Unsupervised Modular Adaptive Region Growing and RegionMix Classification for Wind Turbine Segmentation –Supplementary Material–

Raül Pérez-Gonzalo[1,3,†], Riccardo Magro[1,2,†], Andreas Espersen[3], Antonio Agudo[1]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
[2]Politecnico di Milano, Milan, Italy
[3]Wind Power LAB, Copenhagen, Denmark
{rpg,ace}@windpowerlab.com, riccardo.magro@mail.polimi.it, aagudo@iri.upc.edu

1. Revisiting Region-Growing Segmentation

Region-growing segmentation operates on the principle of pixel aggregation, where the grouping mechanism is based on homogeneity in a feature space that often includes intensity levels, color metrics, textural patterns, or spatial closeness. The algorithm commences with the selection of *seed* points, which serve as the starting locations for region growth. These seeds may be chosen based on a-priori knowledge of the image content, through automated processes that might involve statistical analyses, heuristic methods or in some cases can even be performed manually.

As the iterative process unfolds, the algorithm examines adjacent pixels or subregions and decides whether to merge them with the seed's growing region. This decision is contingent upon a similarity criterion, which vary substantially across proposed solutions. The criterion must be carefully defined to ensure that the resulting segmented regions are meaningful with respect to the image's content and the desired application.

The generic outline of a region-growing algorithm is shown in Algorithm 1.

Algorithm 1 Generic Region-Growing Algorithm

Step 1: Seed Selection
Choose an initial pixel to serve as the $seed$
if interactive selection **then**
    User selects the seed pixel
else
    Seed pixel is chosen randomly or through some criteria
end if
Step 2: Similarity Criterion for Pixels
Define a criterion to determine similarity between two pixels
Step 3: Initial Region Aggregation
Add neighboring pixels similar to $seed$ to form an initial region
Step 4: Similarity Criterion for Regions
Establish a criterion for similarity between a pixel and a region
Step 5: Iteration
Continue aggregation until no more pixels can be added

2. Dual-Threshold Seeded Region-Growing

The Seed Selection strategy and the Dual-Threshold Region-Growing coalesce to form a segmentation algorithm, denoted by Dual-Threshold Seeded Region-Growing (DTRG) and summarized in Algorithm 2.

3. Understanding Local and Seed Thresholds

This section elucidates the operational mechanics of local threshold (τ^l) and seed threshold (τ^s) within the presented Seeded Region-Growing algorithm, using visual aids to illustrate their impact on segmentation. To facilitate comprehension, a consistent color-coding scheme across all illustrations is employed:

- **Seed Pixel**: Highlighted in red-violet, indicating the starting point for region growth.
- **Current Pixel**: Marked in yellow, representing the pixel currently under evaluation, surrounded by its 8-connected neighbors $\aleph^{(1)}_{h,w}$.
- **Neighbors**: Shown in grey, these pixels are potential candidates for inclusion in the region.
- **Integrated Pixels**: Highlighted in orange, indicating pixels already assimilated into the region.
- **Color-Coded Circles**: Indicate the evaluation outcome based on local and seed thresholds: green for fulfillment

[†]These authors contributed equally.

Algorithm 2 Dual-Threshold Seeded Region-Growing

Input: Image $\mathbf{X}$
Output: Segmented regions $\{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N\}$
Initialize $\mathbf{X_c}$ as an empty set
Generate the set of candidate seed pixels C
for all $\mathbf{c}_{h,w} \in C$ **do**
  Promote $\mathbf{c}_{h,w}$ to seed pixel $\mathbf{s}_{h,w}$ using Seed Promotion criteria
  Initialize queue $Q \leftarrow \{\mathbf{s}_{h,w}\}$
  Initialize a new region $\mathbf{R} \leftarrow \{\mathbf{s}_{h,w}\}$
  while Q is not empty **do**
    Dequeue $\mathbf{x}_{h,w}$ from Q
    for all $\mathbf{x}_{h',w'} \in \aleph_{h,w}^{(1)}$ **do**
      Calculate d_l and d_s for $\mathbf{x}_{h',w'}$
      if $d_l \leq \tau^l$ and $d_s \leq \tau^s$ **then**
        Add $\mathbf{x}_{h',w'}$ to R and enqueue $\mathbf{x}_{h',w'}$ to Q
      end if
    end for
  end while
  $\mathbf{X_c} \leftarrow \mathbf{X_c} \cup \mathbf{R}$
  Mark $\mathbf{R}$ as a completed region
end for
return segmented regions $\{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N\}$

of both criteria, red for non-fulfillment, blue for only seed threshold satisfaction, and purple for only local threshold satisfaction.

The following illustrations demonstrate how τ^l and τ^s activate, how their combined effect guides the region-growing process, and how the algorithm discerns which pixels to include in a region based on these criteria.

In Fig. 1a, three neighboring pixels fail to be incorporated into the current region due to significant color differences with the seed and current pixels, thus violating both conditions ($d_s \leq \tau^s$, $d_l \leq \tau^l$). Conversely, two neighbors, congruent in color with the blade's RGB values, satisfy both thresholds and are assimilated into the region.

Figure 1b depicts a situation where three neighbors meet the criteria for inclusion, exhibiting color similarities with both the seed and current pixels, thus fulfilling conditions ($d_s \leq \tau^s$, $d_l \leq \tau^l$). The three excluded neighbors align with the seed pixel in terms of color; however, they diverge substantially from the current pixel, breaching the local threshold. Here, τ^l adeptly delineates a boundary in an image with subtle transitions.

In Fig. 1c, two neighbors are integrated into the region, whereas three are rejected due to their significant color variance from the seed pixel. This case exemplifies the seed threshold's utility in addressing the limitations of the local threshold, which may not detect gradual but significant color transitions, by providing a global reference for color consistency.

Algorithm 3 Adaptive Threshold for Seeded Region-Growing Segmentation

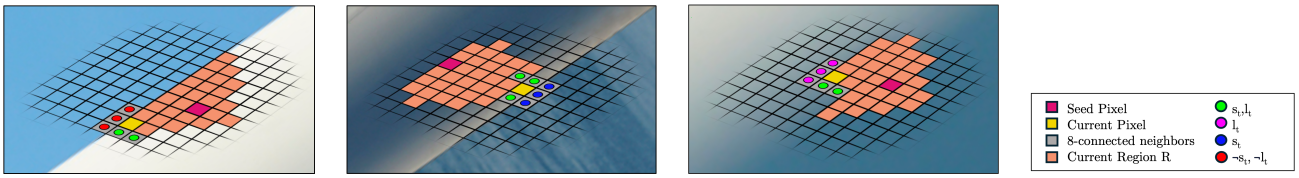
Input: Image $\mathbf{X}$
Output: Adaptive τ^s, τ^l
Initialize τ^s to a preliminary value.
repeat
  Increment τ^s.
  Compute Π.
until There is no further increase in Π.
Establish $\tau^{s*} = \tau^s$.
repeat
  Increment τ^l.
  Compute Π.
until There is no further increase in Π.
Update τ^l to its last value: $\tau^l = \tau^{l*}$
return τ^{s*}, τ^{l*}

In conclusion it is possible to assess that τ^l primarily functions to maintain local color consistency during the region-growing process. For any pixel at the boundary of a growing region in fact, τ^l assesses the color difference between this pixel and its neighborhood. By imposing the condition $d_l < \tau^l$, we restrict the addition of neighboring pixels to the region only if their color intensity does not significantly deviate from that of the bordering pixel. This constraint effectively mitigates the erroneous expansion of a region across the edge of an object, a common issue in scenarios where object boundaries are not distinctly marked.

However, relying solely on τ^l is insufficient due to the presence of gradual color gradients within an image. Such gradients can lead to over-expansion of a region, as the local threshold criterion may continually be satisfied for adjacent pixels despite a considerable overall deviation from the original region color. Here it is where τ^s becomes crucial. It compares the color of a candidate pixel not just with its neighborhood but with the original seed one of the region. By enforcing the condition $d_s < \tau^s$, it is ensured that the pixels added to a region are not only locally consistent but also globally representative of the seed pixel's characteristics. In other words, τ^s solves cases where iteratively each pixel added the deviation from the seed pixel becomes higher and higher without any limits.

4. Adaptive Thresholding

Adaptive Thresholding (AT) algorithm seeks the pair of thresholds based on local image characteristics, fundamental to address a highly diversified dataset. This unsupervised algorithm that adaptively determines τ^s and τ^l based on a dual-thresholding criterion is presented in Algorithm 3. A qualitative example of AT algorithm is illustrated in Fig. 2.



(a) **Simultaneous activation of τ^l and τ^s.** This scenario shows a seed pixel (red-violet) and the current pixel (yellow), surrounded by neighbors (grey) not yet included in the region. Neighbors satisfying both τ^l and τ^s appear with a green circle and red circle mark those failing the criteria.

(b) **Local threshold (τ^l) dynamics.** This scenario shows a seed pixel (red-violet) and the current pixel (yellow), surrounded by neighbors (grey) not yet included in the region. Neighbors satisfying both τ^l and τ^s appear with a green circle. Those failing to satisfy the τ^l appear with a blue circle.

(c) **Seed threshold (τ^s) dynamics.** This scenario shows a seed pixel (red-violet) and the current pixel (yellow), surrounded by neighbors (grey) not yet included in the region. Neighbors satisfying both τ^l and τ^s appear with a green circle. Those failing to satisfy τ^s appear with a purple circle.

(d) **Pixel color legend**.

Figure 1. **Representative illustrations to understand the local τ^l and seed τ^s thresholds** in dual-threshold region-growing algorithm.

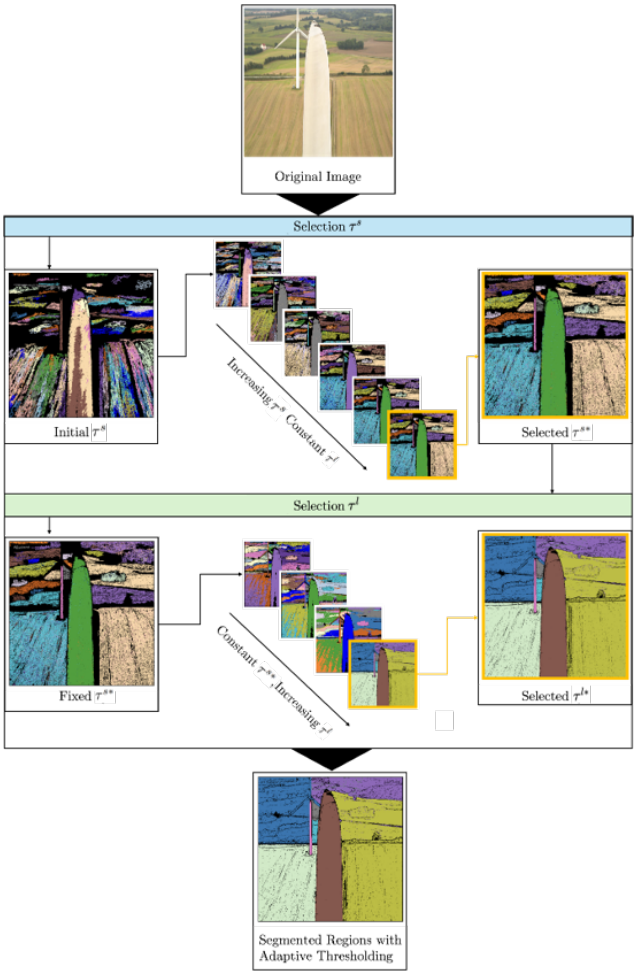


Figure 2. **Visualization of the Adaptive Algorithm:** The process starts from an image and returns the adaptive τ^{s*} and τ^{l*} using Π as an unsupervised metric for gauging segmentation quality.

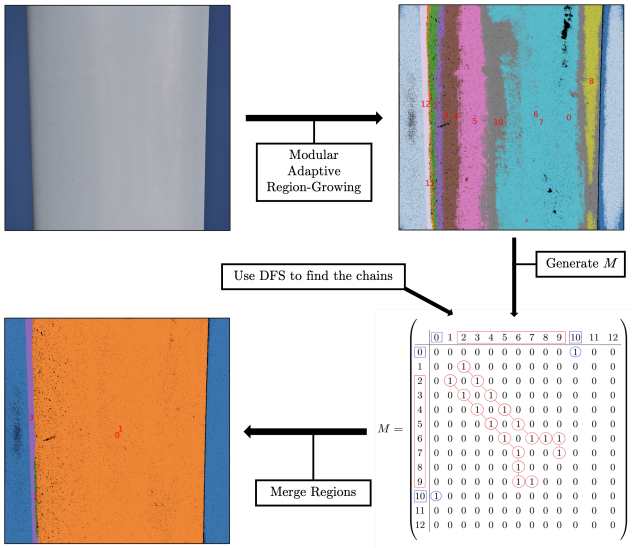


Figure 3. **Comprehensive Overview of the Region Merging Strategy. Top-right:** Initial segmented regions obtained using DTMRG. **Bottom-right:** The merging matrix M, where non-zero entries suggest potential merging of corresponding regions. **Bottom-left:** Final image after the Region-Merging algorithm is applied, illustrating the aggregation of initial segments into larger homogeneous areas. In this figure, the segmented regions are numbered (in red) to highlight the reduction of regions from before to after merging process.

5. Region Merging

The Region Merging process effectively consolidates overlapping regions into cohesive segments, greatly reducing the overall number of regions (see Fig. 3). This approach not only streamlines the classification task but also produces segmentation results that more accurately reflect the image's actual structure, minimizing fragmentation and improving interpretability.

Algorithm 4 Modular Adaptive Seeded Region-Growing Algorithm (MARG)

Input: Image $\mathbf{X}$
Output: Segmented regions $\{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N\}$
Step 1: Adaptive Thresholding
Determine τ^{s*} and τ^{l*}
Step 2: Dual-Threshold Modular Region-Growing
Use τ^{s*} and τ^{l*}, perform DTMRG on $\mathbf{X}$
Step 3: Region Merging

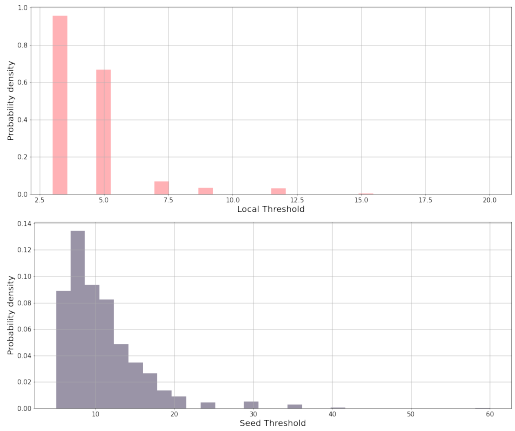


Figure 4. **Optimal local τ^{l*} and seed τ^{s*} threshold distributions over the test dataset.**

5.1. Full Region-Growing Algorithm Outline

All the previous ideas are now combined to propose the comprehensive Modular Adaptive Region-Growing (MARG) algorithm summarized in Algorithm 4. This method can directly split the image $\mathbf{x}$ into different regions $\{\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_N\}$ in an unsupervised fashion, without assuming any training data at all.

6. Hyperparameter Sensitivity

In Fig. 4, we depict the distributions in optimal local and seed thresholds for AT. The high variability in optimal thresholds demonstrates the method's flexibility in adapting to diverse image characteristics, improving region segmentation in challenging conditions.

Moreover, we analyzed the sensitivity of the MARG algorithm to its hyperparameters by assessing how variations in these parameters affect segmentation performance across the test dataset. Specifically, we examined the range within which a hyperparameter could be adjusted without reducing the mIoU and F1-score by more than 1%. The window size k defined in $\aleph_{h,w}^{(k=2)}$ from Seed Selection strategy (Section 3.2 of the main manuscript) can be flexibly modified to $k \in \{2, \ldots, 6\}$. Similarly, the merging overlap from Equation 5 of the main manuscript achieves consistent results within the range $[0.08, 0.32]$.

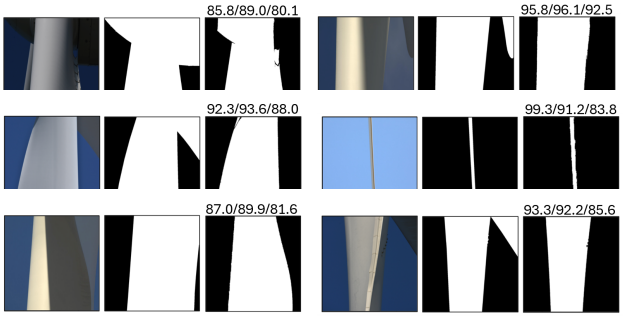


Figure 5. **Typical failure cases on test images of MARG+RegionMix classifier.** The first column presents the input color image, the second displays the ground-truth mask, and the third shows the segmentation estimation. On top of each estimation, the accuracy, F1 and mIoU performance metrics are showcased, respectively. Both sides display the same information.

Failure mode	Percentage of outliers [%]	Accuracy [%]	F1 [%]	mIoU [%]
Blade root	43.8	87.89	85.46	82.86
Multiple blades	42.1	91.45	88.72	85.13
Slender blades	7.5	98.76	92.95	86.32
Atypical lighting	6.6	82.56	82.34	72.64

Table 1. **Distribution of failure modes among outlier cases**. These outliers are taken from the boxplot, refer to Fig. 10 of the main manuscript.

7. Failure Analysis

In Fig. 5, we illustrate some representative instances where our MARG+RegionMix classifier may struggle. This analysis aims to facilitate future research to build on top of our algorithm and improve wind turbine blade segmentation. When analyzing the typical failure cases of our MARG classifier, we observe that the algorithm typically struggles with similar cases as other image segmentation algorithms - despite adopting a total different approach with region classification. These failure cases can be categorized into four main groups: images of the blade's root, images with the presence of multiple blades, slender blade images, and those with atypical lighting conditions.

To provide an in-depth analysis, we studied the outliers present in the windfarm boxplots from Fig. 10 of the main manuscript to analyze robustness. As seen in Tab. 1, around 43% of the outliers depict the blade's root. The model struggles to correctly segment the hub (first row, left side instance), often misclassifying it as background due to the lack of hub images in the training data. These cases correspond to a reduction in mIoU from the overall average to 82.86%. Another 43% of outliers feature multiple blades within the frame (first row, right side; second row, left side instances), leading the model to omit additional blades and classify them as background, as the training set primarily

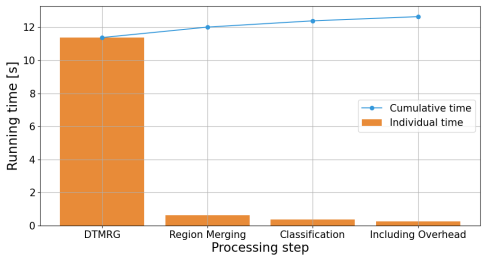


Figure 6. **Computational time of a single image for each step in MARG + Classifier segmentation algorithm**. Computational time for MARG, Region Merging and classification steps are reported. The blue line represents the cumulative time.

Method	GPU Inference Time [s] ↓	CPU Inference Time [s] ↓
BiRefNet [15]	4.93	13.56
SAM [5]	6.40	69.42
CLIPSeg [8]	2.39	3.57
DiffSeg [13]	26.07	79.21
EfficientFormer [6]	4.51	7.32
MobileViT [9]	6.18	13.19
BU-Net [10]	4.89	10.42
Mask2Former-FreqFusion [1]	5.11	42.47
MARG+RegionMix	-	12.23

Table 2. **Computational time comparison on wind turbine blade segmentation** of the distinct state-of-the-art methods.

includes single-blade images. This failure mode results in a mIoU of 85.13%. A smaller fraction (7.5%) of outliers correspond to slender blade images (second row, right side instance), where the regions tend to slightly overestimate the area of slender blades, which visually does not seem alarming, but end up dropping significantly the mIoU performance to 86.32%. Lastly, around 6.6% of outliers arise from unusual lighting conditions (third row instances), such as shadows, which can mislead the model into segmenting a single blade into multiple parts. These scenarios show the most severe degradation, with IoU dropping to 72.6%, and occur primarily because such illumination cases are rare in both training and test sets, reducing the classifier's ability to generalize.

These failures are primarily attributable to dataset limitations and the region classifier, rather than the proposed MARG region-growing algorithm itself. In particular, insufficient training samples for hub and multi-blade configurations limit classifier generalization. Collecting more diverse training data (including hub and multi-blade instances), augmenting with varied lighting conditions, and refining boundary-aware post-processing would mitigate most errors. Furthermore, when compared with competitive segmentation models (Fig. 8), we find that these methods exhibit these weaknesses in blade root and shadowed cases, suggesting that such difficulties are intrinsic to the task rather than specific to our approach.

8. Computational Cost

Similar to previous segmentation algorithms, we analyze the computational cost of the inference process using a system equipped with an NVIDIA RTX 3080 Ti GPU and a 20-core Intel Core i9-10900 processor. To accelerate the selection of optimal local thresholds τ^{l*} and seed thresholds τ^{s*} for AT, the coverage Π of each seed threshold τ^{s} and of each local threshold τ^{l} (given a fixed τ^{s*}) are computed in parallel.

Figure 6 shows the average runtime for each step of the algorithm, including the cumulative time. The total runtime for segmenting a single image is 12.23 seconds, with the most time-consuming step being the DTMRG region-growing algorithm, which takes 11.10 seconds. This is primarily due to the computation of the optimal thresholds in the AT process. Despite parallelization, generating distinct region-growing images remains the primary bottleneck in inference speed.

The remaining steps of the algorithm are significantly faster: Region Merging takes 0.61 seconds, while the classification of each region takes just 0.31 seconds. There is also a small overhead of 0.21 seconds, which includes tasks such as image loading and saving, memory transfers between the CPU and GPU, region assembly, and hole filling.

Our algorithm is desined to run on the CPU, except for the region classification, which can be run with GPU acceleration. We include in Tab. 2 the computational time comparison for segmenting a single image for distinct state-of-the-art algorithms. While it does not achieve the highest speed in CPU timing, our framework provides a highly practical solution when GPU hardware is not available, as it accomplishes top-performing segmentation results and reaches the third highest timing on this hardware compared to state-of-the-art segmentation algorithms. In addition to this, it offers advantages in terms of interpretability, adaptability, and, as mentioned, ease of deployment in environments where GPU resources are limited or unavailable.

9. Generalization Across Windfarms

Table 3 reports performance on each windfarm in terms of mean $\pm$ standard deviation, together with 95% confidence intervals estimated using both a Student-t model and a non-parametric bootstrap. This results are also illustrated in the main manuscript using boxplot distributions in Fig. 10. Results show consistently strong generalization across sites, with accuracies above 95% for all windfarms. The mIoU and F1 scores follow the same trend, with only modest variability depending on windfarm-specific conditions. The close agreement between t-based and bootstrap intervals indicates that the reported means are robust and not strongly affected by distributional assumptions. Notably, performance remains high even on windfarms with greater vari-

Windfarm	Accuracy			mIoU			F1		
	Mean ± Std	CI ($t_{student}$)	CI (boot)	Mean ± Std	CI ($t_{student}$)	CI (boot)	Mean ± Std	CI ($t_{student}$)	CI (boot)
WF1	98.19 ± 5.20	95.76–100.63	95.81–99.44	93.44 ± 8.30	89.55–97.33	89.43–96.64	96.41 ± 4.83	94.15–98.67	94.07–98.25
WF2	99.45 ± 0.24	99.34–99.56	99.34–99.55	95.82 ± 4.95	93.51–98.14	93.49–97.75	97.83 ± 2.70	96.57–99.10	96.56–98.89
WF3	97.53 ± 6.37	94.46–100.60	94.23–99.44	94.74 ± 7.78	90.98–98.49	90.75–97.31	97.12 ± 4.71	94.85–99.39	94.72–98.62
WF4	96.28 ± 8.56	91.88–100.68	91.75–99.17	91.43 ± 11.22	85.66–97.20	85.56–95.53	95.12 ± 7.37	91.33–98.91	91.25–97.69
WF5	97.68 ± 6.08	94.84–100.52	94.62–99.42	95.53 ± 6.98	92.26–98.80	92.07–98.03	97.58 ± 4.04	95.69–99.47	95.55–98.99
WF6	99.45 ± 0.32	99.30–99.61	99.31–99.58	97.82 ± 2.51	96.62–99.03	96.59–98.77	98.94 ± 1.31	98.31–99.57	98.28–99.41
WF7	99.25 ± 1.34	98.62–99.87	98.59–99.67	97.41 ± 4.28	95.41–99.42	95.36–98.79	98.64 ± 2.37	97.53–99.75	97.50–99.39
WF8	98.10 ± 3.09	96.66–99.55	96.59–99.10	96.65 ± 4.29	94.64–98.65	94.51–98.11	98.24 ± 2.38	97.13–99.36	97.06–99.04
WF9	98.53 ± 1.87	97.65–99.41	97.70–99.25	96.84 ± 2.56	95.64–98.04	95.74–97.88	98.96 ± 1.07	98.46–99.46	98.47–99.36
WF10	95.43 ± 8.85	91.29–99.57	91.08–98.67	93.26 ± 11.35	87.95–98.57	87.79–97.59	96.12 ± 6.94	92.87–99.37	92.76–98.73

Table 3. **Quantitative generalization results across test windfarms**. We report mean ± standard deviation, together with 95% confidence intervals computed using both Student-t and bootstrap resampling. Boxplot results are shown in Fig. 10 of the main manuscript.

ability (e.g., WF4 and WF10), highlighting the method's resilience to domain shifts. This demonstrates that our approach generalizes reliably to previously unseen windfarms, addressing a key challenge for deployment in real-world scenarios.

10. Region Classifier Architectures

Our MARG framework first employs our proposed region-growing algorithm (MARG) to generate candidate regions, which are then classified into either blade or background using a CNN- or transformer-based binary classifier; see Fig. 1 and Fig. 7 of the main manuscript. In this section, we compare a range of candidate backbones for the region classifier, summarized in Tab. 4.

We experimented with both convolutional models (VGG [11], ResNet [3], EfficientNet [12]) and transformer-based classifiers (ViT [2], Swin [7], DeiT [14]), trained either from scratch or with official pretrained weights. To adapt these architectures to our task, we extend the input layer with an additional fourth channel that encodes the binary region mask, while leaving the remainder of each model unchanged. The results in Tab. 4 show that most architectures achieve high accuracy on the binary classification task (>97%), yet some differences emerge when balancing validation robustness and efficiency.

Overall, transformer-based models such as ViT-B/16 [2] and DeiT-S [14] reach the highest test accuracy (98.33% and 98.29%, respectively) and F1-score (98.03% and 97.88%). However, they require considerably more parameters and training resources, and their validation accuracy fluctuates more strongly, suggesting potential overfitting to the test set if chosen solely on test performance. By contrast, EfficientNet-B4 [12] with pretrained weights provides a strong trade-off: it achieves the best validation accuracy (98.38%) and lowest validation loss (0.1972), while still reaching competitive test metrics (98.25% accuracy, 97.79% F1, 95.33% mIoU). We therefore selecte-Failure Analysisd EfficientNet-B4 as the backbone for our MARG+RegionMix classifier. This choice ensures that model selection is based on validation performance rather than test set tuning, thereby avoiding overfitting, and at the same time yields an efficient architecture in terms of computational cost.

We also explored whether other CNN- and transformer-based classifiers, such as ResNet-50, ResNet-101 [3], ViT-B/16 [4], Swin-S [7], and DeiT-S [14], would substantially improve binary classification. As seen in Tab. 4, although some of these models slightly outperform EfficientNet-B4 [12] on isolated test metrics, they do not consistently improve validation performance. Moreover, transformer-based models incur higher memory and runtime costs without a clear margin of improvement, which opposes the idea of our algorithm to be deployed on CPU. This suggests that, for the binary classification step in MARG, EfficientNet-B4 offers the best balance between accuracy, generalization, and efficiency.

Moreover, we observed that the use of pretrained weights generally improves both convergence stability and validation performance for CNN-based models. For instance, EfficientNet-B4 and EfficientNet-B0 [12] benefit significantly from initialization with pretrained weights, reaching higher validation accuracy and lower loss compared to training from scratch. In contrast, transformer-based models such as ViT-B/16 [4] and Swin-S [7] are less consistent: while ViT [4] improves with pretraining in terms of accuracy, Swin-S [7] shows reduced performance, likely due to domain mismatch between ImageNet pretraining and our region mask–augmented input. Specifically, Swin-S is the only model whose test performance is significantly lower than its validation accuracy, indicating overfitting caused by the combination of a large model capacity and the relatively small size of our dataset. This highlights that while pretraining is advantageous for convolutional backbones, its benefit for transformer-based classifiers is less clear in this task.

11. In-depth Comparative Comparison

A priori limitation of our method is that the improvements over strong state-of-the-art models, while consistent, are numerically modest when averaged over the test set. All competing algorithms already exceed 90% in standard metrics,

Architecture	Pretr.	Test					Validation	
		Acc. [%]	Prec. [%]	Recall [%]	F1 [%]	mIoU [%]	Loss	Acc. [%]
VGG16 [11]	No	98.13	98.23	96.74	97.78	95.28	0.3487	95.25
ResNet34 [3]	No	97.70	97.91	96.46	97.25	94.69	0.2525	97.93
ResNet34 [3]	Yes	98.01	98.49	96.29	97.64	95.10	0.2669	97.79
ResNet50 [3]	No	97.84	97.95	96.62	97.42	94.89	0.2273	96.97
ResNet50 [3]	Yes	98.01	98.10	96.70	97.52	95.05	0.2522	97.97
ResNet101 [3]	No	97.40	96.61	97.32	96.84	94.31	0.2475	97.03
ResNet101 [3]	Yes	97.52	97.74	96.47	97.12	94.52	0.2452	97.03
EfficientNet-B0 [12]	No	97.67	98.22	96.12	97.31	94.66	0.3541	96.79
EfficientNet-B0 [12]	Yes	97.94	98.45	96.15	97.48	94.98	0.3007	97.98
EfficientNet-B4 [12]	No	97.88	97.46	97.14	97.48	94.97	0.2419	95.84
EfficientNet-B4 [12]	Yes	98.25	98.43	96.49	97.79	95.33	**0.1972**	**98.38**
ViT-B/16 [2]	No	97.32	98.28	95.41	96.72	94.06	0.3361	91.69
ViT-B/16 [2]	Yes	**98.33**	**98.67**	96.57	**98.03**	**95.49**	0.3442	95.33
Swin-S [7]	No	92.75	89.20	97.53	90.69	87.42	0.2420	97.32
Swin-S [7]	Yes	88.74	85.06	**97.59**	88.84	83.14	0.3685	97.71
DeiT-S [14]	No	98.29	98.63	96.40	97.88	95.29	0.3055	95.86
DeiT-S [14]	Yes	98.14	98.09	96.94	97.75	95.28	0.2612	96.49

Table 4. **Quantitative comparison of architectures for our region classifier** with and without public pretrained weights, which is denoted in the Table as "Petr.". We highlight the top-performing metric in bold and underline the second and third best models.

and although ours is the top performer (see Table 3 of the main manuscript), it is only the third fastest on a CPU-only environment (Tab. 2). It is therefore important to assess whether this tradeoff is justified, especially since segmentation serves as a precursor for downstream defect analysis.

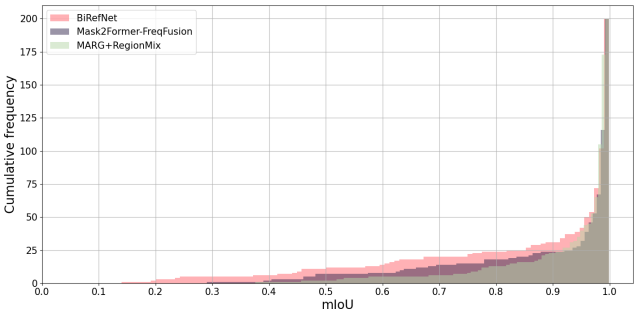
Fig. 7 provides further insight. While average mIoU differences appear small(95.05% for MARG+RegionMix vs. 94.65% for Mask2Former-FreqFusion [1] and 92.38% for BiRefNet [15]), our method markedly reduces poor-quality segmentations. Only 2 of 200 test images fall below an mIoU of 0.5 with MARG+RegionMix, compared to 7 for Mask2Former-FreqFusion and 11 for BiRefNet. This robustness at the lower tail is crucial: downstream defect detection pipelines are highly sensitive to under-segmentation, as missing pixels at blade boundaries can obscure or distort small cracks. Thus, even a 1–2% mean IoU gain translates into substantially fewer failure cases in practice.

Fig. 8 highlights these differences qualitatively. In challenging cases with shadows, cluttered backgrounds, or repairs, baselines often under-segment blades while still reporting high mIoU scores. For example, in the top-left case, Mask2Former-FreqFusion [1] and BiRefNet [15] miss a serrated-edge defect, despite achieving 96.26% and 91.77% mIoU, respectively. Similarly, in the bottom-left case, BiRefNet [15] fails to capture a repair patch, yet still reports 95.18% of mIoU. By contrast, MARG+RegionMix produces more complete masks, preserving defect-relevant regions that would otherwise be lost.

In summary, although the average gains of MARG+RegionMix may appear modest, they correspond to a tangible reduction of problematic segmentations. For downstream applications such as defect detection or repair monitoring, robustness in these difficult cases is more critical than raw throughput. Moreover, our approach remains interpretable and competitive in runtime (top three on CPU), making the tradeoff justifiable for real-world use.

Model	mIoU Threshold							
	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
BiRefNet [15]	2	5	6	11	14	20	24	31
Mask2Former-FreqFusion [1]	0	1	2	7	8	14	18	24
MARG+RegionMix	0	0	1	2	5	6	13	23

(a) mIoU below the given threshold.



(b) Cumulative histogram of mIoU.

Figure 7. **Distribution of mIoU across the test set** (200 images) for BiRefNet [15], Mask2Former-FreqFusion [1], and MARG+RegionMix.

12. Wind Turbine Image Segmentation Data

The dataset used in this work is the same as that introduced in BU-Net [10], and we expand its description here for clarity and completeness. It comprises color images with a resolution of 1024×1024, captured from different sections of the blade. The corresponding masks have been annotated manually to obtain the ground-truth solution. Each image is guaranteed to contain at least one wind turbine blade instance, as all acquisitions were carried out during targeted inspection campaigns. The dataset is therefore curated specifically for the task of wind turbine blade segmentation, ensuring relevance to real inspection scenarios.

Wind turbine blade imagery is inherently difficult to obtain due to safety restrictions, the high cost of inspections, and the sensitive nature of the data. Many images contain proprietary information about the operator's assets, such as turbine design, location, or operational conditions; a significant defect in the data could potentially compromise the reliability or confidentiality of these assets. Unlike generic segmentation benchmarks, this dataset reflects a highly specialized domain where both scarcity and confidentiality are major constraints.

Due to be taken in the wild, the images highly vary in visual appearance, leading to a challenging dataset. See some instances of the wind turbine blade segmentation data in Fig. 8. The blade can diverge in size, shape or illumination conditions. Furthermore, the background is completely distinct depending on the engine used to take the picture. In general, the expected relation between the background and the blade area is 2:1, but it is worth noticing that in many cases the blade area could be very reduced or, conversely, cover almost entirely the image.

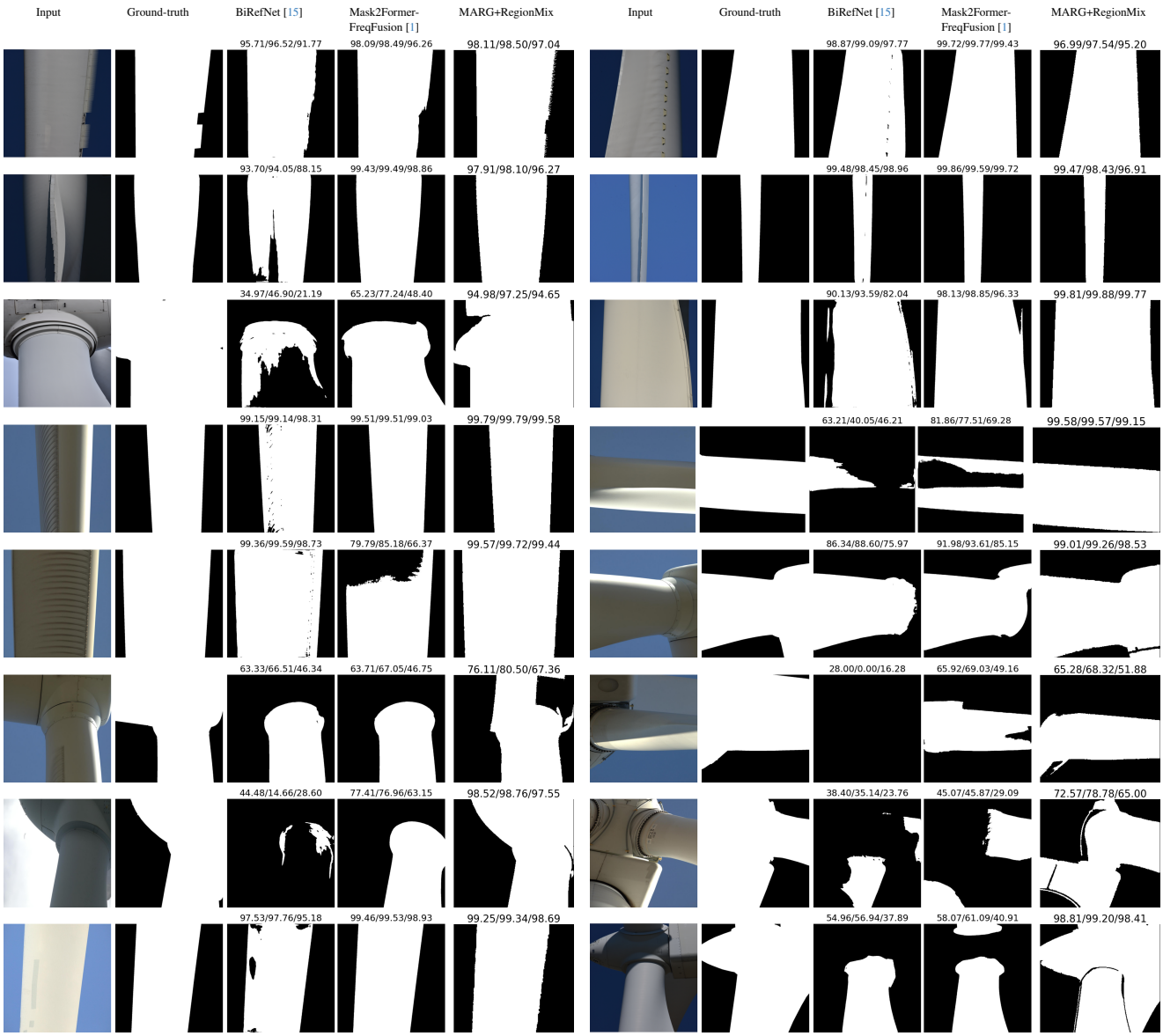


Figure 8. **Qualitative comparison of MARG+RegionMix with competing algorithms**. The first column presents the input color image, the second displays the ground-truth mask, and, from the third to the fifth, it shows the segmentation estimation of BiRefNet [15], Mask2Former-FreqFusion [1] and MARG+RegionMix, respectively. On top of each estimation, the accuracy, F1 and mIoU performance metrics are showcased, respectively. Both sides display the same information.

The dataset was collected from multiple wind turbine operators and inspection companies, capturing images while the turbines were stationary to ensure high-detail acquisition. This diversity was essential to ensure generalization across geographic locations and turbine types. Two type of imaging engines were employed. The first engine is based on ground-based robotic inspections which consist of a robotic arm equipped with a high-resolution industrial camera placed in the floor. The camera was typically positioned 50 to 100 meters from the blade and primarily captures the blade against a sky background. On the other side, aerial drone images were obtained from a closer perspective - typically operated at distances between 2 and 5 meters from the blade surface. These images typically included broader landscape backgrounds, which varied depending on the wind farm's location (onshore or offshore).

The training set is comprised of 1712 images, while the validation and test sets contain 120 and 200 images, respectively. These pictures have been gathered from different windfarms and inspection campaigns, ensuring their inde-

pendence and that the test emulates new data acquired, so we can fairly analyze the generalization of our approach. More specifically, the validation and test sets are formed from randomly selecting 20 images per windfarm. Moreover, the training data was selected from a pool of different windfarms, prioritizing the under-sampled instances that are harder to infer. These images cover the blade sections located at the root, tip or max-cord, commonly known as the shoulder of the blade. In this way, we ensure that we capture the structural diversity of blades, along its different in geometry and appearance.

Acknowledgements: This work has been supported by the projects GRAVATAR PID2023-151184OB-I00 funded by MCIU/AEI/10.13039/501100011033 and ERDF, UE; and by GreenVAR of the Fundación Ramón Areces.

References

[1] Linwei Chen, Ying Fu, Lin Gu, Chenggang Yan, Tatsuya Harada, and Gao Huang. Frequency-aware feature fusion for dense image prediction. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 46(12):10763–10780, 2024. 5, 7, 8

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6, 7

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6, 7

[4] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed W. Zamir, Fahad S. Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 54(10):1–41, 2022. 6

[5] Alexander Kirillov et al. Segment anything. In *International Conference on Computer Vision*, pages 4015–4026, 2023. 5

[6] Yanyu Li et al. Rethinking vision transformers for mobilenet size and speed. In *International Conference on Computer Vision*, 2023. 5

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*, pages 9992–10002, 2021. 6, 7

[8] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7086–7096, 2022. 5

[9] Sachin Mehta and Mohammad Rastegari. Mobilevit: Lightweight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022. 5

[10] Raül Pérez-Gonzalo, Andreas Espersen, and Antonio Agudo. Robust wind turbine blade segmentation from RGB images in the wild. In *IEEE International Conference on Image Processing*, pages 1025–1029, 2023. 5, 7

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 6, 7

[12] Mingxing Tan, Q Efficientnet Le, et al. Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 2019. 6, 7

[13] Junjiao Tian, Lavisha Aggarwal, Andrea Colaco, Zsolt Kira, and Mar Gonzalez-Franco. Diffuse attend and segment: Unsupervised zero-shot segmentation using stable diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3554–3563, 2024. 5

[14] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021. 6, 7

[15] Peng Zheng, Dehong Gao, Deng-Ping Fan, Li Liu, Jorma Laaksonen, Wanli Ouyang, and Nicu Sebe. Bilateral reference for high-resolution dichotomous image segmentation. *CAAI Artificial Intelligence Research*, 3:9150038, 2024. 5, 7, 8