G. U. of Technology, "Semantic drone dataset," 2020, accessed 5 July 2024. http://dronedataset.icg.tugraz.at.

[12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[13] X. Dong, J. Bao, D. Chen, W. Zhang, N. Yu, L. Yuan, D. Chen, and B. Guo, "Cswin transformer: A general vision transformer backbone with cross-shaped windows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 124–12 134.

[14] Z. Zeng, Y. Xiong, S. Ravi, S. Acharya, G. M. Fung, and V. Singh, "You only sample (almost) once: Linear cost self-attention via bernoulli sampling," in *International conference on machine learning*. PMLR, 2021, pp. 12 321–12 332.

[15] W. Wang, W. Chen, Q. Qiu, L. Chen, B. Wu, B. Lin, X. He, and W. Liu, "Crossformer++: A versatile vision transformer hinging on cross-scale attention," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[16] Z. Zhang, W. Lu, J. Cao, and G. Xie, "Mkanet: A lightweight network with sobel boundary loss for efficient land-cover classification of satellite remote sensing imagery," *arXiv preprint arXiv:2207.13866*, 2022.

[17] W. Lu and M. Nguyen, "A lightweight transformer with multi-granularity tokens and connected component loss for land cover classification," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[18] W. Lu, Z. Zhang, and M. Nguyen, "A lightweight cnn-transformer network with laplacian loss for low-altitude uav imagery semantic segmentation," *IEEE Transactions on Geoscience and Remote Sensing*, 2024.

[19] Y. Yuan, J. Xie, X. Chen, and J. Wang, "Segfix: Model-agnostic boundary refinement for segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 489–506.

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, July 2001, pp. 416–423.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[22] G. Bertasius, J. Shi, and L. Torresani, "Deepedge: A multi-scale bifurcated deep network for top-down contour detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4380–4389.

[23] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3982–3991.

[24] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1395–1403.

[25] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[26] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 193–202.

[27] Y. Liu and M. S. Lew, "Learning relaxed deep supervision for better edge detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 231–240.

[28] D. Xu, W. Ouyang, X. Alameda-Pineda, E. Ricci, X. Wang, and N. Sebe, "Learning deep structured multi-scale features using attention-gated crfs for contour prediction," *Advances in neural information processing systems*, vol. 30, 2017.

[29] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3000–3009.

[30] Y. Wang, X. Zhao, Y. Li, and K. Huang, "Deep crisp boundaries: From boundaries to higher-level tasks," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1285–1298, 2018.

[31] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 562–578.

[32] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, "Edter: Edge detection with transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1402–1412.

[33] J. Jie, Y. Guo, G. Wu, J. Wu, and B. Hua, "Edgenat: Transformer for efficient edge detection," *arXiv preprint arXiv:2408.10527*, 2024.

[34] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[36] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[37] K. Pinasthika, B. S. P. Laksono, R. B. P. Irsal, S. Shabiyya, and N. Yudistira, "Sparseswin: Swin transformer with sparse transformer block," *Neurocomputing*, vol. 580, p. 127433, 2024.

[38] Y. Yuan, X. Chen, X. Chen, and J. Wang, "Segmentation transformer: Object-contextual representations for semantic segmentation," *arXiv preprint arXiv:1909.11065*, 2019.

[39] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[40] J. Fu, J. Liu, Y. Wang, Y. Li, Y. Bao, J. Tang, and H. Lu, "Adaptive context network for scene parsing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6748–6757.

[41] S. Kumar, A. Kumar, and D.-G. Lee, "Uavsnet: An encoder-decoder architecture based uav image segmentation network," *arXiv preprint arXiv:2302.13084*, 2023.

[42] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 077–12 090, 2021.

[43] Y. Hong, H. Pan, W. Sun, and Y. Jia, "Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes," *arXiv preprint arXiv:2101.06085*, 2021.

[44] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[45] G. Li, I. Yun, J. Kim, and J. Kim, "Dabnet: Depth-wise asymmetric bottleneck for real-time semantic segmentation," *arXiv preprint arXiv:1907.11357*, 2019.

[46] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[47] Y. Chen, Y. Wang, P. Lu, Y. Chen, and G. Wang, "Large-scale structure from motion with semantic constraints of aerial images," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2018, pp. 347–359.

[48] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[49] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5693–5703.

[50] W. Lu, L. Wei, and M. Nguyen, "Bi-temporal attention transformer for building change detection and building damage assessment," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024.