Supplementary Material for "AMOD: A Large-scale Benchmark for RGB-T Multi-view Aerial Military Object Detection"

Yechan Kim*
GIST
Republic of Korea
yechankim@gm.gist.ac.kr

Jongmin Joo
GIST
Republic of Korea
luck2u99@gm.gist.ac.kr

JongHyun Park
GIST
Republic of Korea
citizen135@gm.gist.ac.kr

Jongmin Yu
University of Cambridge
United Kingdom
jy522@projectg.ai

Moongu Jeon
GIST
Republic of Korea
mgjeon@gist.ac.kr

§A Bounding box extraction from Arma3

This section describes the procedure used to automatically extract 2D bounding boxes (BBoxes), both horizontal bounding boxes (HBBs) and oriented bounding boxes (OBBs), from 3D objects in Arma3. For convenience, we assume the BBox extraction for a single object.

§A.1 Obtaining the model-space BBox

For each target object, we first invoke the function ***boundingBoxReal***, which returns the lower and upper corners of the object's 3D BBox in the object's (local) model coordinate system. Note that Arma3 assumes that each object is located at the origin of its own model space (m). Selecting index 1 yields the upper-corner coordinate (x_m, y_m, z_m); see Fig. 9(a). Given this upper-corner value, we enumerate all sign combinations of x_m, y_m, and z_m to generate the eight canonical model-space corner points $\{p_m^i\}_{i=1}^8$; see Fig. 9(b).

§A.2 Transforming to world coordinates

Each model (m)-space corner point is then mapped into the (global) Arma3 world coordinate frame (w) using the transformation function ***modelToWorldVisual***, producing world-space coordinates p_w^i for $i \in \{1, \cdots, 8\}$; see Fig. 9(c).

§A.3 Projecting into screen coordinates

To obtain 2D pixel-space coordinates, each world (w)-space point is projected onto the rendered image plane via the Arma3 camera projection function ***worldToScreen***; see Fig. 9(d). Though Fig. 9(d) does not explicitly depict it, an additional post-processing step is required to obtain the actual pixel coordinates from ***worldToScreen*** in Arma3. The function ***worldToScreen*** returns normalized screen coordinates expressed within the Arma3's *safe-zone* layout rather than the final pixel grid of the rendered image. Consequently, one must apply safe-zone compensation before obtaining valid pixel-space positions as follows.

Let $(q^i[x], q^i[y])$ denote the 2D coordinates returned by ***worldToScreen***. The final pixel coordinates $(p^i[x], p^i[y])$ are computed by removing the safe-zone offset (SafeZoneX, SafeZoneY) and scaling with respect to the safe-zone width (SafeZoneW) and height (SafeZoneH):

$$\begin{aligned} p^i[x] &= \frac{q^i[x] - \text{SafeZoneX}}{\text{SafeZoneW}} \times W_{\text{img}}; \\ p^i[y] &= \frac{q^i[y] - \text{SafeZoneY}}{\text{SafeZoneH}} \times H_{\text{img}}, \end{aligned} \tag{1}$$

where SafeZoneX, SafeZoneY, SafeZoneW, and SafeZoneH are constants in Arma3; see https://community.bistudio.com/wiki/safeZone for details. In this work, we set W_{img} and H_{img} to 1920 and 1440, respectively, thus generating images with a resolution of 1920×1440.

§A.4 Computing HBB and OBB in screen space

Given the projected points $\{p^i\}_{i=1}^8$, we compute both forms of BBox as follows:

- **Horizontal Bounding Box (HBB).** The HBB is obtained by taking the minimum and maximum of the x- and y-coordinates across all projected points; see Fig. 9(e).
- **Oriented Bounding Box (OBB).** The OBB is obtained by first computing the convex hull of the projected points using the Graham Scan algorithm. The smallest enclosing rectangle of this convex hull is then found using the Rotating Calipers method, yielding the final 2D OBB in screen coordinates; see Fig. 9(f).

§A.5 Summary

This pipeline enables accurate extraction of both HBB and OBB annotations directly from Arma3's internal geometry and rendering functions. Because the pipeline relies solely on engine-grounded transformations and camera projections, the resulting annotations are geometrically consistent and suitableas ground-truth labels for training and evaluating aerial object detection models.

§B Detailed experimental setup and more experimental results for Sec. 4.2 (Main)

Unless otherwise stated, all experiments here are conducted with the same detector, dataset, and optimization configuration. We adopt Oriented R-CNN with a Swin-S backbone as implemented in MMRotate, initialized from ImageNet-pretrained weights.

Dataset and angle splits. For all experiments here, we use the proposed AMOD dataset (RGB split only) with a default image resolution of 1920×1440 pixels. For *single-angle* training, we restrict the training set to one of these angles and keep the validation/test

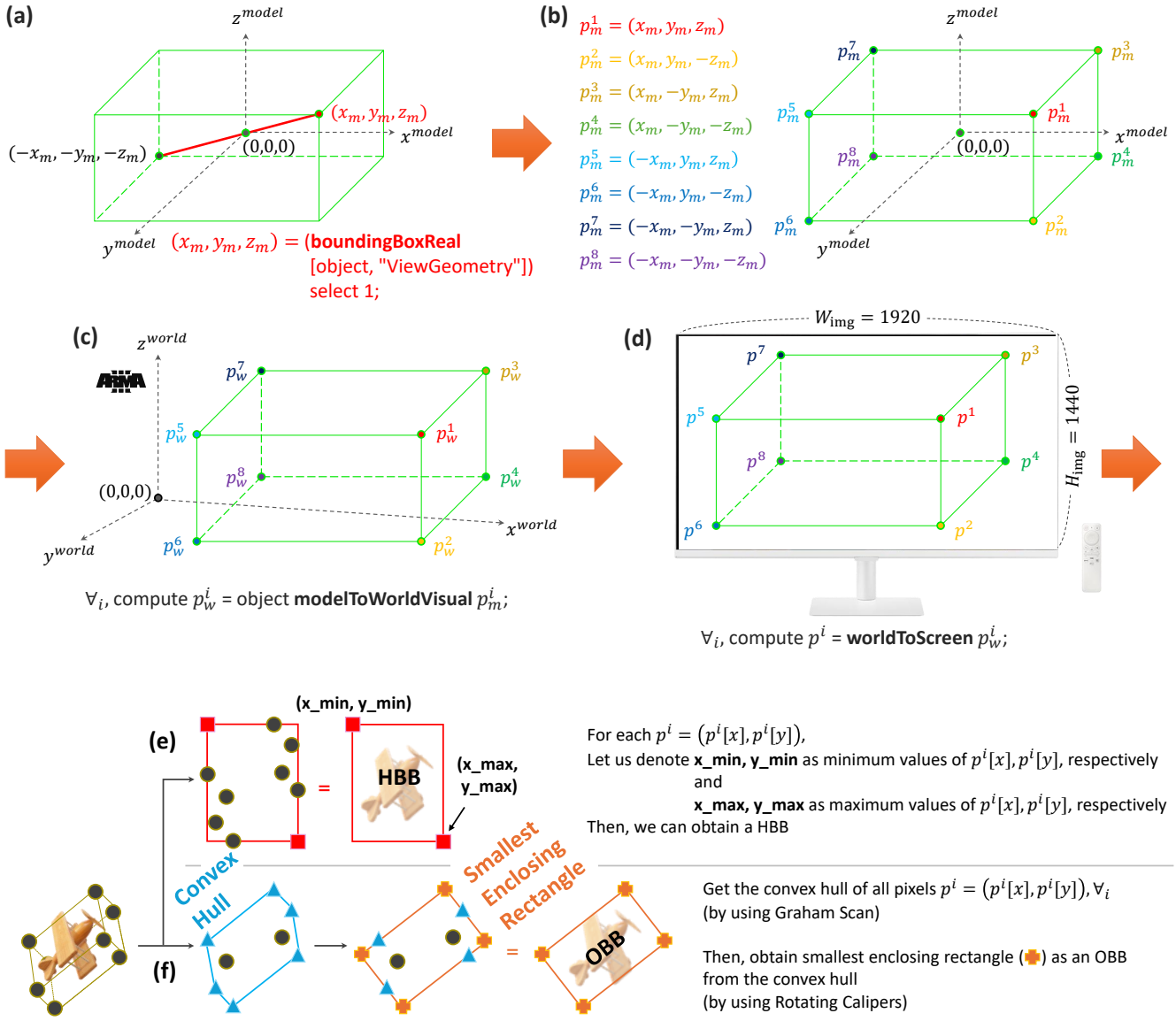

Figure 9: Overview of the pipeline for computing both a 2D horizontal bounding box (HBB) and an oriented bounding box (OBB) for a certain 3D object in Arma3. (a) The 'boundingBoxReal' function is called in Arma3, which returns the lower and upper corners of the object's bounding box (BBox) in the object's own (local) model coordinate system (m), assuming the object itself is located at the origin; selecting index 1 yields the upper-corner coordinate (x_m, y_m, z_m). (b) Using (x_m, y_m, z_m), all eight m-space corner points p_m^i $(i = 1, \cdots, 8)$ are generated by enumerating all sign combinations of $\pm x_m$, $\pm y_m$, and $\pm z_m$. (c) Each model-space corner is transformed into the (global) Arma3 world coordinate frame (w) by invoking the 'modelToWorldVisual' function of Arma3, producing p_w^i. (d) The w-space points are then projected into the image plane frame via the Arma3's 'worldToScreen' function, yielding $p^i \in \mathrm{R}^2$. (e) From these projected points, the HBB is obtained in screen coordinates by taking the minimum and maximum from x and y values in $\{p^i = (p^i[x], p^i[y])\}_{i=1}^{8}$. (f) The convex hull of all projected points is computed using Graham Scan, and the smallest enclosing rectangle of this hull is found using the Rotating Calipers algorithm, resulting in the final 2D OBB in screen coordinates.

sets unchanged (i.e. all angles are used individually). For *multi-angle* training, all six angles are used jointly during training. For the reduced-data experiments in Fig. 5 (Main), we further subsample training scenes with a nested strategy (e.g., $3/6 \subset 4/6 \subset 5/6 \subset 6/6$) while always preserving the full set of angles.

Table 6: Cross-angular evaluation results ($AP_{50:95}$) on AMOD.

	Test						
Train	0°	10°	20°	30°	40°	50°	Mean
0°	51.16	49.62	47.44	43.07	36.88	26.60	42.46
10°	50.92	50.01	48.04	44.79	38.76	28.39	43.49
20°	50.51	49.88	48.57	45.64	40.13	32.71	44.57
30°	48.67	48.27	47.42	45.69	41.46	33.66	44.20
40°	46.56	46.67	46.41	44.80	42.24	37.07	43.96
50°	38.23	38.20	39.04	39.29	37.95	36.49	38.20
All (6/6)	**56.18**	**54.75**	**53.93**	51.03	**47.67**	**42.20**	**50.96**
All (5/6)	55.59	54.53	53.71	**51.21**	47.67	41.49	50.70
All (4/6)	54.14	53.01	51.71	50.15	45.29	39.95	49.04
All (3/6)	52.83	51.76	51.31	48.80	44.86	39.06	48.10
All (2/6)	52.62	51.91	51.03	47.92	43.71	37.44	47.44
All (1/6)	52.48	51.36	50.52	47.71	43.32	37.45	47.14

Training pipeline. We adopt a standard MMRotate training pipeline tailored to AMOD. Each training image is first loaded together with its oriented bounding boxes. We then apply multi-scale resizing within the range (1536×1152) to (2340×1728), corresponding to a 0.8×–1.2× variation around the default 1920×1440 resolution. This multi-scale augmentation encourages robustness to varying spatial resolutions and object scales, effectively mimicking the resolution variability encountered in real aerial imaging. After resizing, we perform a 1024×1024 random crop while disallowing negative crops, followed by random flipping across three directions (horizontal, vertical, diagonal), each with a probability of 0.25. All geometric transforms follow the `le90` rotated-box convention. Images are subsequently normalized using the standard mean and variance employed by MMRotate, padded to the nearest multiple of 32 pixels to ensure compatibility with the Feature Pyramid Network (FPN), and finally packaged into PyTorch tensors.

Evaluation pipeline. For validation and test, we adopt a clean single-scale evaluation protocol without any test-time flipping. Each image is first loaded, followed by identical normalization and padding to ensure full compatibility with the detector backbone. The processed image is then converted into the standard PyTorch tensor format.

Optimization and learning schedule. All models are trained for 30 epochs with stochastic gradient descent (SGD) using a learning rate of 0.005, momentum of 0.9, and weight decay of 10^{-4}: We adopt a step-wise learning rate schedule with a 500-iteration linear warm-up phase (warm-up ratio of 1/3), followed by scheduled decays at epochs 16 and 22. Gradient clipping is applied with a maximum norm of 35 to stabilize optimization. All results reported are evaluated using the best checkpoint on the validation split.

More experimental results. See Tab. 6 for cross-angular evaluation results at $AP_{50:95}$, which summarize how detectors trained at one viewing angle perform when tested across all other angles. This provides a stricter localization-based comparison under the same cross-angle setting compared to AP_{75} in our main manuscript.

§C SAMBA: SAM-assisted Bounding Box Adjustment

§C.1 Motivation

While synthetic datasets are often expected to provide perfect annotations, we found that bounding boxes (BBoxes) generated by the Arma3 engine occasionally suffer from spatial misalignment; see Fig. 11. Contrary to the common assumption of label perfection, synthetic annotations can still exhibit *Loose*, *Over-Tight*, or *Shifted* BBoxes that degrade detector reliability. To address these imperfections, we propose SAM-assisted Bounding Box Adjustment (SAMBA), an automatic refinement algorithm that improves the localization precision of synthetic BBoxes.

SAMBA leverages the Segment Anything Model (SAM) [9] to automatically extract object masks for each initial BBox from Arma3. For each image $\mathcal{I}$ with K objects, let the initial BBoxes of these objects be denoted as $\{b_k^1, b_k^2, ..., b_k^p, ..., b_k^P\}$. The objective of our method is to correct each initial BBox b_k into a refined BBox b_k^*.

For simplicity, we describe our SAMBA algorithm under the simplified assumption where a single image $\mathcal{I}$ contains only one oriented bounding box (OBB) b_k.

- (Step 1) We input the image $\mathcal{I}$ and the BBox b_k into the SAM. b_k is first converted into a horizontal bounding box (HBB), which is the box format supported by SAM. SAM outputs multiple candidate foreground masks, each accompanied by a confidence score. We select the mask with the highest confidence and denote it as $\mathcal{M}_k$.
- (Step 2) Using the mask $\mathcal{M}_k$ obtained in (Step 1), we compute the smallest enclosing bounding box via convex hull [5] and rotating calipers [18]. We treat the resulting BBox as the refined OBB b_k^*[1].
- (Step 3) For all images and all BBoxes, we repeat Steps (1) and (2).

Yet, the success of SAMBA in refining BBoxes critically depends on the accuracy of the segmentation masks produced by the SAM. Hence, to enhance SAM's robustness, SAMBA integrates three auxiliary mechanisms: (i) padding-based BBox augmentation; (ii) negative-point prompting, which suppresses false-positive mask expansion; (iii) connected component filtering, which removes irrelevantly fragmented mask regions. The overall workflow, illustrated in Fig. 10, enables SAMBA to automatically correct localization biases inherent in our data. Although each mechanism is independently effective, we empirically confirm that SAM's reliability is highest when all three are used together; see Fig. 12(d).

§C.2 Trick (i) Padding-based BBox augmentation

There are cases where the image provided by the Arma3 game and the corresponding BBox are not perfectly aligned. In such cases, the BBox fails to fully encompass the object and becomes a *shifted* example. If this misaligned BBox is directly provided to SAM as a prompt, SAM may output an incomplete mask that does not include the whole object. To overcome this limitation, we propose a padding-based BBox augmentation strategy.

Let the initial BBox be denoted as b_k. If there exist P padding strategies in total, the augmented set of BBoxes can be expressed as $\{b_k^1, b_k^2, \cdots, b_k^p, \cdots, b_k^P\}$, where each b_k^p denotes a padded (i.e. enlarged) version of b_k. For each image and its corresponding augmented BBox pair $(\mathcal{I}, b_k^p)$, we denote the SAM-generated mask as $\mathcal{M}_k^p$. All masks $\mathcal{M}_k^p$ in the range $1 \le p \le P$ are then merged

[1] By extracting the minimum and maximum coordinates from the revised OBB, we can also derive an improved HBB.

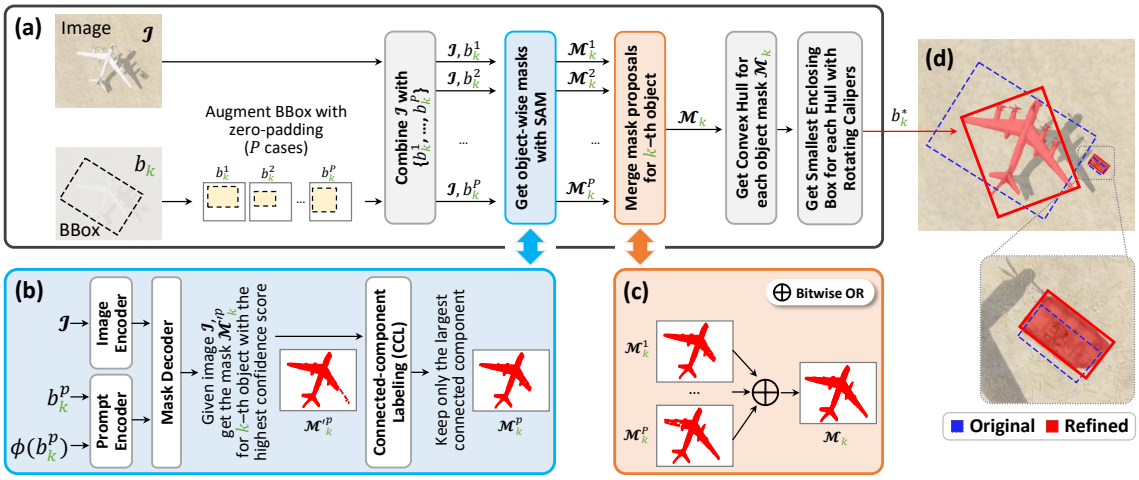

Figure 10: Pipeline of SAMBA. (a) The initial BBox b_k is augmented into P-padded variants $\{b_k^1, b_k^2, \cdots, b_k^p, \cdots, b_k^P\}$, each combined with the image $\mathcal{I}$ and fed into SAM. (b) For each padded box b_k^p, SAM predicts a mask $\mathcal{M}'^p_k$ using the highest-confidence proposal, followed by Connected-component Labeling (CCL) to retain only the largest connected component. (c) For each p, all refined masks $\mathcal{M}_k^p$ are merged via a bitwise OR operation to obtain the final robust mask $\mathcal{M}_k$. (d) The Convex Hull and Rotating Calipers are applied to $\mathcal{M}_k$ to compute the refined BBox (OBB) b_k^*, correcting misaligned (shifted, loose, or over-tight) annotations.

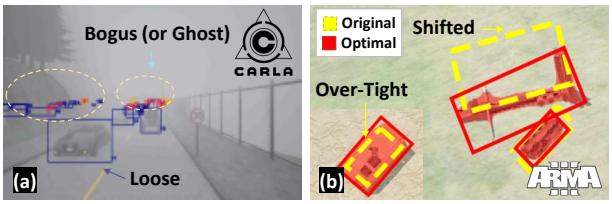

Figure 11: Examples of incorrect bounding boxes provided by (a) the Carla simulator and (b) the Arma3 game. Note that for (a), the image is adapted from [4] with partial modifications.

through a bitwise OR operation ($\bigvee$), and the resulting mask is used as $\mathcal{M}_k$, i.e. $\mathcal{M}_k = \bigvee_{p=1}^{P} \mathcal{M}_k^p$.

§C.3 Trick (ii) Negative-point prompting

SAM can accept one or more point prompts in addition to the BBox. These points are categorized into two types: positive points and negative points. A positive point indicates a location that belongs to the object, whereas a negative point indicates a location that does not belong to the object. After applying a certain padding strategy to this box, we denote the set of its four corner coordinates as $\phi(b_k)$. We provide $\phi(b_k)$ to SAM as negative point prompts together with the image $\mathcal{I}$ and the box b_k.

One may question why positive points are not used. The reason is that a positive point could easily be mistakenly placed at a location that does *not* belong to the true object. A positive point must lie inside the true object region, but any point in the range $b_k = \{(x_1, y_1), \cdots, (x_4, y_4)\}$ is only a candidate and not guaranteed to be a correct object point. For example, consider the center point of the BBox, $(\sigma_x, \sigma_y) = \left(\frac{x_1+x_2+x_3+x_4}{4}, \frac{y_1+y_2+y_3+y_4}{4}\right)$. From the 'Shifted' box illustrated in Fig. 11(b), we can compute the center (σ_x, σ_y) from the original BBox (denoted as 'Original'). However, upon inspection, even the center point of the Arma3-generated box does *not* necessarily lie on the object, as the generated box can be misaligned. This suggests that positive points derived from such noisy boxes may fall outside the true object region and thus mislead SAM. For this reason, our method does not incorporate positive points and instead relies solely on negative points.

§C.4 Trick (iii) Connected component filtering

Connected Component Labeling (CCL) [17] refers to the process of identifying groups of pixels that are mutually connected and treating each group as a single object. In our setting, we consider pixels to be connected horizontally, vertically, and diagonally, corresponding to 8-connectivity. The purpose of applying CCL to the fragmented mask $\mathcal{M}'^p_k$ in Fig. 10(b) is to remove outlier regions.

When CCL is applied to $\mathcal{M}'^p_k$, multiple connected components are generated. Among these components, SAMBA selects only the one with the largest area and assigns it as the final mask M_k, effectively discarding small and irrelevant fragments that act as outliers. For this, we employ a Two-pass CCL algorithm based on Union-Find-Tree [21].

§C.5 Experimental configuration of SAMBA

We use the official implementation of SAM [9]. Although heavier backbones such as ViT-H incur a longer inference time, we prioritize segmentation accuracy over computational cost, since BBox refinement is performed only once per sample and high-quality mask much matters for our task. For the padding-based BBox augmentation, we consider two zero-padding scales: `1.05×` and `1.10×`. In addition, we consider the four negative points by applying a 5-pixel outward padding to the original HBB in both the x- and y-axes.

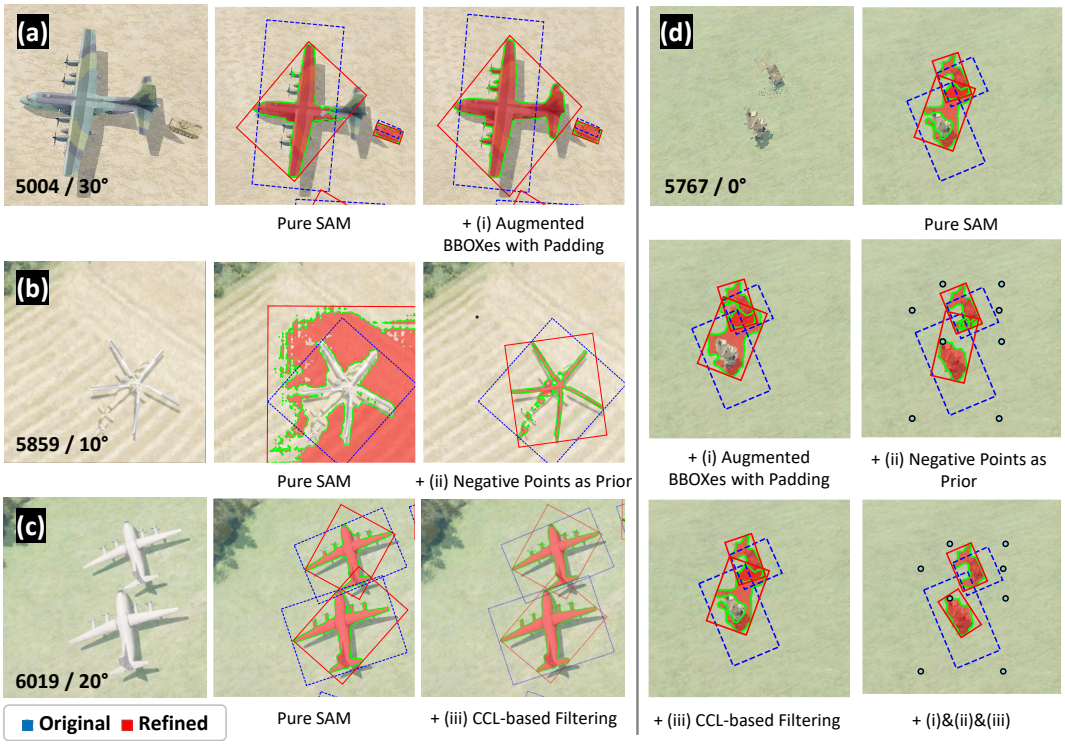

Figure 12: Qualitative results showing 'successful' BBox annotation refinement by SAMBA on the original AMOD dataset.

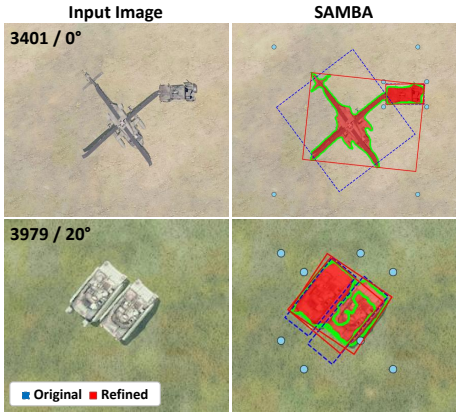

Figure 13: Qualitative examples illustrating 'failure' cases of SAMBA's BBox refinement on the original AMOD dataset.

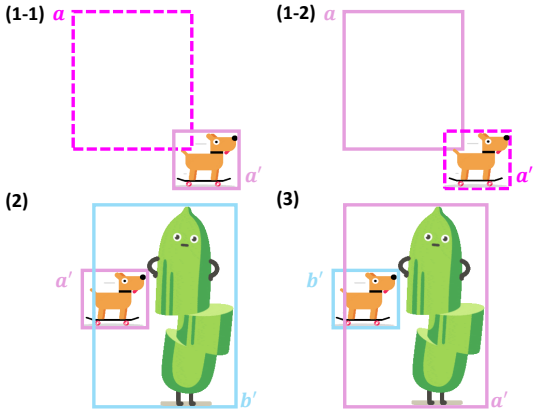

Figure 14: Schematic examples of ambiguous refinement cases requiring human-in-the-loop verification: (1) insufficient overlap between the original and refined bounding boxes, (2) the refined box a' is largely enclosed by another refined box b', or (3) b' is largely enclosed by a'.

§C.6 Qualitative results

Figs. 12 and 13 present qualitative analyses of SAMBA's BBox refinement on diverse instances from the original AMOD dataset, highlighting both successful and failure cases. Overall, the results illustrate how SAMBA improves the geometric alignment of the original annotations.

Successful cases. As shown in Fig. 12, SAMBA consistently refines the original BBoxes into more accurate refined BBoxes across a wide variety of aircraft types, viewing angles, and background textures. The refinement quality improves progressively when the individual components of SAMBA are activated:

- Padding-based Augmented BBoxes introduces additional visual context around the object, stabilizing SAM's initial segmentation and mitigating under-coverage of the object; see Fig. 12(a).
- Negative Points Prompting suppresses over-expansion by explicitly discouraging unwanted regions; see Fig. 12(b).
- CCL-based Filtering extracts object-consistent regions, preventing fragmented or noisy predictions from affecting the final refinement; see Fig. 12(c).

When all three components are combined, SAMBA produces the most stable and geometrically faithful BBoxes; see Fig. 12(d). These examples demonstrate that SAMBA effectively resolves SAM's raw segmentation errors and yields BBoxes that better capture the true footprint of each object.

Failure cases. Unfortunately, Fig. 13 highlights situations in which SAMBA fails to produce meaningful improvements. These failures occur primarily for two reasons:

(1) **Extreme object geometry or viewing conditions**: Thin structures (e.g., helicopter blades or tank guns) or highly skewed viewpoints reduce the reliability of SAM's initial mask. Because SAMBA relies on this mask as its starting point, subsequent refinement modules cannot recover from severely degraded initial segmentation masks.
(2) **Strong background interference and inter-object proximity**: Failure cases also arise when the image contains background textures that closely resemble the object (e.g., low-contrast terrain or shadows) or when multiple objects are positioned in close spatial proximity. In these cases, SAM often produces a mask that merges adjacent instances.

§C.7 Human-in-the-loop annotation refinement with SAMBA

Let two original BBoxes be denoted as a and b. After applying SAMBA, the refined BBoxes are written as a' and b', respectively. To ensure annotation reliability, we perform human-in-the-loop verification whenever the refined results satisfy any of the following three conditions.

(1) **Insufficient overlap between the original and refined BBoxes.** When $\frac{a\cap a'}{a'} < \theta_{\min}$ or $\frac{a\cap a'}{a} < \theta_{\min}$, the refined BBox a' exhibits almost no intersection with the original BBox a. This suggests that either the original annotation is incorrect as shown in Fig. 14(1-1) or SAMBA's refinement substantially deviated from the target object as shown in Fig. 14(1-2). Such cases require manual inspection and, when necessary, direct correction.
(2) a' **is largely enclosed by** b'**.** When
$$\frac{a' \cap b'}{a'} > \theta_{\max},$$
the refined BBox a' is mostly or entirely contained within b' as shown in Fig. 14(2). This situation typically arises when SAM's initial mask erroneously merges neighboring objects or regions. Human verification is therefore required to determine whether a' should indeed belong to b', followed by manual correction if needed.
(3) b' **is largely enclosed by** a'**.** When
$$\frac{a' \cap b'}{b'} > \theta_{\max},$$
the refined BBox b' lies almost entirely within a' as shown in Fig. 14(3). This symmetric condition also signals potential object merging or mis-segmentation due to SAM's initial prediction. Manual inspection is therefore required, followed by manual correction if necessary.

Note that, we use $\theta_{\min}$ and $\theta_{\max}$ as the lower and upper thresholds that determine insufficient overlap and excessive enclosure. We set $\theta_{\min} = 0.2$ and $\theta_{\max} = 0.6$.

The human-in-the-loop refinement stage serves as a safeguard complementing SAMBA's automatic BBox correction. It ensures that severely misaligned or ambiguous refinements are systematically identified and corrected, thereby maintaining high annotation fidelity.

§D Real-world Domain Adaptation

§D.1 Preliminary Background

Though our synthetic data offers scalable and precisely annotated samples, the visual characteristics of synthetic images significantly differ from real-world data in aspects such as color tone, texture, and background complexity [19]. This discrepancy is generally referred to as *domain gap*. Even in real-world scenarios, domain gaps can still naturally arise due to varying environmental factors, sensors, and imaging modalities [1, 20].

§D.2 Overall pipeline

To bridge this gap, we adopt a pipeline consisting of domain transfer, pretraining, and finetuning a detection model, as shown in Fig. 19. We employ UNIT [14] to transform our synthetic AMOD images ($\mathcal{X}_S$) into photo-realistic ones that resemble the target domain ($\mathcal{X}_T$). Both domains $\mathcal{X}_S$ and $\mathcal{X}_T$ are embedded into a shared latent space ($\mathcal{Z}$), from which visually realistic and target-oriented images are augmented[2], as illustrated in Figs. 15 and 17.

After translating AMOD images, we pretrain the object detector on these samples to learn domain-invariant features that generalize well across synthetic and target real data. Subsequently, we finetune the pretrained detector using labeled real-world data to adapt to the final target distribution.

§D.3 Stochastic Boundary Smoothing (SBS): a pretraining-specific augmentation for visible-to-thermal adaptation

As illustrated in Fig. 18, the process of translation into thermal-spectrum is inherently *ill-posed*. Consequently, this process is fundamentally a one-to-many translation, which no deterministic model can fully capture. For instance, though UNIT [14] we adopt produces visually coherent outputs, its deterministic nature results in limited representations that cannot reflect the diverse thermal variations encountered in real-world sensing. Unfortunately, existing studies [2, 6, 11, 13, 15, 16, 22] largely treat such translation as a deterministic mapping. A straightforward remedy might be stochastic image translation [8, 10, 12, 23]. However, this direction often introduces cumbersome network architectures or complicated training procedures like multi-step sampling.

To address this challenge, we avoid modifying the translator itself and introduce stochasticity through augmentation instead. Specifically, we apply perturbations to the pseudo-thermal translated images during detector pretraining. This allows us to rely

[2]The embedding in $\mathcal{Z}$ is fed into the generator $G_{S\to T}$ which transfers it from $\mathcal{X}_S$ to $\mathcal{X}_T$, producing a synthetic image in the target style. Both $G_{S\to T}$ and $G_{T\to S}$ are trained, yet we only utilize $G_{S\to T}$ of the UNIT.

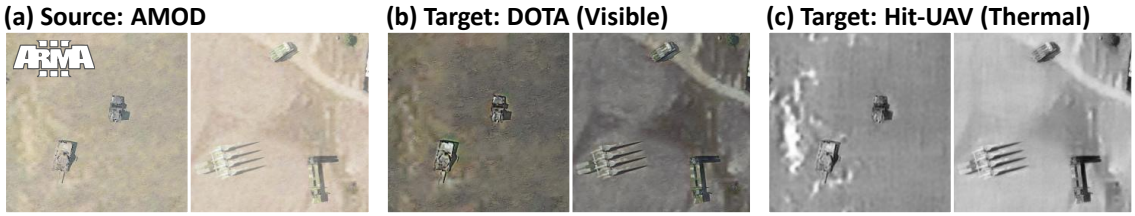

Figure 15: Examples of translated synthetic image from AMOD using UNIT.

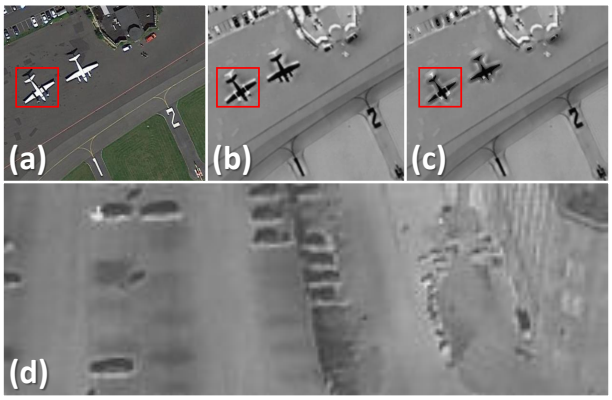

Figure 16: (a) Example visible image from DIOR-R. (b) thermal-style translation of (a) using a pretrained UNIT model. (c) Result after applying the proposed Stochastic Boundary Smoothing (SBS), where the object boundaries become intentionally blurred to better emulate the characteristic edge attenuation observed in real airborne thermal sensors. (d) Example real thermal image from HIT-UAV.

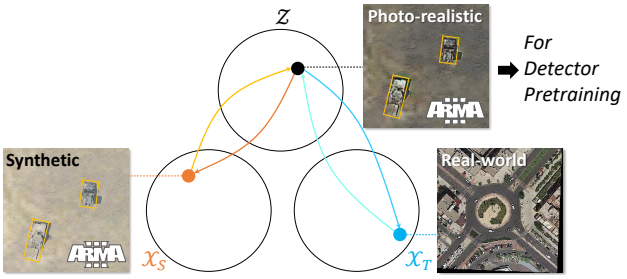

Figure 17: Illustration of the shared latent space ($\mathcal{Z}$) for domain adaptation between our synthetic ($\mathcal{X}_S$) and target real-world ($\mathcal{X}_T$) aerial image domains, inspired by UNIT [14]. Both domains are mapped to $\mathcal{Z}$, from which Arma3-rendered photo-realistic images are generated as intermediate samples to bridge $\mathcal{X}_S$ and $\mathcal{X}_T$; see Fig. 15 for translated synthetic image examples (from $\mathcal{Z}$) of AMOD.

on simpler deterministic translators while still expanding thermal variability.

To design a meaningful perturbation strategy, we take into account that thermal sensors suppress inner textures while preserving sharp boundary gradients, yet those boundaries can still become

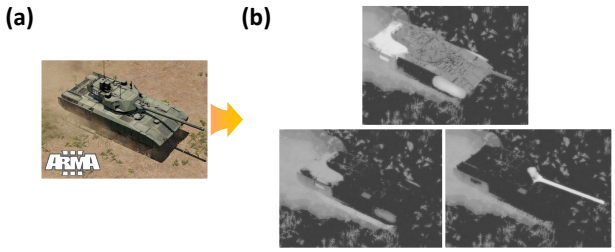

Figure 18: Example of visible-to-thermal variations. (a) shows the same visible image, while (b) presents multiple thermal renderings produced under varying ambient and object heat conditions in Arma3, with the time condition kept unchanged. It illustrates that visible-to-thermal translation is inherently ill-posed, as an identical optical input can correspond to diverse thermal outcomes.

blurred due to weak thermal contrast or sensor smoothing [3, 7]. Motivated by this fact, we introduce ***stochastic boundary smoothing*** (SBS), a simple but effective augmentation strategy during AMOD-pretraining for thermal-infrared detection, applied to each object at each iteration. Consequently, selectively smoothing contour mitigates the overly crisp edges in synthetic visible sources, effectively shrinking the visible-thermal domain gap, while improving thermal variability.

Let f denote the transform applied to the image in each iteration during pretraining. In Fig. 19, f is basically an identity mapping. However, for SBS, we replace f with:

$$f(\mathbf{X}) = \begin{cases} \mathbf{X}, & z = 0, \\ \mathbf{X} \odot (1 - \mathbf{C}) + G_\sigma(\mathbf{X}) \odot \mathbf{C}, & z = 1, \end{cases} \tag{2}$$

where $\mathbf{X}$ is a translated image (size: $w \times h$) and z is drawn from a Bernoulli distribution with a probability of p. $\odot$ denotes the Hadamard product. G_σ represents a 2D Gaussian blur operator parameterized by the standard deviation σ. $\mathbf{C}$ is the aggregated mask of contours over all instances as:

$$\mathbf{C} = \bigvee_i \Gamma(\mathbf{M}_i; \tau_i), \tag{3}$$

where $\bigvee$ denotes a pixel-wise logical OR operator applied over all contour masks. $\mathbf{M}_i \in \{0,1\}^{w \times h}$ is a mask of i-th object in $\mathbf{X}$. For $\mathbf{M}_i$, τ_i is a contour thickness randomly drawn from a uniform distribution $\mathcal{U}(\tau_{\min}, \tau_{\max})$. Γ is a contour-band operator that produces a binary band (shape: $\{0,1\}^{w \times h}$) that lies in the boundary of $\mathbf{M}_i$ with τ_i as:

$$\Gamma(\mathbf{M}_i; \tau_i)(x) = \mathbf{1}\left[\left|\mathrm{dist}(x, \partial\mathbf{M}_i)\right| \le \tau_i\right], \tag{4}$$

where x denotes each pixel of $\mathbf{X}$ and $\mathrm{dist}(x, \partial\mathbf{M}_i) = \min_{y \in \partial\mathbf{M}_i} ||x - y||_2$. $\partial\mathbf{M}_i$ is a contour of i-th object.

For SBS, each instance-wise mask $\mathbf{M}_i$ is automatically extracted by Segment Anything Model (SAM) [9] with bounding box prompts;

Table 7: Evaluation results (AP_{50}) on HIT-UAV. Effect of the SBS perturbation (f: SBS) under AMOD pretraining.

Pretrained from		Finetuned for HIT-UAV				
Source Data	Pretraining-specific perturbation	Person	Car	Others	Mean	Δ
ImageNet	–	85.30	81.29	57.12	74.57	–
AMOD° (■)	✗	86.59	82.00	58.51	75.70	+1.13
	✓ (f: SBS)	87.60	81.91	61.40	76.97	+2.40

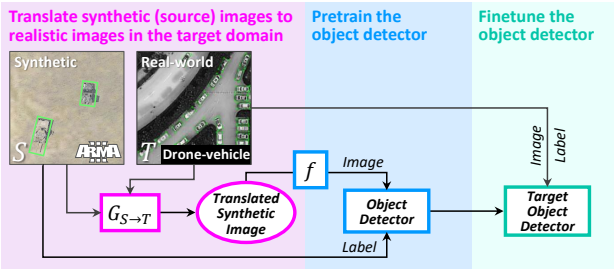

Figure 19: Overall domain adaptation pipeline used in this work. Here, $\mathcal{X}_T$ is a proxy target-style domain used for translation during pretraining, which may differ from the final downstream benchmark used for finetuning/evaluation.

see Sec. §C for implementation details of SBS. Note that while SBS is designed for AMOD-based pretraining, it could potentially be extended to other datasets as well, as shown in Fig. 16. SBS can be seamlessly incorporated into other datasets as well, as can be seen in Fig. 16. We begin by extracting object masks from a DIOR-R visible (see Fig. 16(a)) image using SAM, translate the image into a thermal-like domain with a pretrained UNIT model (see Fig. 16(b)), and then apply SBS (see Fig. 16(c)). The difference between (b) and (c) shows that SBS deliberately smooths object boundaries to replicate the characteristic edge attenuation observed in real airborne thermal sensors (see Fig. 16(d)). This observation confirms that SBS can serve as a general-purpose augmentation module by boundary smoothing for enhancing thermal-style realism across diverse datasets. We defer this to future work, as the primary focus of this paper is on the AMOD dataset.

Adopting our ***stochastic boundary smoothing*** (SBS) (see Sec. §D.3) during pretraining consistently leads to additional performance gains. For all experiments, $p = 0.5$, $\sigma = 11$, $\tau_{\min} = 2$, $\tau_{\max} = 5$ are used in SBS. The performance reported on the HIT-UAV thermal benchmark in Tab. 4 (Main) is obtained after applying AMOD pretraining, where our SBS method is incorporated during the pretraining stage. A comparison of performance before and after applying SBS is provided in Tab. 7. ***This simple perturbation helps narrow the visible-to-thermal gap by alleviating overly sharp boundaries of source visible images, while mimicking the spatial patterns of thermal imagery***.

§D.4 Detailed experimental setup and more analysis for Sec. 4.4 (Main)

Pretraining setup. We follow the same training configuration described in Sec. §B, except for the training epochs. For all experiments in Sec. F, the detector is pretrained for 5 epochs. Before pretraining, AMOD is translated into the corresponding target-domain style; see Sec. 4 (Main) for details.

Finetuning setup. After pretraining, only the backbone weights are transferred to the downstream detector. The detection head is **re-initialized** to match the number of classes of the target dataset (e.g., DIOR-R, DroneVehicle, HIT-UAV). Finetuning is conducted for 25 epochs, with all optimizer parameters (learning rate, momentum, weight decay, LR schedule, and warm-up) based on the conventional training protocol used for each target dataset.

Generalization beyond military-oriented semantics. Although AMOD is constructed entirely from military-oriented 3D assets, an interesting observation arises from our evaluations: ***pretraining on AMOD consistently improves generalization to real-world benchmarks whose semantic categories are not directly related to military equipment*** (e.g., DIOR-R[3], DroneVehicle, HIT-UAV).

More strikingly, AMOD-pretraining facilitates thermal-infrared generalization more effectively than pretraining on real optical datasets such as DOTA or DIOR-R; see Tabs. 4 and 5 (Main). This indirectly suggests that AMOD helps the models to capture structural or geometric factors that are particularly relevant for downstream aerial detection. One plausible explanation is that the *multi-angular captures* in AMOD expose the model to a wide spectrum of perspective distortions and contour variations that frequently occur in real aerial imagery, regardless of object semantics. As a result, AMOD effectively serves as a form of geometric pretraining, providing robust inductive biases that extend beyond the dataset's own semantic domain. We find this cross-semantic generalization particularly interesting, as most real-world benchmarks contain object types markedly different from those in AMOD, yet still benefit from AMOD-pretrained representations.

References

[1] Eunsu Baek, Keondo Park, Jiyoon Kim, and Hyung-Sin Kim. 2024. Unexplored faces of robustness and out-of-distribution: Covariate shifts in environment and sensor domains. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 22294–22303.

[2] Mahroosh Banday and Brejesh Lall. 2025. Multi spectral visible-thermal IR image translation using improved u-net & conditional diffusion. *Neurocomputing* (2025), 131006.

[3] Lijing Cai, Xiangyu Dong, Kailai Zhou, and Xun Cao. 2024. Exploring video denoising in thermal infrared imaging: Physics-inspired noise generator, dataset, and model. *IEEE Trans. Image Process.* 33 (2024), 3839–3854.

[4] Mohamad Mofeed Chaar, Jamal Raiyn, and Galia Weidl. 2025. Improve bounding box in Carla Simulator. *arXiv preprint arXiv:2509.16773* (2025).

[5] Ronald L. Graham. 1972. An efficient algorithm for determining the convex hull of a finite planar set. *Info. Proc. Lett.* 1 (1972).

[6] Zonghao Han, Shun Zhang, Yuru Su, Xiaoning Chen, and Shaohui Mei. 2024. DR-AVIT: Toward diverse and realistic aerial visible-to-infrared image translation. *IEEE Trans. Geosci. Remote Sens.* 62 (2024), 1–13.

[7] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. 2015. Multispectral pedestrian detection: Benchmark dataset and baseline. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 1037–1045.

[8] Beomsu Kim, Gihyun Kwon, Kwanyoung Kim, and Jong Chul Ye. 2024. Unpaired image-to-image translation via neural schr\" odinger bridge. In *Int. Conf. Learn. Represent. (ICLR)*.

[9] Alexander Kirillov et al. 2023. Segment anything. In *IEEE Int. Conf. Comput. Vis. (ICCV)*.

[10] Yunwei Lan, Zhigao Cui, Xin Luo, Chang Liu, Nian Wang, Menglin Zhang, Yanzhao Su, and Dong Liu. 2025. When Schrodinger Bridge Meets Real-World Image Dehazing with Unpaired Training. In *IEEE Int. Conf. Comput. Vis. (ICCV)*. 8756–8765.

[11] Dong-Guw Lee, Myung-Hwan Jeon, Younggun Cho, and Ayoung Kim. 2023. Edge-guided multi-domain rgb-to-tir image translation for training vision tasks with challenging labels. In *IEEE Int. Conf. Robot. Autom. (ICRA)*.

[3]Note that we intentionally train and evaluate only classes associated with mobile objects (e.g., *Airplane*, *Ship*, and *Vehicle*), while excluding static categories like *Tennis court* and *Stadium*.

[12] Bo Li, Kaitao Xue, Bin Liu, and Yu-Kun Lai. 2023. Bbdm: Image-to-image translation with brownian bridge diffusion models. In *IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 1952–1961.

[13] Na Li, Haining Wang, Huijie Zhao, and Wen Ou. 2025. Cross-Modal Visible-to-Infrared Image Translation in Remote Sensing Guided by Thermal Features. *IEEE Trans. Geosci. Remote Sens.* (2025).

[14] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised image-to-image translation networks. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*. 1–9.

[15] Decao Ma, Juan Su, Bing Li, Yong Xian, Shaopeng Li, and Yao Ding. 2025. Self cycle strategy for unpaired visible-to-infrared image translation. *Pattern Recognit.* (2025), 112253.

[16] Mehmet Akif Özkanoğlu and Sedat Ozer. 2022. InfraGAN: A GAN architecture to transfer visible images to infrared domain. *Pattern Recognit. Lett.* 155 (2022), 69–76.

[17] Azriel Rosenfeld et al. 1966. Sequential operations in digital picture processing. *J. ACM* 13, 4 (1966).

[18] Godfried T Toussaint. 1983. Solving geometric problems with the rotating calipers. In *IEEE Melecon*, Vol. 83.

[19] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. 2021. Pixel-wise crowd understanding via synthetic data. *Int. J. Comput. Vis.* 129, 1 (2021), 225–245.

[20] Ancong Wu, Wei-Shi Zheng, Hong-Xing Yu, Shaogang Gong, and Jianhuang Lai. 2017. RGB-infrared cross-modality person re-identification. In *IEEE Int. Conf. Comput. Vis. (ICCV)*. 5380–5389.

[21] Kesheng Wu et al. 2009. Optimizing two-pass connected-component labeling algorithms. *Pattern Anal. Appl.* 12, 2 (2009).

[22] Lei Zhao, Mengwei Li, Bo Li, and Xingxing Wei. 2025. Diverse Visible-to-Thermal Image Translation via Controllable Temperature Encoding. *IEEE Trans. Multimedia* (2025).

[23] Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. 2022. Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*. 3609–3623.