LOCALIZED MULTI-OBJECT WATERMARKING

Anonymous authors
Paper under double-blind review

ABSTRACT

As images are increasingly reused and recomposed, tracing the provenance of individual objects requires watermarks that remain recoverable after partial theft. Beyond conventional global image watermarking, localized watermarking has recently emerged to protect individual objects within an image. However, localized watermarks remain vulnerable to object-level partial theft, where cropping, transformation, and recomposition of a watermarked object may preserve only a small portion of the embedded signal. Furthermore, the problem becomes more challenging when a composite image contains multiple objects, requiring independent message extraction and localization. To address these challenges, we propose MoMark, a localized multi-object watermarking framework that embeds independent messages into localized regions and jointly recovers their messages and spatial support. We employ a two-stage training strategy: single-object message extraction and localization, followed by multi-object training with partial-support objectives. At inference, MoMark pools messages over local patches and separates or merges watermarked regions by message similarity, without requiring ground-truth masks. Evaluated on complex segmentation masks with multiple independently watermarked objects, MoMark improves object-level message extraction and localization across geometric distortions, non-geometric distortions, and a variety of partial-theft settings.

1 INTRODUCTION

Image watermarking is a primary tool for copyright protection and provenance tracing, and its importance has grown with the spread of AI-generated content. Existing approaches include post-hoc watermarking (Al-Haj, 2007; Zhu et al., 2018; Xu et al., 2025; Shih, 2017; Tancik et al., 2020) and in-generation watermarking (Wen et al., 2023; Yang et al., 2024; Li et al., 2025; Ci et al., 2024; Asnani et al., 2024; Gunn et al., 2025; Lu et al., 2025). Most of these methods embed a single message in the entire image and implicitly assume that the image is reused as a whole. This assumption does not hold when only part of an image is reused. In *partial image theft* (Liu et al., 2025), an adversary cuts out a selected region of a watermarked image, such as a single object, applies geometric transformations such as shrinking, rotation, or translation, and pastes the transformed region onto another image. Only a small, transformed fraction of the original watermark signal then survives, and conventional global decoders often fail to recover the message.

Partial theft makes the object, rather than the image, the unit of reuse, so provenance must be traced at the object level. This changes the extraction problem in two ways. First, a message must be recoverable from whatever support survives, which may be a small, cropped, or rescaled portion of the original object. Second, a composite assembled from stolen objects naturally contains several independently watermarked instances. Provenance extraction therefore requires determining, from the attacked image alone, which messages are present and where each one lies, even when each instance survives only as a small transformed fragment and the number of instances is unknown.

While prior work has advanced robustness to partial theft and the embedding of multiple localized messages, existing multi-message evaluations primarily consider spatially separated regions and do not fully characterize per-instance message recovery and localization. WOFA (Liu et al., 2025) improves robustness to partial theft through theft-oriented mask augmentation, but it associates a single global message with each source image and therefore does not address composites containing multiple independently watermarked objects. Localized methods such as WAM (Sander et al., 2025) and MaskWM (Hu et al., 2026) embed different messages in different regions of the same

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

image. However, joint message extraction and instance localization from transformed subregions of irregular objects remain insufficiently characterized in existing works.

We propose *MoMark*, a post-hoc framework for *localized Multi-object waterMarking* that embeds an independent message into each object instance and recovers the message and location of each object from a watermarked image. Since a single image may contain objects with different messages, the extractor decodes messages at the region level. The extractor predicts a mask logit for watermark presence and bit logits at each pixel, and obtains the message of a region by pooling the bit logits weighted by the softmax-normalized mask logits. We train the extractor in two stages: single-object extraction and localization, followed by multi-object and partial-support recovery, in which the extractor learns to localize and decode small surviving fragments of watermarked objects. Since the number of watermarked objects is unknown at inference, MoMark groups patches with similar pooled messages into objects and refines the boundaries between them, which separates adjacent objects and merges disconnected fragments of the same object without ground-truth masks.

Our contributions are summarized as follows:

- We formulate multi-object watermarking for object-level provenance under partial theft, embedding independent messages into irregular object regions and recovering both the messages and their spatial support, and evaluate it with instance-level metrics for message recovery and localization.
- We propose MoMark, which decodes messages at the region level and is trained to localize and decode small fragments of multiple watermarked objects. At inference, it groups regions by the similarity of their decoded messages, recovering an unknown number of objects without ground-truth masks or known messages.
- We evaluate multi-object watermarking under geometric and non-geometric distortions and diverse partial-theft scenarios, using instance-level bit accuracy and localization metrics, demonstrating improved robustness over WOFA, WAM, and MaskWM and generalization to more objects than seen during training.

2 Related Work

Localized watermarking. Conventional global image watermarking methods typically associate an image with a single message, either through post-hoc embedding (Al-Haj, 2007; Zhu et al., 2018; Xu et al., 2025; Shih, 2017; Tancik et al., 2020) or during image generation (Wen et al., 2023; Yang et al., 2024; Ci et al., 2024; Asnani et al., 2024; Gunn et al., 2025). However, images may combine independently sourced objects that can be selectively extracted and reused, motivating region-specific message embedding and attribution. WAM (Sander et al., 2025) predicts watermark presence and message bits at each pixel and separates multiple messages using DBSCAN over pixel-wise message predictions. MaskWM (Hu et al., 2026) introduces explicit mask conditioning into the Encoder–Distortion–Decoder framework and supports region-specific embedding with different message lengths. For multi-message extraction, it partitions the predicted watermark mask using connected-component analysis and decodes each region independently. Both methods extend image-level attribution by recovering multiple localized messages within a single image.

Watermarking under image manipulation. Robust watermarking aims to preserve message recovery under changes in image appearance or spatial correspondence. HiDDeN (Zhu et al., 2018) incorporates training distortions such as cropping, blurring, pixel dropout, and JPEG compression, while StegaStamp (Tancik et al., 2020) addresses perspective and photometric distortions arising from printing and photography. RoPaSS (Ma et al., 2025) addresses partial screen-shooting through symmetric watermark patterns and parameter estimation for image resynchronization. WOFA (Liu et al., 2025) explicitly considers *partial image theft*, incorporating masked-region extraction, geometric transformations, and recomposition onto another host image into training. These approaches primarily target single-message recovery rather than the joint localization and attribution of multiple independently watermarked objects. Building on localized watermarking and transformation robustness, we study multi-object localization and attribution under partial image theft. In this setting, independently watermarked objects may be cropped, rescaled, occluded, or repositioned before being combined into a new image, requiring each recovered message to remain associated with the correct surviving object region.

3 METHOD

MoMark embeds an independent message into each object of an image and, after manipulation, recovers the message and region of each object without access to the object masks, the embedded messages, or the number of objects.

3.1 ARCHITECTURE AND OBJECT-LEVEL DECODING

Embedder. Given an image x, an object mask M, and a binary message $\ell \in \{0,1\}^b$, the embedder predicts a residual $\delta = E_\theta(x, M, \ell)$. A UNet encodes the image and mask and fuses their features with a learned message representation. The residual is modulated by a just-noticeable-difference (JND) map and a gain s:

$$\widetilde{x}_{\mathrm{wm}} = \mathrm{clip}_{[-1,1]}\left(x + \mu s\,\mathrm{JND}(x) \odot \delta\right), \qquad x_{\mathrm{wm}} = M \odot \widetilde{x}_{\mathrm{wm}} + (1 - M) \odot x,$$

where $\mu = 1.0$ controls the JND scaling. The final composition preserves the original pixels outside the target mask. For multi-object images, each object is assigned an independent message, its residual is computed from the same original image, and the masked residuals are accumulated to form the watermarked image.

Extractor. The extractor first applies a three-level stationary wavelet transform (SWT) and processes the resulting multi-scale representation with a convolutional encoder–decoder (UNet). SWT exposes spatially aligned frequency responses at multiple scales, which is intended to facilitate the extraction of weak signals while preserving the spatial correspondence required for localization (ablation in Appendix B.2). Let Ω denote the set of all spatial locations in the extractor output. At each $p \in \Omega$, the extractor predicts a mask logit a_p indicating watermark presence and bit logits $z_p \in \mathbb{R}^b$. The extractor pools the bit logits over a candidate region R using a_p:

$$\mathcal{P}(R) = \sum_{p \in R} \alpha_p(R) z_p, \qquad \alpha_p(R) = \frac{\exp(a_p)}{\sum_{q \in R} \exp(a_q)}, \tag{1}$$

where $\mathcal{P}(R) \in \mathbb{R}^b$ denotes the region-level message logits, weighted by the predicted mask.

Object-level decoding. Object-level decoding groups these pixel-wise predictions into watermarked objects and recovers the message of each (Figure 1(B)). We first partition the image into 32×32 patches and pool the bit logits over the predicted foreground pixels within each patch using $\mathcal{P}$. Patches with similar decoded messages are grouped into object candidates, whose combined supports are re-pooled to obtain object-level messages. This message-based grouping separates adjacent regions carrying different messages while associating spatially disconnected regions that carry the same message. We then refine object boundaries on a grid of $C \times C$ cells. A cell is considered a boundary cell when its 3×3 cell neighborhood contains at least two different objects. For each boundary cell, we pool the bit logits over its predicted foreground pixels and reassign these pixels to the neighboring object whose message has the highest bit agreement. All foreground pixels within the cell are then reassigned to the object whose message has the highest agreement; background pixels remain unchanged. Then we re-pool the updated regions and repeat this procedure n times. Decoding settings, cell eligibility criteria, and ablations on C and n are provided in Appendix B.2.

3.2 TRAINING FOR LOCALIZED MULTI-OBJECT WATERMARKING

We train the embedder and extractor jointly, first on single-object images to establish message extraction and localization, and then on multi-object images and partial watermarked regions (Figure 1(A)). During the first phase, JND modulation, image distortions, and stronger fidelity terms are progressively enabled as message recovery improves.

Single-Object Extraction and Localization. Given a binary message $\ell \in \{0,1\}^b$ and its segmentation mask M, we pool over the full domain and optimize

$$\mathcal{L}^s_\ell = \mathrm{BCE}\left(\mathcal{P}_{\mathcal{A}(x_{\mathrm{wm}})}(\Omega), \ell\right), \qquad \mathcal{L}^s_M = \mathrm{BCE}\left(a(\mathcal{A}(x_{\mathrm{wm}})), \mathcal{A}(M)\right), \tag{2}$$

where $\mathcal{A}$ is a training distortion (e.g., rotation and resizing). Because the mask logits also determine the pooling weights, mask supervision jointly learns the watermark support and where

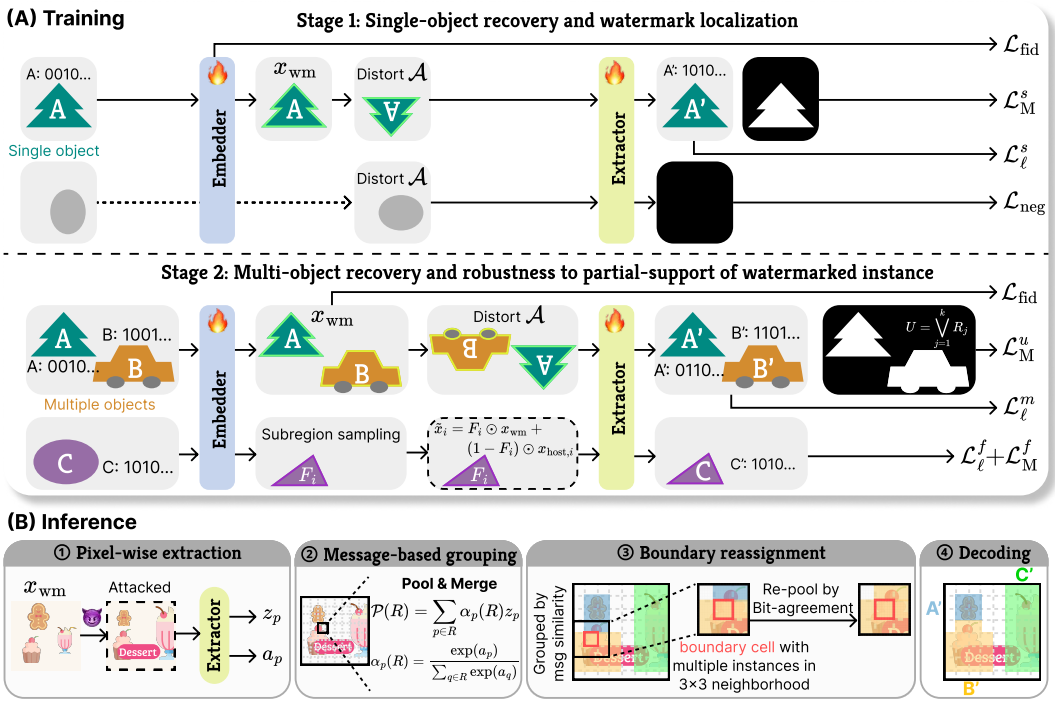

Figure 1: Overview of MoMark's two-stage training and inference pipeline.

reliable message evidence should be pooled. To suppress false-positive predictions, we additionally supervise the predicted mask of an unwatermarked image x_{unwm} with an all-zero target, $\mathcal{L}_{\text{neg}} = \text{BCE}(a(x_{\text{unwm}}), 0)$. We also regularize image fidelity using

$$\mathcal{L}_{\text{fid}} = \lambda_{\text{mse}} \, \text{MSE}(x, x_{\text{wm}}) + \lambda_{\text{chr}} \mathbb{E}[\Delta^2_{Cb} + \Delta^2_{Cr}], \tag{3}$$

where C_b and C_r denote the linear chrominance projections of an RGB residual. The chroma penalty discourages color shifts and helps reduce visible color artifacts, as illustrated in Figure 6. The Stage 1 objective is therefore

$$\mathcal{L}_1 = \lambda_\ell \mathcal{L}^{\text{s}}_\ell + \lambda_{\text{M}} \mathcal{L}^{\text{s}}_{\text{M}} + \lambda_{\text{neg}} \mathcal{L}_{\text{neg}} + \mathcal{L}_{\text{fid}}. \tag{4}$$

Multi-Object and Partial-support Recovery. To train for multi-object watermarking, we sample up to $k = 3$ non-overlapping objects from the training set and assign an independent b-bit message $\ell_j \in \{0,1\}^b$ to each object. Let R_j denote the watermarked region of object j, given by its mask. We supervise the message of each object with $\mathcal{L}^m_\ell$, pooled over its own region, and localization with $\mathcal{L}^u_M$ on the union mask $U = \bigvee_{j=1}^{k} R_j$:

$$\mathcal{L}^m_\ell = \frac{1}{k} \sum_{j=1}^{k} \text{BCE}\left(\mathcal{P}_{\mathcal{A}(x_{\text{wm}})}(\mathcal{A}(R_j)), \ell_j\right), \qquad \mathcal{L}^u_M = \text{BCE}\left(a(\mathcal{A}(x_{\text{wm}})), \mathcal{A}(U)\right). \tag{5}$$

Multi-object supervision alone still assumes that sufficient object support remains available for message pooling. To make message extraction robust to severe reductions in spatial support, as can occur under partial theft, we explicitly train the extractor to recover the complete message from small surviving regions. Specifically, we construct additional partial-support composites from single-object samples that are not used by the multi-object supervision. For each selected source object, we sample connected subregions $F_i \subseteq M$, each targeting a log-uniformly sampled fraction between 5% and 50% of the original object-mask area. Each subregion is then pasted onto a clean image:

$$\tilde{x}_i = F_i \odot x_{\text{wm}} + (1 - F_i) \odot x_{\text{host},i}. \tag{6}$$

The model recovers the complete b-bit message from each fragment and localizes its surviving support using

$$\mathcal{L}^f_\ell = \frac{1}{N_f} \sum_{i=1}^{N_f} \text{BCE}\left(\mathcal{P}_{\tilde{x}_i}(\Omega), \ell_i\right), \qquad \mathcal{L}^f_M = \frac{1}{N_f} \sum_{i=1}^{N_f} \frac{1}{2} \left(\mathcal{L}^{\text{fg}}_{M,i} + \mathcal{L}^{\text{bg}}_{M,i}\right), \tag{7}$$

where N_f is the total number of fragment composites generated in the current batch and ℓ_i is the corresponding source message. $\mathcal{L}_{M,i}^{\mathrm{fg}}$ and $\mathcal{L}_{M,i}^{\mathrm{bg}}$ denote binary cross-entropy losses computed from the predicted mask logits $a(\tilde{x}_i)$, averaged separately over the foreground and background pixels defined by F_i, respectively. Because messages are pooled over the full domain Ω rather than over F_i, the extractor must locate the surviving subregion and recover the message from it.

Finally, to preserve image quality while learning the multi-object and partial-support recovery in Stage 2, we additionally impose a PSNR-budget penalty on image fidelity ($\mathcal{L}_{\mathrm{fid}}$); $\lambda_{\mathrm{psnr}} \max\left(0, \frac{\mathrm{MSE}(x, x_{\mathrm{wm}})}{\epsilon_\tau} - 1\right)$, where $\epsilon_\tau = 4 \cdot 10^{-\tau/10}$ is the MSE budget corresponding to a target PSNR $\tau = 40$ dB for images in $[-1, 1]$. The penalty is activated only when the embedding distortion exceeds this budget, equivalently when the PSNR falls below τ. Thus, this enforces a soft lower bound on embedding PSNR rather than matching it to an exact value. The Stage 2 objective is

$$\mathcal{L}_2 = \lambda_\ell \left(\mathcal{L}_\ell^{\mathrm{m}} + \mathcal{L}_\ell^{\mathrm{f}}\right) + \lambda_{\mathrm{M}} \left(\mathcal{L}_{\mathrm{M}}^{\mathrm{u}} + \mathcal{L}_{\mathrm{M}}^{\mathrm{f}}\right) + \mathcal{L}_{\mathrm{fid}}. \tag{8}$$

4 Experiments

4.1 Experimental Setup

Baselines. WOFA[1] embeds a single global watermark without per-object conditioning, so it is reported only at $k = 1$. Its released weights are trained at 200×200 with a 30-bit message and operate only at that resolution. For WAM, we use the released COCO checkpoint, which supports only 32-bit messages, with $k \geq 1$ messages embedded. At $k > 1$, we cluster its per-pixel bit field with DBSCAN ($\varepsilon = 1$, $\mathrm{min_samples} = 1000$), following the configuration of WAM. For MaskWM, we use MaskWM-ED, the variant that conditions both the encoder and the decoder on masks so that watermarks are embedded in designated local regions, with its released 32-, 64-, and 128-bit checkpoints. The architectures, capacity, and training time of each baseline are summarized in Table 4. Examples of watermarked images and residuals ($\times 10$) are shown in Figure 2. For comparison, the watermarked images of all methods are matched to a PSNR of approximately 39 dB per image before attacks (Appendix A.2).

Implementation Details. We train MoMark on COCO train2017 (Lin et al., 2014), which contains 118,287 images with object segmentation masks. For single-object training, we sample embedding masks from four types: irregular masks (16%), rectangular boxes (50%), full-image masks (16%), and object segmentation masks (18%). Training uses a resolution of 256×256, a batch size of 16, and 32-bit messages, with 200,000 Stage 1 steps followed by 100,000 Stage 2 steps. During Stage 1, as the bit accuracy reaches $0.75/0.90/0.93/0.95$, we enable JND modulation, reduce the minimum mask area from 5% to 2%, activate the distortion stack, and strengthen the MSE and chroma fidelity terms, respectively. Both stages use AdamW with a learning rate of 1.2×10^{-3}, 2,000 warm-up updates, and cosine decay, and the optimization schedule is restarted at the beginning of Stage 2. Both networks remain trainable throughout training. Training distortions and loss coefficients are listed in Table 5. For object-level decoding, we use $C = 32$ and $n = 3$. We also conduct ablation studies on losses in Table 12. Our code is available at MoMark for reproducibility.

We evaluate the robustness of localized watermarking methods via three different attack categories: (i) geometric distortions: crop & scale, rotation, perspective, and horizontal flip, (ii) non-geometric distortions: brightness, Gaussian blur, contrast, salt & pepper noise, random block dropout, and (iii) theft variations (e.g., theft+move: pasting stolen objects onto another image at different positions, theft+area: retaining cropped subregions of the stolen objects, theft+shrink: resizing the stolen objects to smaller sizes, and occlusion: covering parts of the selected objects).

4.2 Results

Robustness under image manipulation. We evaluate WOFA ($k = 1$ only), WAM, MaskWM, and MoMark on 1,000 COCO2017 validation images per k with corresponding segmentation masks.

[1]The original WOFA model does not predict binary masks; we estimate the union mask of WOFA by computing the local RMS magnitude of its extracted noise pattern within a (15×15) window. We binarize this map using a fixed threshold calibrated on a disjoint training subset; this is our postprocessing.

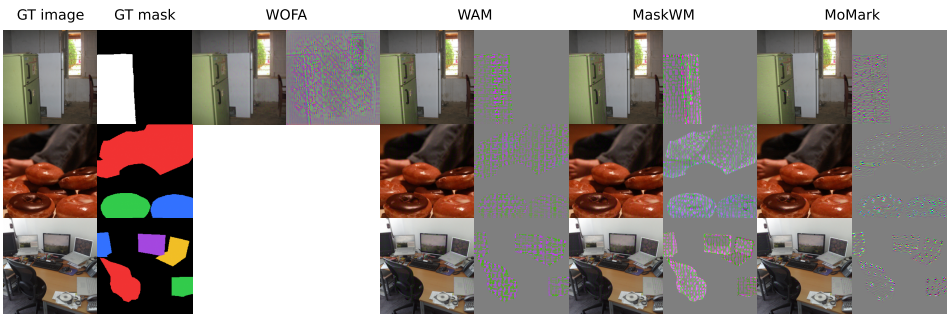

Figure 2: Examples of watermarked images and residuals ($\times 10$) with matched PSNR

Table 1 reports performance on clean watermarked images and attack-group averages for attacked images. IoU denotes union IoU for $k = 1$ and instance IoU for $k = 3$ and $k = 5$. Detailed per-attack results are provided in Appendix B.1.

For $k = 1$, MoMark achieves bit accuracies of 0.976 under geometric distortions and 0.999 under theft attacks, compared with 0.764/0.892 for WAM and 0.803/0.829 for MaskWM, respectively. Under theft attacks, MoMark also achieves a union IoU of 0.979, whereas WOFA, despite being trained for partial image theft, achieves a bit accuracy of 0.789 and a TPR of 0.705. Although MoMark achieves lower union IoU than MaskWM under geometric and non-geometric distortions at $k = 1$, it achieves higher instance IoU across all three attack categories in the multi-object settings ($k = 3, 5$). At $k = 3$, MoMark achieves bit accuracies of 0.921, 0.957, and 0.962 under geometric, non-geometric, and theft attacks, respectively, with corresponding TPRs of 1.000, 0.999, and 1.000. Its advantage under theft extends to instance localization, with an instance IoU of 0.737 versus 0.448 for WAM and 0.442 for MaskWM. At $k = 5$, MoMark retains 0.911 bit accuracy, 1.000 TPR, and 0.593 instance IoU under theft, compared with 0.659/0.868/0.301 for WAM and 0.647/0.701/0.315 for MaskWM. Together with its bit accuracies of 0.868 and 0.911 under geometric and non-geometric distortions at $k = 5$, these results demonstrate generalization beyond the maximum of three watermarked objects per image used during training.

To evaluate watermark extraction from small surviving regions of stolen objects, we additionally conduct a retention sweep, retaining 75% to 10% of each object's bounding-box area and intersecting the retained region with its segmentation mask. Due to space limitations, the results are provided in Table 8 in the appendix. At 10% retention with $k = 5$, MoMark maintains a bit accuracy of 0.846 and returns 4.843 instances on average, close to the expected count of five. In contrast, WAM and MaskWM return only 0.718 and 3.154 instances, respectively. This underestimation of the instance count contributes to their lower instance IoUs of 0.124 and 0.403, compared with 0.712 for MoMark, highlighting its stronger recovery of individual watermarked objects when only small portions of the objects remain.

We additionally evaluate computational efficiency on clean watermarked images, reporting embedding, extraction, and object-level decoding latency together with the number of returned instances in Figure 5.

Instance-specific partial theft out of k objects. We evaluate a complementary scenario in which only one object in a multi-object composition is stolen. Table 2 summarizes performance over four attack categories: theft+move, theft+area 0.5, theft+shrink 0.5, and occlusion 0.5, where 0.5 specifies the retention or spatial scaling ratio for the corresponding attack. Detailed results, including additional ratios, are provided in Table 9. Across both $k = 3$ and $k = 5$, MoMark consistently achieves the highest bit accuracy under theft+move, area reduction, and shrinking. In particular, under theft+move, MoMark achieves bit accuracies of 0.999 and 0.996 at $k = 3$ and $k = 5$, respectively, with a TPR of 1.000 in both settings and instance IoUs of 0.946 and 0.913. Shrinking is substantially more challenging for all methods. At $k = 5$ under theft+shrink 0.75, MoMark achieves 0.959 bit accuracy and 0.817 instance IoU, compared with 0.519/0.186 for WAM and 0.646/0.490 for MaskWM. Under occlusion, MoMark maintains an image-level TPR of 1.000 at both object counts, with bit accuracies of 0.926/0.967 at $k = 3$ and 0.899/0.926 at $k = 5$ for occlusion 0.75/0.5, respectively. These results demonstrate that MoMark achieves stronger instance-specific message recovery and localization than the competing methods under partial-theft attacks.

Table 1: Attack-group averages on 1000 images per k with PSNR matched to 39 dB. IoU denotes union IoU for k=1 and instance IoU for k=3 and k=5. The detailed results are provided in Table 6

Method	Bit accuracy				TPR				IoU			
	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft
WOFA	0.902	0.658	0.812	0.789	0.980	0.366	0.829	0.705	0.680	0.446	0.361	0.636
WAM	0.999	0.764	0.987	0.892	**1.000**	0.553	0.997	0.808	0.962	0.872	0.889	0.896
MaskWM	**1.000**	0.803	0.989	0.829	**1.000**	0.656	0.998	0.657	0.979	**0.955**	**0.922**	0.947
MoMark	**1.000**	**0.976**	**0.999**	**0.999**	**1.000**	**0.999**	**1.000**	**1.000**	**0.986**	0.941	0.913	**0.979**

(a) $k = 1$

Method	Bit accuracy				TPR				IoU			
	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft
WAM	0.867	0.662	0.807	0.728	0.980	0.888	0.882	0.892	0.676	0.365	0.554	0.448
MaskWM	0.794	0.608	0.776	0.698	**1.000**	0.689	0.993	0.730	0.531	0.371	0.480	0.442
MoMark	**0.977**	**0.921**	**0.957**	**0.962**	**1.000**	**1.000**	**0.999**	**1.000**	**0.775**	**0.641**	**0.667**	**0.737**

(b) $k = 3$

Method	Bit accuracy				TPR				IoU			
	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft	Clean	Geo.	Non-geo.	Theft
WAM	0.775	0.621	0.724	0.659	0.967	0.868	0.849	0.868	0.498	0.246	0.400	0.301
MaskWM	0.719	0.571	0.706	0.647	0.996	0.584	0.988	0.701	0.397	0.217	0.364	0.315
MoMark	**0.929**	**0.868**	**0.911**	**0.911**	**1.000**	**1.000**	**0.999**	**1.000**	**0.621**	**0.491**	**0.536**	**0.593**

(c) $k = 5$

Table 2: Instance-specific attack results for $k = 3$ and $k = 5$; theft+move, theft+area, theft+shrink, and occlusion

		WAM			MaskWM			MoMark		
k	attack type	bit	TPR	inst.	bit	TPR	inst.	bit	TPR	inst.
3	Clean	0.867	0.980	0.676	0.794	**1.000**	0.531	**0.977**	**1.000**	**0.775**
	Attacked	0.718	0.632	0.477	0.716	0.597	0.582	**0.899**	**0.808**	**0.793**
5	Clean	0.775	0.967	0.498	0.719	0.996	0.397	**0.929**	**1.000**	**0.621**
	Attacked	0.671	0.550	0.375	0.678	0.549	0.495	**0.878**	**0.783**	**0.731**

Qualitative results. Figure 3(a) presents qualitative results for $k = 1, 3, 5$ under clean inputs and one representative attack from each category: crop-and-scale for geometric distortions, contrast adjustment for non-geometric distortions, and theft-and-shrink for theft attacks. Predicted regions are matched one-to-one to GT objects based on the similarity between their decoded messages and the GT messages. MaskWM exhibits missing regions or incomplete mask predictions under crop-and-scale, suggesting difficulty in maintaining accurate localization when spatial correspondence is disrupted. Another notable limitation is incorrect instance attribution despite successful region detection. In the theft-and-shrink examples, inaccurately decoded messages cause detected regions to be matched to the wrong GT instances. WAM also exhibits incomplete masks and misses instances in the multi-object setting, with these limitations becoming more apparent as the number of objects increases. Additional qualitative results for union and instance masks across all attack categories are provided in Appendix B.3.

Figure 3(b) shows instance-specific theft, where a selected object is composited onto another image after translation, partial retention, or shrinking. Gray regions indicate predictions assigned to a message identity other than that of the selected GT object. WAM exhibits incomplete masks and poorly delineated boundaries in several examples, particularly under shrinking. As in (a), MaskWM sometimes produces incorrect instance assignments: even when the stolen region is relatively large and accurately localized, errors in message recovery lead to incorrect attribution. In contrast, MoMark more closely preserves the GT mask boundaries, including fine structures, while maintaining the correct message assignment in the illustrated examples. Additional qualitative results are provided in Appendix B.3.

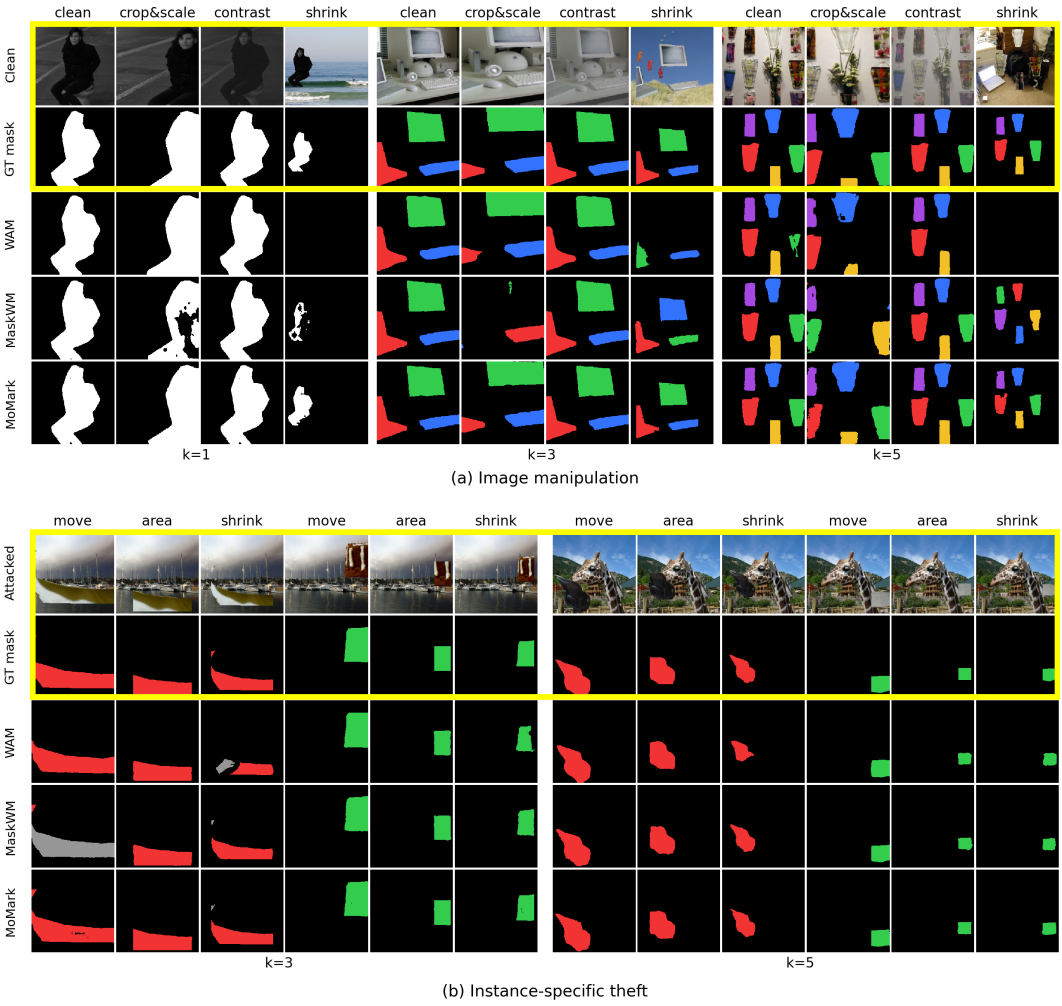

Figure 3: Qualitative comparison of watermark localization. (a) Localization under clean inputs and representative examples of each of the three attack categories for $k = 1, 3, 5$. (b) Instance-specific localization when a selected watermarked object undergoes partial-theft transformations. Colors indicate message identities after one-to-one message-based matching to GT objects. In (b), gray denotes predicted regions assigned to an identity other than that of the selected GT object

Table 3: Object-level inpainting with $k = 5$ objects. Messages are embedded before editing; we inpaint $n \in \{1, 2, 3, 4\}$ objects and report extraction results for the remaining $k - n$ objects

Method	1 object edited				2 objects edited				3 objects edited				4 objects edited			
	bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.
WAM	0.550	**0.442**	0.267	0.092	0.534	0.271	0.206	0.064	0.522	0.147	0.160	0.044	0.514	0.051	0.115	0.026
MaskWM-base	0.538	0.289	0.091	0.055	0.526	0.161	0.063	0.040	0.520	0.096	0.044	0.030	0.516	0.045	0.033	0.025
MoMark-base	0.513	0.057	0.016	0.006	0.512	0.043	0.017	0.006	0.511	0.022	0.017	0.006	0.508	0.014	0.015	0.005
MaskWM-FT	0.585	0.290	**0.447**	0.184	0.577	0.165	**0.361**	0.175	0.573	0.084	0.280	0.164	0.567	0.045	0.178	0.147
MoMark-FT	**0.659**	0.363	0.408	**0.246**	**0.659**	**0.285**	0.348	**0.241**	**0.661**	**0.176**	**0.295**	**0.244**	**0.661**	**0.087**	**0.233**	**0.235**

Advanced attacks. Table 3 presents watermark extraction performance under object-level inpainting using Stable Diffusion2-Inpainting (Rombach et al., 2022). Following MaskWM's use of VAE-based fine-tuning, we fine-tune MoMark for 200k steps by adding VAE reconstruction to the training distortions and compare it with the VAE-fine-tuned MaskWM checkpoint (`ED_32bits_vae_ft.pth`). To evaluate extraction from surviving objects in a multi-object inpainting setting, we start with $k = 5$ independently watermarked objects, jointly inpaint n selected objects, and evaluate message extraction and localization for the remaining $5 - n$ objects. Across all inpainting settings, MoMark-FT achieves higher bit accuracy and instance IoU than WAM and MaskWM-FT. When one object is edited, MoMark-FT achieves a bit accuracy of 0.659 and an in-

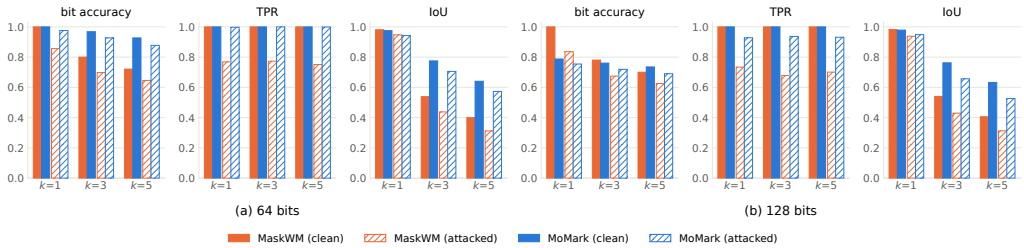

Figure 4: Multi-object watermarking with 64- and 128-bit messages

stance IoU of 0.246, compared with 0.585/0.184 for MaskWM-FT and 0.550/0.092 for WAM. This advantage persists when four objects are edited, with MoMark-FT achieving 0.661 bit accuracy and 0.235 instance IoU for the remaining object, compared with 0.567/0.147 for MaskWM-FT. Additional results for VAE reconstruction, regeneration, and outpainting are provided in Appendix B.1.

Longer messages. We further examine whether MoMark scales beyond the default 32-bit message by comparing it with MaskWM at 64 and 128 bits. Our longer-message models retain the architectural design used for 32-bit training, whereas MaskWM uses message-projection layers of size $b \to b^2$ in the encoder and $b^2 \to b$ in the decoder for a b-bit message. WAM is excluded because its released checkpoint supports only 32-bit messages. Figure 4 summarizes clean performance and the equally weighted average over all attack categories for $k \in \{1, 3, 5\}$, with detailed results provided in Table 10.

At $k = 1$, MoMark achieves higher attack-average bit accuracy at 64 bits, whereas MaskWM achieves higher bit accuracy on clean, non-geometric, and theft inputs at 128 bits. We focus below on the multi-object settings, which are the primary target of our framework. At 64 bits, MoMark achieves higher bit accuracy and instance IoU than MaskWM on both clean inputs and the attack average at $k = 3$ and $k = 5$. Its attack-average bit accuracy is 0.923/0.874 at $k = 3/5$, compared with 0.691/0.640 for MaskWM, while attack-average instance IoU reaches 0.703/0.570 versus 0.432/0.305. At 128 bits, MoMark retains higher attack-average bit accuracy at both multi-object counts, achieving 0.719/0.690 at $k = 3/5$, compared with 0.675/0.625 for MaskWM. Clean bit accuracy is slightly lower at $k = 3$ (0.760 vs. 0.780) but higher at $k = 5$ (0.735 vs. 0.700). MoMark also achieves higher instance IoU on both clean inputs and the attack average, with attack-average instance IoU of 0.657/0.526 compared with 0.430/0.312. Here, attack averages assign equal weight to each of the 14 attack conditions. At matched embedding fidelity, these results demonstrate an advantage in attack-average message recovery and instance localization in multi-object settings across both message lengths, although message recovery becomes more challenging at 128 bits.

5 Conclusion

We presented MoMark, a localized multi-object watermarking framework that recovers independent object messages and their spatial support after image manipulation and partial theft. Two-stage training combines single-object supervision with multi-object and partial-support recovery, and object-level decoding separates watermarked instances without requiring ground-truth masks or a known object count. Experiments demonstrate improved message extraction and localization in multi-object settings, including extraction from small surviving regions and generalization from up to three objects during training to five at evaluation. These advantages extend to 64- and 128-bit messages, with MoMark outperforming MaskWM in both bit accuracy and instance IoU across both multi-object settings. These findings highlight that detecting the overall watermarked area alone is insufficient for object-level provenance: extracted messages must also be associated with the correct individual objects. By using message evidence to separate instances and training explicitly for recovery from partial support, MoMark advances localized watermarking toward object-level provenance in images assembled from independently watermarked content.

AI USE STATEMENT

We used generative AI tools to assist with grammar and clarity checks, code cleanup, and identification of inconsistencies between reported results and their descriptions. We reviewed the AI-generated content and take responsibility for the final content of this paper.

ETHICS STATEMENT

This work aims to support object-level provenance tracking and copyright protection through localized multi-object watermarking. Watermark predictions may contain false positives or false negatives and should not be treated as conclusive evidence of ownership or authenticity. Unauthorized watermark embedding may also enable misleading attribution claims. Practical deployment should respect content owners' rights and use watermark evidence alongside independent provenance information.

REPRODUCIBILITY STATEMENT

Section 3 describes the architecture, training objectives, and instance decoding procedure of our proposed method, MoMark. The experimental setup specifies the datasets, baseline configurations, evaluation metrics, and PSNR-matching procedure. Additional implementation details in the appendix cover the training curriculum, partial-support sampling, and attack details. Appendix B.2 provides decoding settings and ablation studies. The anonymous repository for reproducibility is provided at MoMark.

REFERENCES

Ali Al-Haj. Combined dwt-dct digital image watermarking. *Journal of computer science*, 3(9): 740–746, 2007.

Vishal Asnani, John Collomosse, Tu Bui, Xiaoming Liu, and Shruti Agarwal. Promark: Proactive diffusion watermarking for causal attribution. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10802–10811. IEEE, 2024.

Hai Ci, Pei Yang, Yiren Song, and Mike Zheng Shou. Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification. In *European conference on computer vision*, pp. 338–354. Springer, 2024.

Samuel Gunn, Xuandong Zhao, and Dawn Song. An undetectable watermark for generative image models. In *International Conference on Learning Representations*, volume 2025, pp. 6612–6637, 2025.

Runyi Hu, Jie Zhang, Shiqian Zhao, Nils Lukas, Jiwei Li, Qing Guo, Han Qiu, and Tianwei Zhang. Mask image watermarking. *Advances in Neural Information Processing Systems*, 38:146313–146346, 2026.

Kecen Li, Zhicong Huang, Xinwen Hou, and Cheng Hong. Gaussmarker: Robust dual-domain watermark for diffusion models. *arXiv preprint arXiv:2506.11444*, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Gaozhi Liu, Silu Cao, Zhenxing Qian, Xinpeng Zhang, Sheng Li, and Wanli Peng. Watermarking one for all: A robust watermarking scheme against partial image theft. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8225–8234. IEEE, 2025.

Shilin Lu, Zihan Zhou, Jiayou Lu, Yuanzhi Zhu, and Adams Kong. Robust watermarking using generative priors against image editing: From benchmarking to advances. In *International Conference on Learning Representations*, volume 2025, pp. 83902–83936, 2025.

Zehua Ma, Han Fang, Xi Yang, Kejiang Chen, and Weiming Zhang. Ropass: Robust watermarking for partial screen-shooting scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19332–19339, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Tom Sander, Pierre Fernandez, Alain Oliviero Durmus, Teddy Furon, and Matthijs Douze. Watermark anything with localized messages. In *International Conference on Learning Representations*, volume 2025, pp. 79569–79599, 2025.

Frank Y Shih. *Digital watermarking and steganography: fundamentals and techniques*. CRC press, 2017.

Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2114–2123. IEEE, 2020.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

Rui Xu, Mengya Hu, Deren Lei, Yaxi Li, David Lowe, Alex Gorevski, Mingyu Wang, Emily Ching, and Alex Deng. Invismark: Invisible and robust watermarking for ai-generated image provenance. In *2025 IEEE/CVF winter conference on applications of computer vision (WACV)*, pp. 909–918. IEEE, 2025.

Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12162–12171. IEEE, 2024.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *European conference on computer vision*, pp. 682–697. Springer, 2018.

A Implementation Details

A.1 Model Structures and Capacity

Table 4 summarizes model structure, training time, message capacity, and dataset; the training hardware is described below. As described in §4.1, WOFA operates at 200×200 with a 30-bit message, whereas the remaining methods use 256×256 images and 32-bit messages in their default configurations. The reported training times follow the hardware used in the corresponding works: WOFA was trained on a single RTX 3090 Ti, WAM on eight NVIDIA V100 GPUs, MaskWM on a single RTX A6000, and MoMark on a single RTX 6000 Ada. Notably, WAM reports approximately 90 hours of training using eight V100 GPUs, indicating a substantially larger aggregate training cost than the single-GPU configurations used by MaskWM and MoMark, whereas MaskWM and MoMark require approximately 15.8 and 62 hours, respectively, on a single GPU. WOFA does not report its total training time. Since these measurements were obtained on substantially different hardware configurations, they should be interpreted as reported training costs rather than as a controlled wall-clock comparison.

A.2 Experimental Details

Table 5a presents the training distortions (augmentations). We follow WAM's multi-watermark augmentation configuration for geometric distortions in training. The mixed-mask sampling probabilities are 16% irregular, 50% box, 16% full-image, and 18% segmentation masks, with mask inversion probability 0.5. These probabilities and the box-sampling parameters are unchanged from WAM's multi-watermark defaults. We extend the rotation-angle range from $[-10°, 10°]$ to $[-90°, 90°]$ to improve robustness.

Table 4: Model structure (parameters in millions), message capacity, dataset, and total training time

Method	Embedder	Extractor	Capacity (bits)	Dataset	GPU hours (h)
WOFA (Liu et al., 2025)	MLP+UNet (33.9M)	UNet+CNN/MLP (8.2M)	30	OPA (62k)	—
WAM (Sander et al., 2025)	VAE (1.1M)	ViT-B (93.3M)	32	COCO2014 (83k)	720
MaskWM (Hu et al., 2026)	UNet (31.1M)	U^2-Net+UNet (32.3M)	32, 64, 128	COCO2014 (83k)	15.8
MoMark	UNet (10.5M)	SWT+UNet (6.1M)	32, 64, 128	COCO2017 (118k)	62

Table 5: Training distortions and loss coefficients.

(a) Training distortions and parameter ranges

	Transform	Training range
Geometric	crop&scale	edge ratio 0.33–1.0
	resize	edge ratio 0.7–1.5
	rotate	$-90°$ to $+90°$
	perspective	distortion scale 0.1–0.5
	horizontal flip	—
Non-geometric	JPEG	quality 40–80
	Gaussian blur	kernel 3–17
	median filter	kernel 3–7
	brightness, contrast	factor 0.5–2.0
	hue	-0.1–0.1

(b) Loss coefficients

Coefficient	Stage 1	Stage 2
λ_ℓ	20	20
λ_{M}	3	3
λ_{neg}	1	—
λ_{mse}	$0.5 \to 300$	300
λ_{chr}	$0 \to 300$	300
λ_{psnr}	—	50

Table 5b summarizes the objectives used in the two training stages. Stage 1 establishes single-object message extraction and localization using $\mathcal{L}_\ell^s$, $\mathcal{L}_{\mathrm{M}}^s$, and the negative-image objective $\mathcal{L}_{\mathrm{neg}}$, while the fidelity terms are progressively ramped as decoding stabilizes. Stage 2 replaces the single-message objective with $\mathcal{L}_\ell^m$ and introduces $\mathcal{L}_\ell^f$ and $\mathcal{L}_{\mathrm{M}}^f$ to explicitly optimize multi-object and partial-support recovery.

We match each watermarked image to a PSNR of 39 dB between methods for fair comparison. Given an original image (x) and its watermarked counterpart (x_{wm}), both represented in $[0, 1]$, we rescale the embedding residual:

$$r = x_{\mathrm{wm}} - x, \qquad \widetilde{x}_{\mathrm{wm}} = \mathrm{clip}(x + \alpha r, 0, 1).$$

We determine the scalar ($\alpha \geq 0$) independently for each image by binary search, accounting for clipping, such that

$$\mathrm{MSE}(x, \widetilde{x}_{\mathrm{wm}}) \approx 10^{-39/10},$$

where MSE is averaged over all spatial locations and RGB channels of the entire image. We apply the same procedure to all compared methods. Attacks are applied to the PSNR-matched images, with no further PSNR adjustment afterward.

B Additional Results

B.1 Quantitative Results

Robustness to distortions and theft attacks Table 6 reports performance on 1,000 COCO images per k. WOFA uses its native resolution of 200×200 with 30-bit messages at $k = 1$, while the other methods use 256×256 images with 32-bit messages. WOFA uses the same source crops, masks, and attack severities, with transformations sampled at its native resolution. Bit accuracy and IoU use the original message evaluations, whereas TPR@1% FPR is evaluated using embeddings with shared deterministic messages (30 bits for WOFA). For MaskWM at $k > 1$, decoding uses connected components and region-wise decoding.

At $k = 1$, MoMark achieves the highest attack-group-average bit accuracy across geometric, non-geometric, and theft attacks, reaching 0.976, 0.999, and 0.999, respectively. Its advantage extends to multi-object recovery: at $k = 3$, the corresponding bit accuracies are 0.921, 0.957, and 0.962, compared with 0.662, 0.807, and 0.728 for WAM and 0.608, 0.776, and 0.698 for MaskWM. MoMark also achieves the highest attack-group-average instance IoU in all three categories at both $k = 3$ and $k = 5$. Under theft attacks at $k = 5$, it obtains 0.911 bit accuracy and 0.593 instance IoU,

compared with 0.659/0.301 for WAM and 0.647/0.315 for MaskWM, despite training with at most three objects per image.

The per-attack results also reveal differences between message recovery and localization. Under 90° rotation at $k = 3$, MoMark achieves 0.849 bit accuracy and 0.552 instance IoU, compared with 0.503/0.128 for WAM and 0.562/0.325 for MaskWM. However, MoMark does not uniformly outperform the baselines in union localization: at $k = 1$, MaskWM achieves higher average union IoU under geometric and non-geometric distortions (0.955/0.922 versus 0.941/0.913). Individual attacks also remain challenging for instance localization; under salt-and-pepper noise at $k = 5$, WAM achieves higher instance IoU than MoMark (0.489 versus 0.458). Overall, these results support stronger attack-group-average message recovery and multi-object instance localization, while highlighting that improvements in message recovery do not necessarily imply better union-mask recovery. Qualitative results are shown in Figures 7, 8, and 9.

Table 6: PSNR matched to approximately 39 dB per watermarked image before attacks. Evaluated on 1,000 COCO val2017 images per k

Attack	WOFA			WAM			MaskWM			MoMark		
	bit	TPR	union	bit	TPR	union	bit	TPR	union	bit	TPR	union
clean	0.902	0.980	0.680	0.999	**1.000**	0.962	**1.000**	**1.000**	0.979	**1.000**	**1.000**	**0.986**
Geometric distortions												
crop 0.75	0.705	0.466	0.699	0.943	0.974	0.952	0.652	0.296	0.936	**0.999**	**0.999**	**0.981**
hflip	0.526	0.017	0.353	0.998	**1.000**	0.957	0.998	**1.000**	0.978	**1.000**	**1.000**	**0.982**
rotation 90°	0.495	0.002	0.123	0.507	0.025	0.917	0.768	0.622	**0.952**	**0.911**	**1.000**	0.855
perspective 0.5	0.906	0.981	0.611	0.609	0.212	0.664	0.793	0.706	**0.956**	**0.994**	**0.998**	0.945
Average	0.658	0.366	0.446	0.764	0.553	0.872	0.803	0.656	**0.955**	**0.976**	**0.999**	0.941
Non-geometric distortions												
brightness 1.5	0.775	0.766	0.543	0.996	**0.999**	**0.943**	0.997	0.998	0.835	**0.999**	**0.999**	0.918
gaussian blur	0.810	0.883	0.336	0.998	**1.000**	0.940	**1.000**	**1.000**	**0.974**	**1.000**	**1.000**	0.950
contrast 0.5	0.934	0.975	0.426	0.999	**1.000**	0.961	**1.000**	**1.000**	0.976	**1.000**	**1.000**	**0.985**
salt & pepper 0.05	0.757	0.708	0.131	0.998	**1.000**	0.959	**1.000**	**1.000**	**0.969**	0.999	**1.000**	0.818
block dropout 0.8	0.785	0.812	0.367	0.944	0.986	0.643	0.949	0.994	0.857	**0.999**	**0.999**	**0.896**
Average	0.812	0.829	0.361	0.987	0.997	0.889	0.989	0.998	**0.922**	**0.999**	**1.000**	0.913
Theft attacks												
theft	0.818	0.819	0.662	0.999	**1.000**	0.965	**1.000**	**1.000**	0.975	**1.000**	**1.000**	**0.992**
theft+move +crop 0.75	0.648	0.255	0.666	0.916	0.968	0.953	0.554	0.081	0.933	**1.000**	**1.000**	**0.980**
occlusion 0.5	0.891	0.959	0.567	0.988	**1.000**	0.838	0.993	**1.000**	0.939	**1.000**	**1.000**	**0.960**
theft+area 0.5	0.781	0.726	0.680	0.992	**1.000**	0.959	0.999	**1.000**	0.962	**1.000**	**1.000**	**0.989**
theft+shrink 0.75	0.808	0.766	0.604	0.565	0.071	0.765	0.597	0.206	0.925	**0.994**	**1.000**	**0.975**
Average	0.789	0.705	0.636	0.892	0.808	0.896	0.829	0.657	0.947	**0.999**	**1.000**	**0.979**

(a) $k = 1$

Attack	WAM				MaskWM				MoMark			
	bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.
clean	0.867	0.980	0.912	0.676	0.794	**1.000**	**0.942**	0.531	**0.977**	**1.000**	0.941	**0.775**
Geometric distortions												
crop 0.75	0.749	0.964	0.899	0.526	0.554	0.499	0.839	0.268	**0.960**	**1.000**	**0.931**	**0.693**
hflip	0.858	0.983	0.902	0.668	0.733	0.997	**0.940**	0.523	**0.967**	**1.000**	0.916	**0.722**
rotation 90°	0.503	0.935	0.791	0.128	0.562	0.577	**0.882**	0.325	**0.849**	**1.000**	0.720	**0.552**
perspective 0.5	0.539	0.671	0.558	0.137	0.583	0.681	**0.898**	0.367	**0.909**	**0.999**	0.829	**0.598**
Average	0.662	0.888	0.787	0.365	0.608	0.689	**0.890**	0.371	**0.921**	**1.000**	0.849	**0.641**
Non-geometric distortions												
brightness 1.5	0.852	0.982	**0.888**	0.649	0.801	**1.000**	0.809	0.488	**0.963**	**1.000**	0.857	**0.696**
gaussian blur	0.868	0.988	0.883	0.667	0.792	**1.000**	**0.928**	0.523	**0.969**	**1.000**	0.867	**0.701**
contrast 0.5	0.866	0.975	0.913	0.676	0.793	**1.000**	0.937	0.528	**0.976**	**1.000**	**0.938**	**0.773**
salt & pepper 0.05	0.863	0.980	0.908	**0.668**	0.793	**1.000**	**0.923**	0.522	**0.962**	**1.000**	0.710	0.583
block dropout 0.8	0.587	0.486	0.556	0.110	0.703	0.966	0.796	0.339	**0.917**	**0.997**	**0.809**	**0.584**
Average	0.807	0.882	0.830	0.554	0.776	0.993	**0.879**	0.480	**0.957**	**0.999**	0.836	**0.667**
Theft attacks												
theft	0.864	0.979	0.917	0.675	0.796	**1.000**	0.936	0.534	**0.975**	**1.000**	**0.965**	**0.792**
theft+move +crop 0.75	0.724	0.959	0.896	0.503	0.524	0.280	0.817	0.211	**0.962**	**1.000**	**0.927**	**0.704**
occlusion 0.5	0.726	0.890	0.688	0.349	0.788	0.997	0.848	0.510	**0.956**	**1.000**	**0.852**	**0.668**
theft+area 0.5	0.806	0.940	0.901	0.601	0.843	**1.000**	0.913	0.709	**0.980**	**1.000**	**0.958**	**0.842**
theft+shrink 0.75	0.519	0.694	0.602	0.113	0.537	0.374	0.781	0.244	**0.935**	**1.000**	**0.893**	**0.679**
Average	0.728	0.892	0.801	0.448	0.698	0.730	0.859	0.442	**0.962**	**1.000**	**0.919**	**0.737**

(b) $k = 3$

Table 6: continued

Attack	WAM				MaskWM				MoMark			
	bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.
clean	0.775	0.967	0.892	0.498	0.719	0.996	**0.922**	0.397	**0.929**	**1.000**	0.910	**0.621**
Geometric distortions												
crop 0.75	0.689	0.951	0.874	0.370	0.538	0.419	0.793	0.143	**0.920**	**1.000**	**0.896**	**0.545**
hflip	0.769	0.971	0.882	0.491	0.662	0.979	**0.919**	0.373	**0.919**	**1.000**	0.856	**0.559**
rotation 90°	0.503	0.907	0.717	0.054	0.533	0.405	**0.847**	0.154	**0.791**	**0.998**	0.649	**0.415**
perspective 0.5	0.523	0.643	0.490	0.070	0.551	0.533	**0.866**	0.199	**0.841**	**1.000**	0.742	**0.444**
Average	0.621	0.868	0.741	0.246	0.571	0.584	**0.856**	0.217	**0.868**	**1.000**	0.786	**0.491**
Non-geometric distortions												
brightness 1.5	0.760	0.968	**0.864**	0.469	0.721	0.996	0.778	0.366	**0.919**	**1.000**	0.815	**0.560**
gaussian blur	0.776	0.977	0.865	0.497	0.713	0.997	**0.903**	0.381	**0.925**	**1.000**	0.814	**0.556**
contrast 0.5	0.773	0.964	0.891	0.494	0.715	0.995	**0.913**	0.387	**0.929**	**1.000**	0.904	**0.617**
salt & pepper 0.05	0.769	0.965	0.886	**0.489**	0.714	0.994	**0.897**	0.382	**0.916**	**1.000**	0.673	0.458
block dropout 0.8	0.542	0.372	0.535	0.051	0.668	0.958	**0.775**	0.303	**0.868**	**0.997**	0.773	**0.489**
Average	0.724	0.849	0.808	0.400	0.706	0.988	**0.853**	0.364	**0.911**	**0.999**	0.796	**0.536**
Theft attacks												
theft	0.772	0.967	0.898	0.497	0.715	0.998	0.913	0.388	**0.934**	**1.000**	**0.943**	**0.649**
theft+move +crop 0.75	0.664	0.961	0.873	0.339	0.513	0.220	0.757	0.098	**0.921**	**1.000**	**0.882**	**0.559**
occlusion 0.5	0.631	0.855	0.656	0.202	0.716	0.987	**0.800**	0.392	**0.903**	**1.000**	0.791	**0.521**
theft+area 0.5	0.715	0.921	0.874	0.420	0.768	0.999	0.878	0.582	**0.940**	**1.000**	**0.936**	**0.726**
theft+shrink 0.75	0.511	0.637	0.520	0.046	0.522	0.301	0.687	0.113	**0.858**	**0.999**	**0.808**	**0.509**
Average	0.659	0.868	0.764	0.301	0.647	0.701	0.807	0.315	**0.911**	**1.000**	**0.872**	**0.593**

(c) $k = 5$

Table 7: Generative attacks at matched 39 dB on 1,000 paired images per object-count cohort k

Method	k	VAE								Regeneration				SD2 outpainting			
		SD1.4				MSE											
		bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.	bit	TPR	union	inst.
WAM	1	0.518	0.052	0.030		0.531	0.078	0.042		0.697	0.507	0.364		0.780	0.683	0.444	
	3	0.503	0.057	0.019	0.006	0.506	0.075	0.026	0.009	0.570	**0.534**	0.288	0.135	0.589	**0.599**	0.275	0.149
	5	0.501	0.041	0.011	0.002	0.502	0.054	0.016	0.003	0.538	**0.476**	0.211	0.069	0.547	**0.510**	0.190	0.076
MaskWM-base	1	0.503	0.005	0.004		0.511	0.020	0.013		0.694	0.430	0.292		0.703	0.443	0.220	
	3	0.500	0.010	0.001	0.001	0.502	0.029	0.004	0.002	0.568	0.424	0.180	0.112	0.558	0.382	0.112	0.072
	5	0.500	0.008	0.001	0.000	0.501	0.005	0.001	0.000	0.536	0.312	0.120	0.056	0.526	0.270	0.064	0.031
MoMark-base	1	0.500	0.013	0.009		0.503	0.012	0.001		0.573	0.104	0.046		0.562	0.081	0.036	
	3	0.501	0.023	0.009	0.001	0.500	0.002	0.001	0.000	0.526	0.092	0.029	0.018	0.510	0.039	0.014	0.006
	5	0.500	0.039	0.008	0.001	0.500	0.000	0.001	0.000	0.514	0.082	0.021	0.008	0.506	0.044	0.008	0.002
MaskWM-FT	1	0.631	0.314	0.341		0.649	**0.360**	0.365		**0.819**	**0.746**	**0.587**		**0.876**	**0.880**	**0.709**	
	3	0.529	0.078	0.189	0.067	0.535	0.093	0.212	0.072	0.620	0.413	0.463	0.243	0.618	0.445	0.529	0.234
	5	0.515	0.059	0.145	0.031	0.517	0.067	0.164	0.035	0.570	0.319	0.399	0.137	0.566	0.365	0.452	0.125
MoMark-FT	1	**0.700**	**0.342**	**0.572**		**0.684**	0.271	**0.552**		0.708	0.406	0.528		0.731	0.499	0.675	
	3	**0.625**	**0.384**	**0.516**	**0.210**	**0.626**	**0.357**	**0.507**	**0.218**	**0.664**	0.532	**0.503**	**0.286**	**0.663**	0.541	**0.562**	**0.290**
	5	**0.594**	**0.186**	**0.478**	**0.147**	**0.594**	**0.150**	**0.474**	**0.151**	**0.635**	0.363	**0.494**	**0.216**	**0.630**	0.356	**0.531**	**0.212**

Advanced generative attacks Table 7 reports robustness to VAE reconstruction and generative manipulations. MaskWM-base and MaskWM-FT use the base ED and VAE-fine-tuned ED checkpoints, respectively. Following SD1.4 VAE fine-tuning, MoMark-FT achieves the highest bit accuracy and instance IoU among the compared methods across all four attack settings at $k = 3$ and $k = 5$.

Under SD1.4 VAE reconstruction at $k = 3$, MoMark-FT achieves a bit accuracy of 0.625 and an instance IoU of 0.210, compared with 0.529/0.067 for MaskWM-FT and 0.503/0.006 for WAM. Its advantage also extends to the MSE VAE setting, where it achieves 0.626 bit accuracy and 0.218 instance IoU, compared with 0.535/0.072 for MaskWM-FT.

The gains persist under regeneration and outpainting as the number of objects increases. At $k = 5$, MoMark-FT achieves bit accuracies of 0.635 and 0.630 under regeneration and SD2 outpainting, respectively, compared with 0.570 and 0.566 for MaskWM-FT. The corresponding instance IoUs are 0.216 and 0.212, versus 0.137 and 0.125 for MaskWM-FT. These results demonstrate stronger multi-object message recovery and instance localization, although MoMark-FT does not consistently achieve the highest image-level TPR. The advantage is also specific to the multi-object settings: at $k = 1$, MaskWM-FT achieves higher bit accuracy under regeneration and outpainting.

Robustness to reduced bounding-box retention We first study a partial-theft setting in which the stolen object is further cropped before being composited onto a new host image. Specifically, we retain only a fraction of the object's bounding-box area and intersect the crop with its segmentation mask, thereby progressively reducing the spatial support of the stolen watermark while preserving the partial-theft scenario. Table 8 reports performance as this retained bounding-box fraction decreases from 0.75 to 0.1. For $k = 1$, all localized methods remain relatively robust at moderate retention ratios, while MoMark maintains the strongest message extraction as the available support becomes small. At a fraction of 0.1, MoMark achieves 0.996 bit accuracy and 0.996 TPR, compared with 0.917/0.971 for WAM, 0.917/0.971 for MaskWM, and 0.640/0.293 for WOFA. The advantage becomes more pronounced in the multi-object setting. At $k = 3$ and a fraction of 0.1, MoMark retains 0.913 bit accuracy, 0.998 TPR, and 0.822 instance IoU, compared with 0.606/0.579/0.239 for WAM and 0.709/0.923/0.603 for MaskWM. The same behavior persists at $k = 5$, where MoMark achieves 0.846 bit accuracy and 0.712 instance IoU with only 0.1 bounding-box retention. Moreover, MoMark returns 4.843 instances on average, close to the target $k = 5$, compared with 0.718 for WAM and 3.154 for MaskWM. As the retained support decreases, WAM returns progressively fewer instances, accompanied by declining instance IoU. In contrast, MoMark maintains higher bit accuracy and instance IoU and returns instance counts closer to the target, indicating stronger robustness to severe spatial truncation.

Instance-specific theft attacks out of k objects Table 9 provides detailed instance-specific results on 1,000 validation images. For theft, area retention, and shrink attacks, only the selected object's message is pasted onto an unrelated host and scored. In theft+area r, we select a rectangular region whose area is approximately r times the source object's bounding-box area, intersect it with the

Table 8: Region fractions refer to the selected bounding-box area before intersection with the object mask. IoU is union IoU at $k = 1$, instance IoU otherwise; "clusters" denotes the mean number of returned instance candidates. Blue values round to the corresponding ground-truth count k

Bbox fraction	Method	Bit accuracy↑			TPR↑			IoU↑			Clusters	
	k	1	3	5	1	3	5	1	3	5	3	5
clean	WOFA	0.902	—	—	0.980	—	—	0.680	—	—	—	—
	WAM	0.999	0.867	0.775	**1.000**	0.980	0.967	0.962	0.676	0.498	2.408	3.126
	MaskWM	**1.000**	0.794	0.719	**1.000**	**1.000**	0.996	0.979	0.531	0.397	2.261	3.100
	MoMark	**1.000**	**0.977**	**0.929**	**1.000**	**1.000**	**1.000**	**0.986**	**0.775**	**0.621**	2.958	4.623
0.75	WOFA	0.811	—	—	0.804	—	—	0.703	—	—	—	—
	WAM	0.996	0.845	0.754	**1.000**	0.962	0.951	0.962	0.652	0.472	2.278	2.916
	MaskWM	**1.000**	0.817	0.743	**1.000**	**1.000**	0.997	0.968	0.600	0.471	2.459	3.502
	MoMark	**1.000**	**0.978**	**0.939**	**1.000**	**1.000**	**1.000**	**0.990**	**0.809**	**0.675**	2.993	4.719
0.5	WOFA	0.781	—	—	0.726	—	—	0.680	—	—	—	—
	WAM	0.992	0.806	0.715	**1.000**	0.940	0.921	0.959	0.601	0.420	2.046	2.495
	MaskWM	0.999	0.843	0.768	**1.000**	**1.000**	0.999	0.962	0.709	0.582	2.736	4.062
	MoMark	**1.000**	**0.980**	**0.940**	**1.000**	**1.000**	**1.000**	**0.989**	**0.842**	**0.726**	3.010	4.793
0.25	WOFA	0.724	—	—	0.560	—	—	0.601	—	—	—	—
	WAM	0.973	0.727	0.643	0.998	0.842	0.828	0.951	0.470	0.299	1.587	1.731
	MaskWM	0.989	0.813	0.738	0.998	0.998	0.982	0.950	0.746	0.595	2.839	4.143
	MoMark	**1.000**	**0.968**	**0.924**	**1.000**	**1.000**	**1.000**	**0.983**	**0.859**	**0.765**	3.032	4.867
0.1	WOFA	0.640	—	—	0.293	—	—	0.459	—	—	—	—
	WAM	0.917	0.606	0.555	0.971	0.579	0.526	0.923	0.239	0.124	0.810	0.718
	MaskWM	0.917	0.709	0.639	0.971	0.923	0.876	0.912	0.603	0.403	2.441	3.154
	MoMark	**0.996**	**0.913**	**0.846**	**0.996**	**0.998**	**0.998**	**0.959**	**0.822**	**0.712**	3.094	4.843

Table 9: Instance-specific theft attacks, averaged over k object indices and 1000 images. Theft scores the selected message; occlusion scores all k messages.

Attack on one object	Method	Bit accuracy↑		TPR↑		Union IoU↑		Instance IoU↑	
	k	3	5	3	5	3	5	3	5
theft+move	WAM	0.732	0.695	0.775	0.642	0.846	0.788	0.650	0.537
	MaskWM	0.632	0.631	0.372	0.334	0.874	0.794	0.824	0.756
	MoMark	**0.999**	**0.996**	**1.000**	**1.000**	**0.943**	**0.906**	**0.946**	**0.913**
theft+area 0.75	WAM	0.871	0.793	0.813	0.674	0.878	0.834	0.735	0.591
	MaskWM	0.942	0.895	0.985	0.920	0.890	0.819	0.875	0.810
	MoMark	**1.000**	**0.997**	**1.000**	**1.000**	**0.942**	**0.905**	**0.945**	**0.913**
theft+area 0.5	WAM	0.824	0.742	0.723	0.564	0.862	0.807	0.657	0.501
	MaskWM	0.908	0.847	0.960	0.841	0.865	0.774	0.857	0.770
	MoMark	**0.998**	**0.990**	**0.999**	**0.995**	**0.926**	**0.879**	**0.931**	**0.890**
theft+shrink 0.75	WAM	0.531	0.519	0.453	0.293	0.635	0.548	0.281	0.186
	MaskWM	0.709	0.646	0.641	0.368	0.672	0.507	0.650	0.490
	MoMark	**0.983**	**0.959**	**0.999**	**0.990**	**0.882**	**0.805**	**0.887**	**0.817**
theft+shrink 0.5	WAM	0.500	0.500	0.061	0.035	0.267	0.213	0.036	0.018
	MaskWM	0.514	0.506	0.055	0.023	0.103	0.051	0.091	0.044
	MoMark	**0.631**	**0.599**	**0.233**	**0.136**	**0.665**	**0.581**	**0.563**	**0.518**
occlusion 0.75	WAM	0.770	0.727	0.961	0.955	0.844	0.860	0.490	0.413
	MaskWM	0.795	0.724	**1.000**	0.997	**0.918**	**0.908**	0.553	0.417
	MoMark	**0.926**	**0.899**	**1.000**	**1.000**	**0.918**	0.895	**0.679**	**0.584**
occlusion 0.5	WAM	0.816	0.746	0.971	0.958	0.863	0.866	0.565	0.442
	MaskWM	0.810	0.728	**1.000**	0.998	**0.925**	**0.910**	0.555	0.412
	MoMark	**0.967**	**0.926**	**1.000**	**1.000**	0.924	0.897	**0.732**	**0.604**

object mask, and transfer the surviving content. Thus, theft+area 0.5 refers to half of the bounding-box area before mask intersection, rather than half of the foreground pixels. In theft+shrink s, both spatial dimensions are scaled by s; for example, $s = 0.5$ halves the width and height, yielding approximately one quarter of the original area. In the instance-specific setting, theft attacks transfer only the selected object. Occlusion instead places an opaque rectangular patch within the selected object's bounding box; overlapping parts of neighboring objects are also occluded, and all GT masks are updated accordingly. For occlusion, the selected object is occluded and all k messages are scored. For theft, area-retention, and shrink attacks, only one GT instance remains; Table 9 reports union IoU and instance IoU separately. Qualitative results for instance-specific theft are shown in Figures 10, 11, and 12.

Table 10: Detailed results for Figure 4. 64/128-bit comparison on 1000 images per k with PSNR matched to 39 dB per image

k	Method	Bit accuracy				TPR				Union IoU				Instance IoU			
		Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft
1	MaskWM	**1.000**	0.734	0.991	0.819	**1.000**	0.612	**1.000**	0.663	**0.981**	**0.964**	**0.927**	0.953	—	—	—	—
	MoMark	**1.000**	**0.929**	**0.999**	**0.990**	**1.000**	**0.992**	**1.000**	**1.000**	0.975	0.949	0.909	**0.970**	—	—	—	—
3	MaskWM	0.800	0.592	0.783	0.698	**1.000**	0.580	0.998	0.702	**0.947**	**0.910**	**0.886**	0.879	0.539	0.353	0.495	0.448
	MoMark	**0.968**	**0.872**	**0.955**	**0.943**	**1.000**	**1.000**	**1.000**	**1.000**	0.935	0.867	0.848	**0.907**	**0.775**	**0.665**	**0.699**	**0.745**
5	MaskWM	0.721	0.560	0.710	0.650	**1.000**	0.515	0.998	0.692	**0.929**	**0.877**	**0.862**	0.835	0.400	0.207	0.377	0.330
	MoMark	**0.927**	**0.823**	**0.907**	**0.892**	**1.000**	**0.997**	**1.000**	**1.000**	0.915	0.816	0.821	**0.868**	**0.640**	**0.523**	**0.573**	**0.613**

(a) 64 bits.

k	Method	Bit accuracy				TPR				Union IoU				Instance IoU			
		Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft	Clean	Geo.	Non-Geo.	Theft
1	MaskWM	**1.000**	0.689	**0.983**	**0.806**	**1.000**	0.521	**1.000**	0.639	**0.982**	0.944	**0.927**	0.943	—	—	—	—
	MoMark	0.788	**0.701**	0.778	0.773	**1.000**	**0.748**	**1.000**	**1.000**	0.979	**0.958**	0.915	**0.976**	—	—	—	—
3	MaskWM	**0.780**	0.574	**0.752**	0.678	**1.000**	0.358	0.993	0.621	**0.952**	0.868	**0.879**	0.873	0.540	0.339	0.485	0.446
	MoMark	0.760	**0.665**	0.744	**0.738**	**1.000**	**0.779**	**0.999**	**1.000**	0.948	**0.906**	0.846	**0.934**	**0.762**	**0.562**	**0.662**	**0.726**
5	MaskWM	0.700	0.547	0.680	0.632	**1.000**	0.411	0.998	0.636	**0.938**	0.824	**0.855**	0.832	0.406	0.204	0.373	0.338
	MoMark	**0.735**	**0.641**	**0.714**	**0.706**	**1.000**	**0.764**	**0.999**	**0.997**	0.933	**0.873**	0.813	**0.910**	**0.633**	**0.422**	**0.540**	**0.595**

(b) 128 bits.

Extension to longer messages Table 10 provides the detailed results underlying the longer-message comparison summarized in Figure 4. WAM is excluded because its released checkpoint supports only 32-bit messages, while training longer-message variants would require substantial additional computation. We therefore compare MoMark with MaskWM using their 64- and 128-bit configurations.

At 64 bits, MoMark shows a clear advantage in multi-object message extraction. For $k = 3$, its bit accuracy is 0.872/0.955/0.943 under geometric, non-geometric, and theft attacks, compared with 0.592/0.783/0.698 for MaskWM. At $k = 5$, the corresponding results are 0.823/0.907/0.892 versus 0.560/0.710/0.650. The difference is particularly pronounced for instance localization: under theft, MoMark achieves instance IoUs of 0.745 and 0.613 at $k = 3$ and $k = 5$, respectively, compared with 0.448 and 0.330 for MaskWM. MaskWM nevertheless attains slightly higher union IoU in several settings, indicating that its main degradation lies in separating the watermarked support into individual message-bearing instances rather than detecting the overall watermarked area.

At 128 bits, the comparison is more mixed in the single-object setting. MaskWM achieves higher bit accuracy on clean, non-geometric, and theft inputs at $k = 1$, whereas MoMark achieves higher bit accuracy under geometric attacks and higher TPR and union IoU under both geometric and theft attacks. In the multi-object settings, MoMark consistently achieves higher instance IoU on clean inputs and across all three attack categories. Its bit accuracy is higher under geometric and theft attacks at both object counts, but slightly lower under non-geometric attacks at $k = 3$ (0.744 vs. 0.752). At $k = 5$, MoMark achieves higher bit accuracy across all three attack categories, including 0.714 versus 0.680 under non-geometric attacks. Under theft at $k = 5$, it achieves 0.706 bit accuracy and 0.595 instance IoU, compared with 0.632 and 0.338 for MaskWM. Overall, increasing the message length remains challenging, particularly at 128 bits, but MoMark maintains a consistent advantage in multi-object instance localization and higher attack-average bit accuracy at both object counts.

Efficiency Figure 5 compares inference latency, decomposed into embedding, extracting, and object-level decoding, on 1,000 clean watermarked images per k, together with the average number of returned instances. At $k = 3$ and $k = 5$, WAM requires 109.2 and 121.1 ms per image in total, with its DBSCAN-based decoding accounting for 85.4 and 86.2 ms, respectively. In comparison, MoMark requires $71.4/100.5$ ms in total, including $25.6/31.3$ ms for object-level decoding. MaskWM achieves lower total latency ($28.6/39.3$ ms), with object-level decoding taking $8.6/11.5$ ms. For MaskWM at $k > 1$, extracting comprises a single mask-network forward pass, while

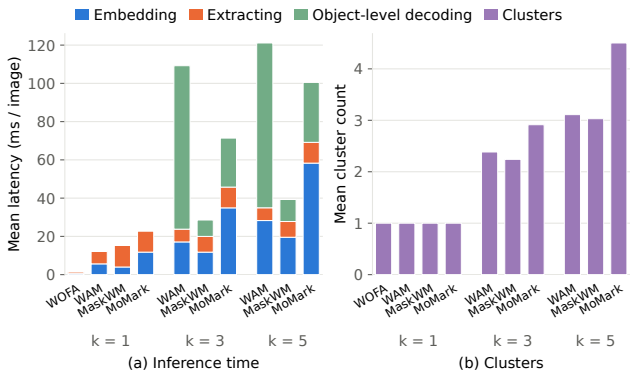

Figure 5: (a) Latency (ms) of embedding, extraction, object-level decoding of each method and (b) mean number of returned instances

object-level decoding includes connected-component discovery and message-network inference for each retained region, without redundant mask-network evaluations. The returned instance counts provide additional context. On the same timing inputs, MoMark returns $2.914/4.503$ instances on average at $k = 3/5$, compared with $2.384/3.110$ for WAM and $2.241/3.031$ for MaskWM. Thus, MoMark more closely tracks the target object counts while reducing WAM's decoding cost. These counts measure returned candidates rather than correctly recovered instances; extraction accuracy and localization quality are evaluated separately in Table 1.

Excluding embedding, MoMark requires $36.48/42.19$ ms for extraction and object-level decoding at $k = 3/5$, compared with $92.12/92.83$ ms for WAM, corresponding to speedups of $2.53\times$ and $2.20\times$, respectively. At $k = 1$, however, MoMark is slower on these stages (11.16 vs. 6.71 ms). Including embedding, its speedups over WAM at $k = 3/5$ are $1.53\times$ and $1.21\times$, respectively.

B.2 Ablation studies

Boundary reassignment. We ablate the boundary cell size C and the number of reassignment passes n introduced in Section 3, using 1,000 clean validation images per k. Our decoding initializes candidates from the 4-connected components of the predicted foreground mask, thresholded at $\sigma(a_p) > 0.5$, and filters out initial and split candidates smaller than 32 pixels. Within each component, patch-level messages are grouped by single linkage using a normalized Hamming-distance threshold of 0.25 to separate distinct messages. The resulting regions are re-pooled and merged using the same criterion, including spatially disconnected regions. The initial grouping cells remain fixed at 32×32 pixels throughout this ablation. For boundary reassignment, a $C \times C$-pixel cell is eligible if it contains at least $\max(1, \lfloor 0.25C^2 \rfloor)$ foreground pixels and its 3×3 cell neighborhood contains at least two distinct foreground instances. This corresponds to minimum foreground counts of 16, 64, and 256 for $C = 8, 16, 32$, respectively. The foreground pixels of each eligible cell are reassigned to the neighboring instance whose message best agrees with the cell's pooled bits, while background pixels remain unchanged. Region messages are re-pooled after each pass. The decoding returns at most 16 instances, ordered by area, and an empty list if no candidates remain.

All variants in Table 11 use the same trained checkpoint and differ only in their decoding configuration. To isolate the effect of decoding, we use the checkpoint's original watermarked outputs without PSNR matching, unlike the main comparisons. Without boundary reassignment, instance IoU is 0.725 at $k = 3$ and 0.588 at $k = 5$. Our default configuration, with $C = 32$ and $n = 3$, increases these values to 0.785 and 0.649, respectively, while bit accuracy changes only from $0.969/0.920$ to $0.973/0.927$. These results indicate that reassignment primarily improves spatial instance attribution, with relatively small changes in message recovery.

Localization performance varies modestly among the larger-patch configurations, whereas computational cost differs substantially. At $k = 5$, three passes with 16×16 boundary cells yield an instance IoU of 0.640 at 69.1 ms, compared with 0.649 at 31.4 ms for 32×32 boundary cells. Increasing the latter to five passes provides only a marginal gain ($0.649 \rightarrow 0.651$) while increasing decoding time to 48.3 ms. The coarse-to-fine $32 \rightarrow 16$ variant achieves the highest instance IoU of 0.658 but requires 94.7 ms, whereas 8×8 boundary cells require 208.4 ms without improving localization.

Table 11: Ablation on boundary cell size C and the number of boundary reassignment passes n at inference. The initial grouping patch size is fixed at 32×32 pixels. Each pass consists of boundary reassignment followed by region-level message re-pooling. The $32 \rightarrow 16$ configuration applies three passes at $C = 32$, followed by three passes at $C = 16$, for six passes in total. † denotes the default configuration.

		$k = 3$			$k = 5$		
C	n	bit	inst.	time (ms)	bit	inst.	time (ms)
—	—	0.969	0.725	4.0	0.920	0.588	5.2
8	3	0.971	0.728	183.6	0.926	0.593	208.4
16	3	0.973	0.767	57.7	0.930	0.640	69.1
16	5	0.973	0.769	95.3	0.930	0.644	113.7
32†	3	0.973	0.785	25.5	0.927	0.649	31.4
32	5	0.973	0.786	39.4	0.927	0.651	48.3
$32 \rightarrow 16$	3	0.973	0.781	78.6	0.929	0.658	94.7

We therefore adopt 32×32 boundary cells with three reassignment passes as a favorable trade-off between instance localization and decoding efficiency.

Loss ablation. Table 12 examines how the training objectives, SWT representation, and single-object training stage contribute to message recovery, localization, and image fidelity. All variants are evaluated at their native embedding strength without PSNR matching. Attacked results average all non-clean attack conditions equally within each object count, followed by equal averaging across $k = 1, 3, 5$; instance IoU uses $k = 3, 5$. The complete MoMark model achieves attack-average bit accuracy of 0.892 and instance IoU of 0.554, with a clean-image PSNR of 37.2 dB. Although individual variants improve selected metrics, the complete configuration provides a favorable balance between robustness and image fidelity.

Explicit multi-object supervision is important for robustness under attacks. Removing $\mathcal{L}_\ell^m$ reduces attack-average bit accuracy from 0.892 to 0.723 and instance IoU from 0.554 to 0.408, while increasing PSNR from 37.2 to 44.2 dB. Fragment supervision complements this objective: removing both $\mathcal{L}_\ell^f$ and $\mathcal{L}_M^f$ reduces attack-average bit accuracy from 0.892 to 0.881 and instance IoU from 0.554 to 0.454, despite a slightly lower PSNR of 36.8 dB. Mask supervision is particularly important for spatial recovery. Removing $\mathcal{L}_M^s$ and $\mathcal{L}_M^u$ reduces attack-average union IoU from 0.794 to 0.533 and instance IoU from 0.554 to 0.414, despite a slight increase in bit accuracy to 0.898. Additionally removing $\mathcal{L}_M^f$ leads to near-zero localization performance. These results support combining multi-object, fragment, and mask objectives to recover both messages and their spatial support.

The remaining components further support this balance. Removing the negative-image objective reduces attack-average bit accuracy from 0.892 to 0.882 and TPR from 0.924 to 0.885. Removing SWT reduces these metrics to 0.870 and 0.866, respectively. Omitting Stage 1 reduces attack-average bit accuracy to 0.830 and instance IoU to 0.518, supporting the two-stage training strategy. Removing the fidelity objective improves attack-average bit accuracy to 0.935 and instance IoU to 0.708, but reduces PSNR sharply from 37.2 to 10.9 dB. These gains therefore come at a substantial cost to image fidelity. Because embedding strength is not matched across variants, the ablations characterize each configuration's resulting robustness–fidelity trade-off rather than robustness at a fixed distortion budget. Taken together, they motivate the complete MoMark configuration as a balanced design for message recovery, instance localization, and image fidelity.

Three terms for image fidelity. The first term of $\mathcal{L}_{\text{fid}}$ is a pixel MSE between the cover x and the watermarked image x_{wm}, which directly optimizes the quantity PSNR reports.

MSE alone, however, is not enough. It treats every direction in RGB equally, so a red–green color shift and a brightness change of the same energy cost the same. The eye does not: it tolerates small brightness changes but is quick to notice color fringing along edges. An MSE penalty alone does not explicitly control chrominance distortion. We therefore add a chroma penalty to reduce visible color artifacts. Writing $\delta = x_{\text{wm}} - x$ for the residual and $C_b(\cdot), C_r(\cdot)$ for its two BT.601 chrominance projections (the color-opponent axes of the YCbCr transform, which are zero for a purely gray,

Table 12: Ablations on 1000 images per k. "Attacked" averages all non-clean attack conditions equally within each k, then averages $k = 1, 3, 5$ equally; instance IoU uses $k = 3, 5$. PSNR is reported on clean watermarked images. The $\mathcal{L}_{\text{fid}}$ removal arm is excluded from best/second-best highlighting because of its low PSNR

Setting		PSNR	bit	TPR	union	inst.
MoMark	Clean	37.2	**0.967**	**1.000**	0.959	<u>0.717</u>
	Attacked	—	<u>0.892</u>	<u>0.924</u>	<u>0.794</u>	**0.554**
$\mathcal{L}_\ell^m = 0$	Clean	**44.2**	0.956	**1.000**	**0.986**	**0.739**
	Attacked	—	0.723	0.660	**0.852**	0.408
$\mathcal{L}_\ell^f = \mathcal{L}_{\text{M}}^f = 0$	Clean	36.8	<u>0.965</u>	**1.000**	<u>0.964</u>	0.614
	Attacked	—	0.881	**0.953**	0.767	0.454
$\mathcal{L}_{\text{M}}^s = \mathcal{L}_{\text{M}}^u = 0$	Clean	37.3	**0.967**	**1.000**	0.882	0.638
	Attacked	—	**0.898**	0.878	0.533	0.414
$\mathcal{L}_{\text{M}}^s = \mathcal{L}_{\text{M}}^u = \mathcal{L}_{\text{M}}^f = 0$	Clean	37.4	0.667	<u>0.333</u>	0.004	0.000
	Attacked	—	0.640	0.284	0.003	0.000
$\mathcal{L}_{\text{neg}} = 0$	Clean	37.5	0.962	**1.000**	0.952	0.701
	Attacked	—	0.882	0.885	0.783	<u>0.539</u>
$\mathcal{L}_{\text{fid}} = 0$	Clean	10.9	0.951	1.000	0.992	0.742
	Attacked	—	0.935	0.998	0.971	0.708
Without SWT	Clean	<u>37.6</u>	0.959	**1.000**	0.940	0.679
	Attacked	—	0.870	0.866	0.775	0.524
Without stage 1	Clean	37.1	0.926	**1.000**	0.943	0.676
	Attacked	—	0.830	0.838	0.792	0.518



(a) Original image (b) Watermarked image without $\mathcal{L}_{\text{chr}}$ (c) Watermarked image of MoMark

			Stage 1		Stage 2	
MSE	Chroma	PSNR	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
✓	—	—	39.70	0.9816	35.76	0.9625
—	✓	—	15.16	0.6596	16.32	0.6740
✓	✓	—	39.74	0.9829	36.28	0.9690
✓	—	✓			38.94	0.9758
—	✓	✓			38.43	0.9726
✓	✓	✓			37.20	0.9755

Figure 6: Qualitative chroma-loss comparison (left) and clean fidelity-loss ablations (right). PSNR and SSIM are measured at native embedding strength without PSNR matching on 1000 images per $k \in \{1, 3, 5\}$ and averaged equally across k. Checkmarks specify Stage 2 losses; Stage 1 uses the indicated MSE/chroma losses with PSNR penalty disabled. Repeated Stage 1 values are omitted in the last three rows.

luminance-only residual),

$$\mathcal{L}_{\text{chr}} = \|C_b(\delta)\|^2 + \|C_r(\delta)\|^2.$$

Figure 6 separates the contributions of MSE and chroma supervision from the effect of the PSNR-floor penalty. Stage 1, where the PSNR-floor loss is disabled, provides the direct comparison of MSE and chroma losses. MSE is essential for reconstruction fidelity, whereas adding chroma supervision improves SSIM from 0.9816 to 0.9829 while maintaining comparable PSNR (39.70 versus 39.74 dB), complementing the qualitative color comparison. Stage 2 continues training for an additional 100k steps under the multi-object objectives. Here, PSNR is directly influenced by the floor penalty and its interaction with the other losses, so the final PSNR ranking should not be interpreted in isolation as a measure of the usefulness of MSE or chroma supervision. For the combined MSE–chroma configuration, adding the floor penalty improves PSNR from 36.28 to 37.20 dB and SSIM from 0.9690 to 0.9755. These results support complementary roles for reconstruction and chroma supervision in Stage 1 and for the floor penalty in mitigating fidelity degradation during Stage 2. The qualitative comparison in Figure 6 (left) further illustrates the benefit of the chroma penalty: omitting it results in more visible color artifacts in the watermarked image.

B.3 Qualitative Results

Figures 7, 8, and 9 show union and instance mask predictions for clean watermarked images and their attacked counterparts under 14 attack conditions. MaskWM accurately predicts the union mask but

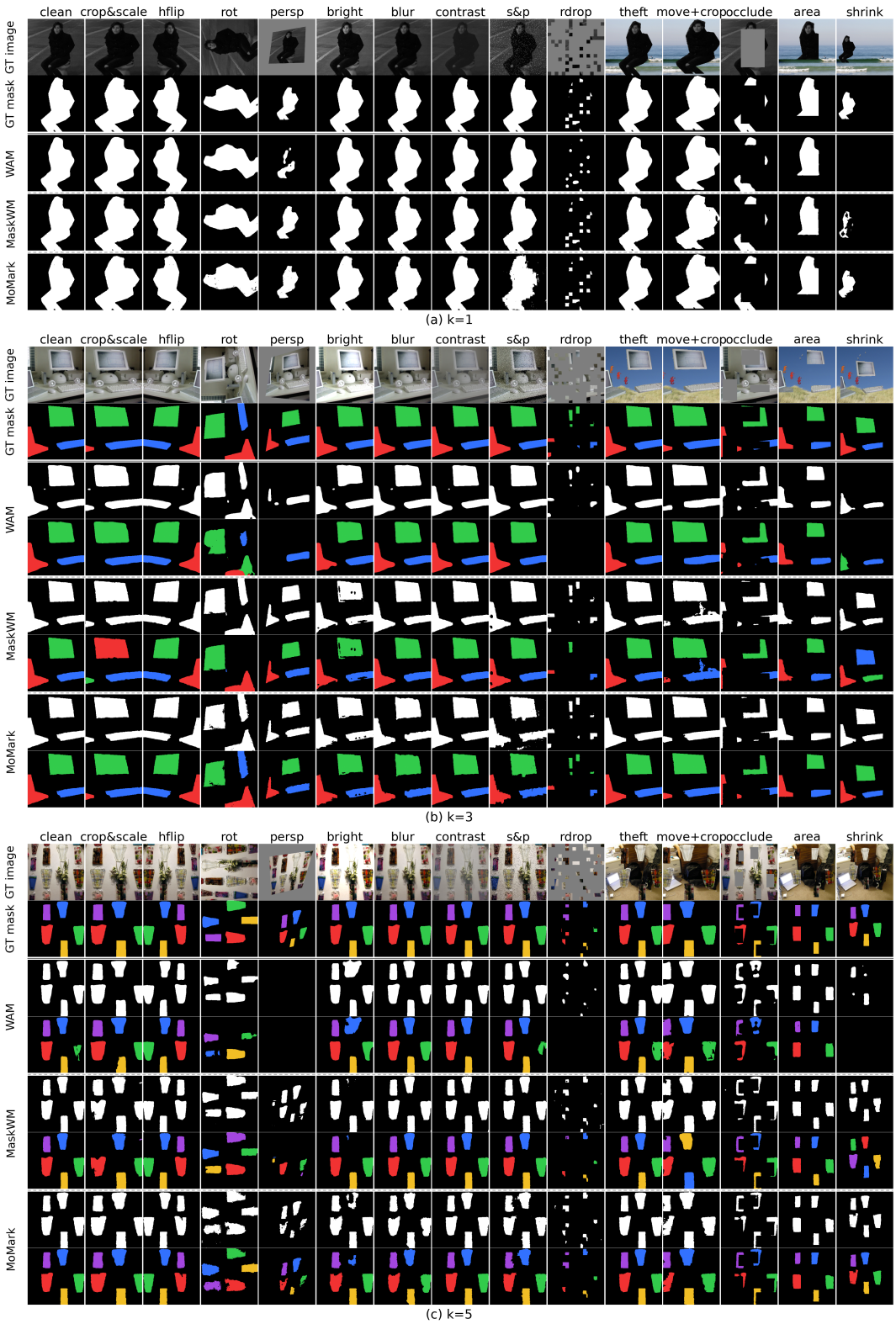

Figure 7: Qualitative results under geometric distortions, non-geometric distortions, and partial-theft attacks (1)

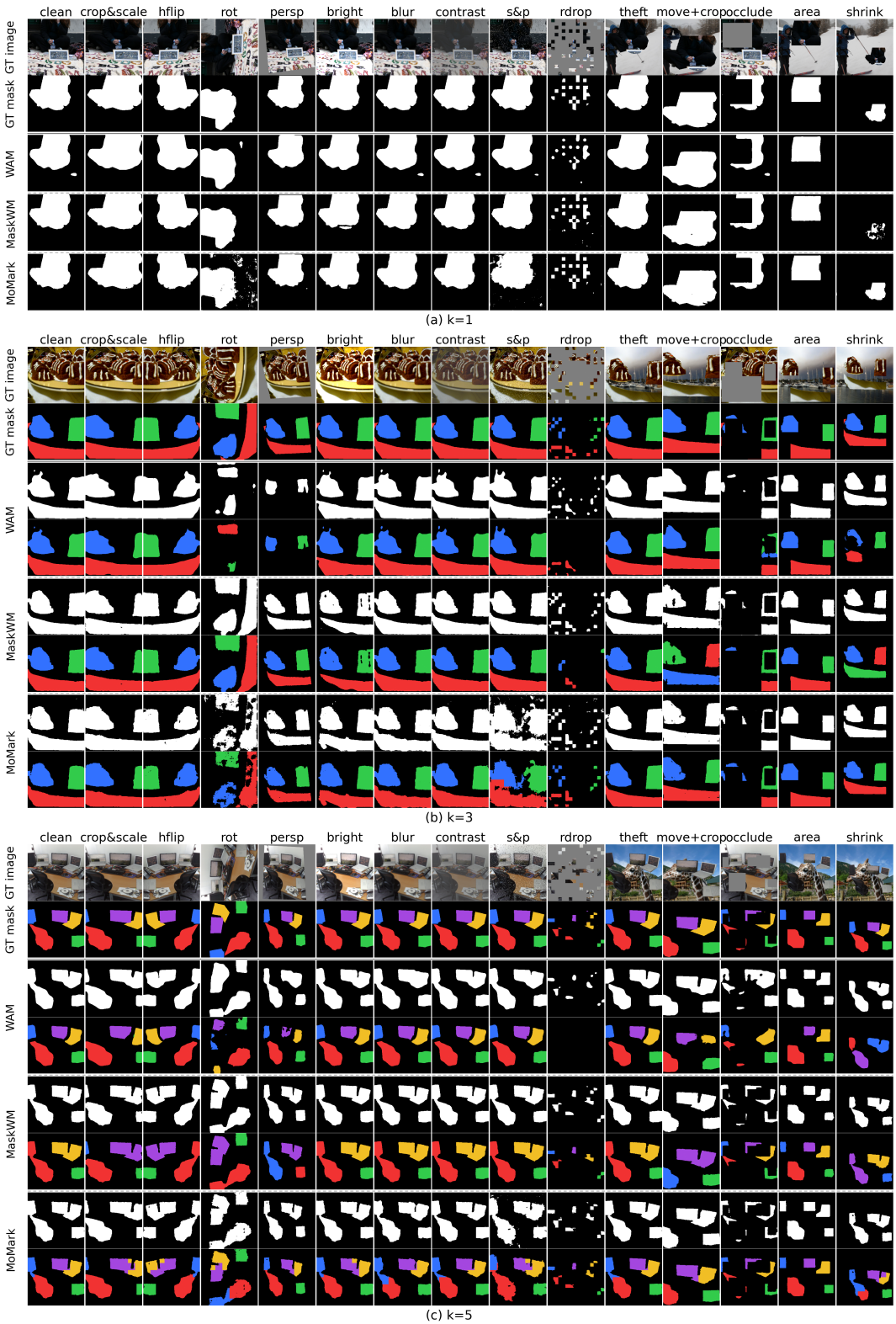

Figure 8: Qualitative results under geometric distortions, non-geometric distortions, and partial-theft attacks (2)

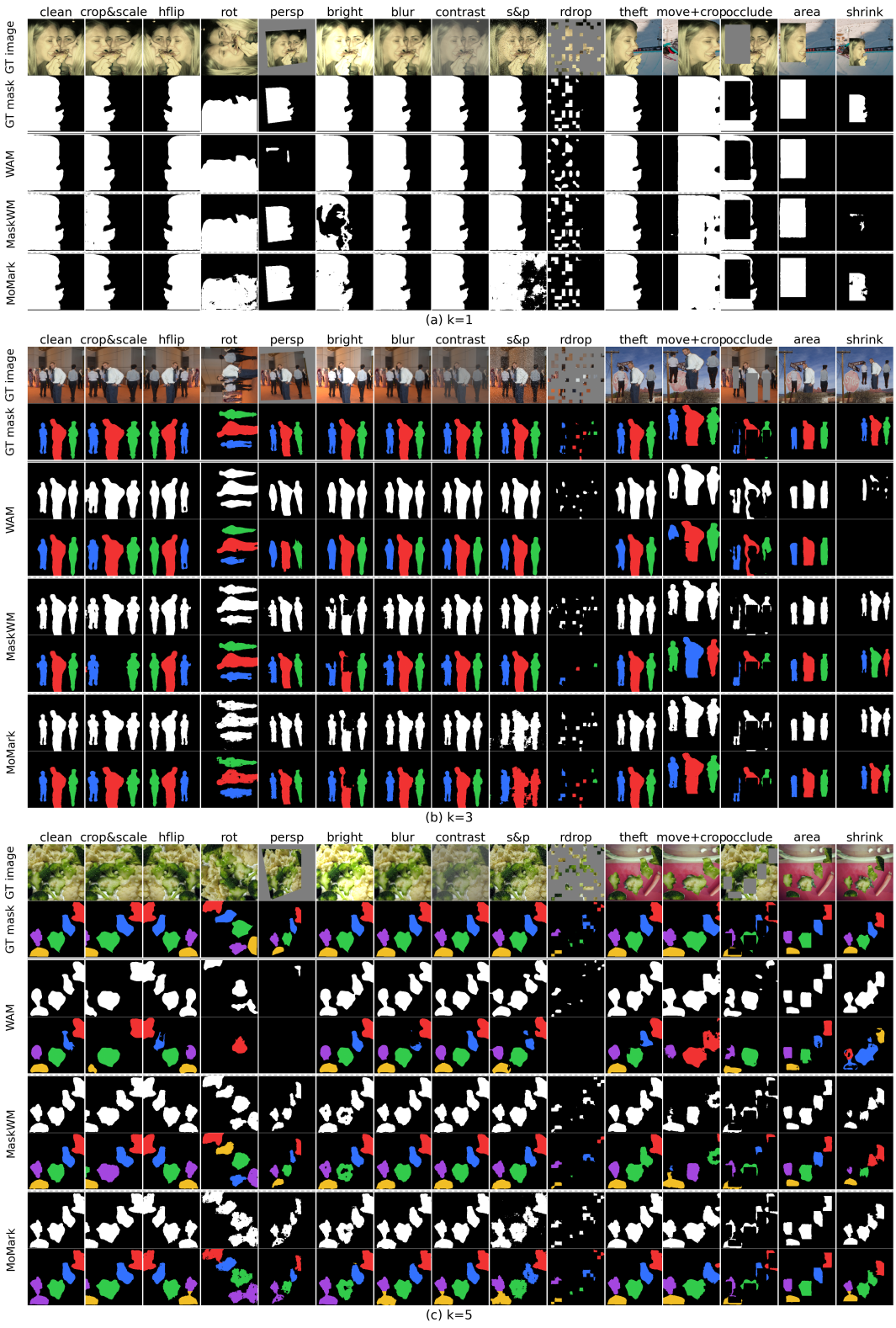

Figure 9: Qualitative results under geometric distortions, non-geometric distortions, and partial-theft attacks (3)

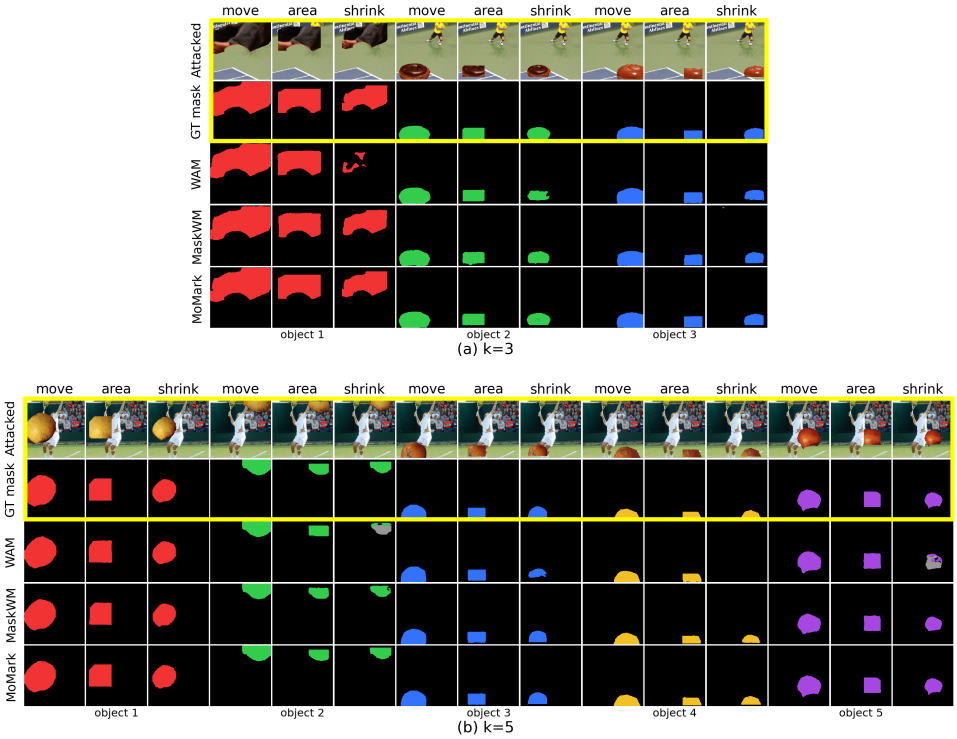

Figure 10: Qualitative results under instance-specific theft (1)

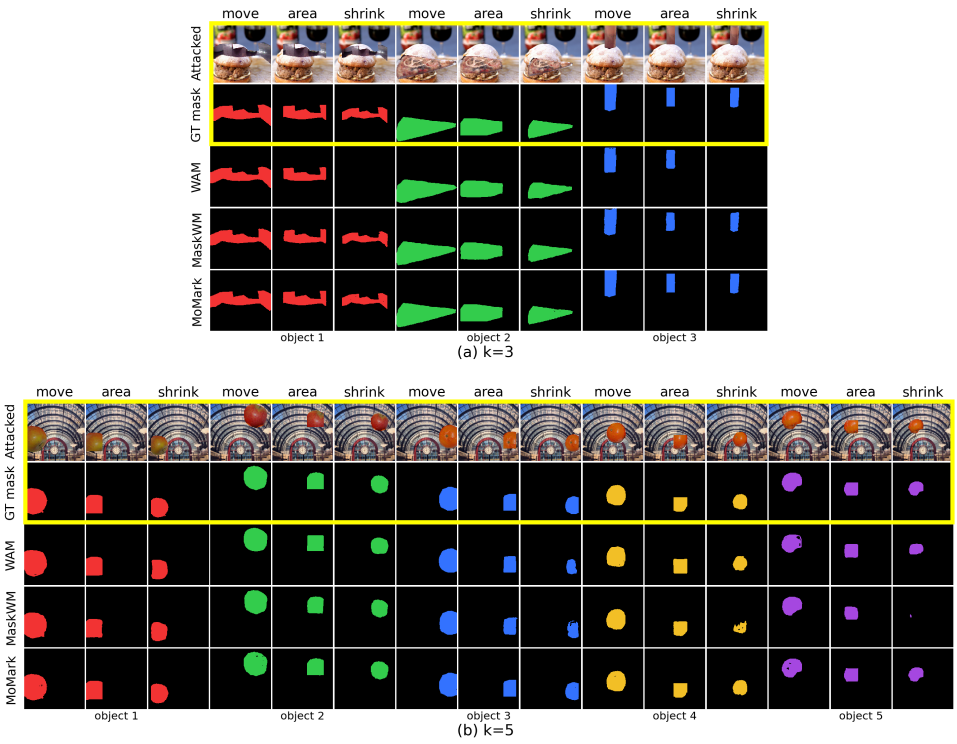

Figure 11: Qualitative results under instance-specific theft (2)

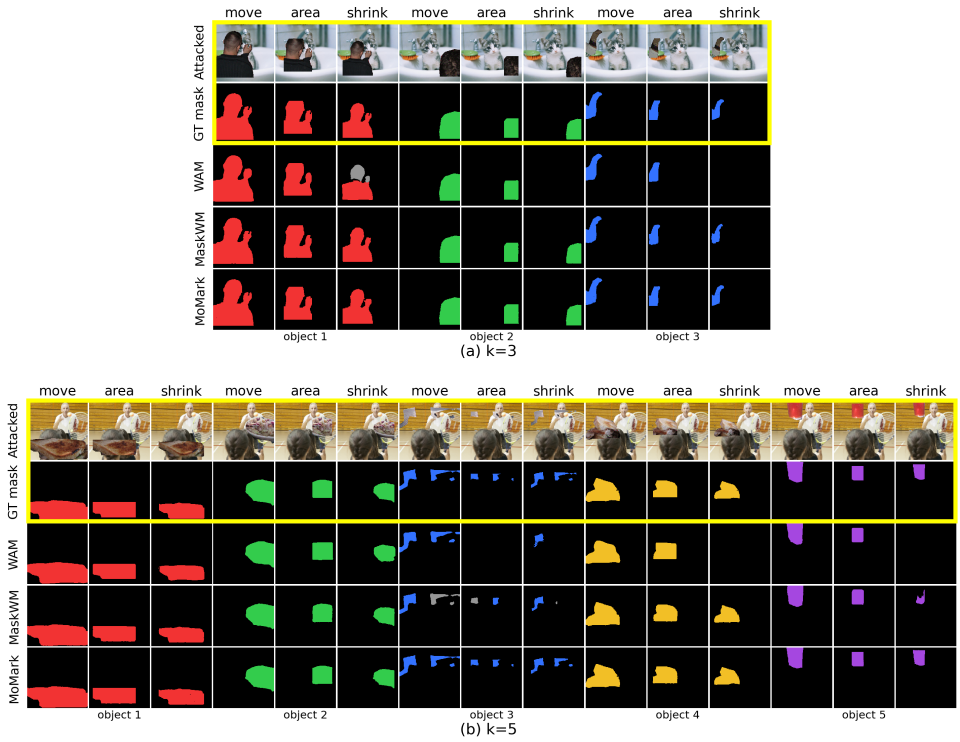

Figure 12: Qualitative results under instance-specific theft (3)

can underestimate the number of instances. For example, in the $k = 5$ example in Figure 8, where the instances are close together, MaskWM returns only three instances even for the clean watermarked image. The theft-and-shrink examples reveal further limitations: WAM misses substantial portions of the watermarked support or fails to detect it altogether, while MaskWM recovers much of the union mask but sometimes assigns the recovered regions to incorrect message identities. In the displayed examples, MoMark more consistently separates the watermarked instances and associates them with the correct messages.

Furthermore, Figures 10, 11, and 12 show mask predictions under instance-specific theft attacks at $k = 3$ and $k = 5$. Gray masks denote predicted regions assigned to an identity other than that of the selected GT object. Both WAM and MaskWM are particularly vulnerable to theft-and-shrink attacks, often failing to detect the stolen instances. They also struggle to accurately localize the surviving watermarked support when only part of the stolen region is retained, as illustrated by the "area" examples in Figure 12. Across the displayed examples, MoMark generally produces masks that more closely match the surviving watermarked regions.