Robust Image Self-Recovery against Tampering using Watermark Generation with Pixel Shuffling

Supplementary Material

A. More Details

A.1. Common Degradation

A.1.1. Training

In this section, we detail the common degradations applied during training. Specifically, we implement four typical types of degradation: Gaussian Noise, JPEG Compression, Gaussian Filter, Median Filter. The details of each degradation are as follows:

- **Gaussian Noise** A Gaussian-distributed noise with a randomly selected standard deviation $\sigma \in [1, 16]$ is added to the container image.
- **JPEG Compression** A differentiable JPEG compression is applied to the container image, with the quality factor $Q \in [75, 95]$.
- **Gaussian Filter** A Gaussian smoothing filter with a randomly selected kernel size $k = 3$ and standard deviation $\sigma = 1.0$ is applied to the container image.
- **Median Filter** A median filter with a randomly selected kernel size $k = 3$ is applied to the container image to simulate nonlinear smoothing effects.

A.1.2. Evaluation

In this section, we detail the common degradations applied during evaluation. Specifically, we implement eight typical types of degradation: Gaussian Noise, JPEG Compression, Poisson Noise, Gaussian Filter, Median Filter, Hue Adjustment, Brightness Adjustment and Contrast Adjustment. The details of each degradation are as follows:

- **Gaussian Noise** A Gaussian-distributed noise with a randomly selected standard deviation $\sigma \in [1, 9]$ is added to the container image.
- **JPEG Compression** A differentiable JPEG compression is applied to the container image, with the quality factor $Q = 90$.
- **Poisson Noise** A Poisson-distributed noise with an intensity parameter $\alpha = 4$ is added to the container image.
- **Gaussian Filter** A Gaussian smoothing filter with a randomly selected kernel size $k = 3$ and standard deviation $\sigma = 1.0$ is applied to the container image.
- **Median Filter** A median filter with a randomly selected kernel size $k = 3$ is applied to the container image to simulate nonlinear smoothing effects.
- **Hue Adjustment** The hue of the container image is adjusted by a random shift $\Delta h \in [-0.1, 0.1]$ in HSV color space to simulate color perturbations.
- **Brightness Adjustment** The brightness of the container image is adjusted by a random scaling factor $\beta \in [0.9, 1.1]$.
- **Contrast Adjustment** The contrast of the container image is modified by a random scaling factor $\gamma \in [0.7, 1.3]$.

A.2. Masking Strategy

As illustrated in Fig. 5, we adopt two masking strategies from [28] during training—Irregular Masking and Box-shaped Masking—to simulate tampering attacks. For each training sample, one of the two strategies is randomly selected with equal probability (*i.e.*, 50%), and a sampled region is replaced with a randomly chosen image. The details of each masking strategies are as follows:

- **Irregular Masking** uses random brush strokes, where parameters such as stroke angle (up to 4 degrees), length, and width (ranging from 20 to 50 pixels) are varied. Each image is overlaid with between 1 and 5 such strokes.
- **Box-shaped Masking** generates rectangular regions with a fixed 10-pixel margin. The dimensions of each box are randomly sampled between 50 and 150 pixels for both width and height, and 1 to 3 boxes are applied per image.

A.3. Additional Implementation Details

We use an NVIDIA RTX 3090 Ti (24GB) for training and evaluation. The watermarking stage requires approximately 46 ms, and the self-recovery stage takes 166 ms. Thus, the total time for a full pipeline execution is approximately 212 ms. We apply the discrete wavelet transform (DWT) and its inverse (IWT) with the Haar wavelet to the input and output of the Invertible Watermarking (IW) modules, respectively. DWT changes the image shape from $I \in \mathbb{R}^{H \times W \times 3}$ to $I \in \mathbb{R}^{\frac{H}{2} \times \frac{W}{2} \times 12}$ and this transformation is reversed by IWT. The transformer encoder within the Watermark Generator (WG) consists of multi-head self-attention layers with 6 heads and a token embedding dimension of 192. The Image Enhancement (IE) modules adopt the configuration from the original paper [18], with an upscale factor of 1, indicating that the resolution remains unchanged.

A.4. Computational Cost

We compare the computational cost of ReImage and baseline models in terms of parameter size (Param), FLOPs, memory usage (Memory), and multiply–add operations (MAdds), all measured with an input image size of 256×256. As shown in Tab. 4, ReImage achieves lower FLOPs (64.76G) and MAdd (36.44G) compared to Imuge and Imuge+, while requiring fewer parameters overall. This

(a) Irregular Masking

(b) Box-shaped Masking

Figure 5. **Masking Strategy** (a) Irregular masking is implemented using random brush strokes and (b) box-shaped masking produces rectangular masked regions.

Table 4. **Computational Cost Comparison of Different Methods.** We compare the number of parameters (Param), floating-point operations (FLOPs), memory usage, and multiply–add operations (MAdd) across representative self-recovery models.

Methods	Param	FLOPs	Memory	MAdd
W-RAE	1.11M	18.35G	557MB	18.42G
Imuge	13.3M	90.52G	969MB	90.37G
Imuge+	32.0M	190.88G	1580MB	191.03G
ReImage	9.94M	64.76G	4570MB	36.44G

indicates that ReImage is computationally more efficient in practice. Although the memory usage of ReImage (4570MB) is relatively higher than that of baseline methods, ReImage runs smoothly on widely accessible mid-range GPUs such as the RTX 3090, demonstrating its practical deployability in real-world applications.

B. Additional Experimental Results

B.1. Recovery Quality under Varying Level of Noise

To evaluate the robustness of ReImage, we additionally conduct experiments under varying levels of distortion. Specifically, we evaluated our method under JPEG compression ($\text{QF} \in 60, 70, 80, 90$), Gaussian noise ($\sigma \in 1, 3, 5$), and Gaussian filter ($k \in 3, 5$). As shown in Tab. 5, the performance of all models degrades as distortions become more severe. Nevertheless, ReImage consistently outperforms the baselines across the entire range of settings. Under Gaussian noise, the PSNR of all models decreases as the standard deviation increases. However, ReImage consistently outperforms both Imuge+ and W-RAE by 8dB and 5dB, respectively, across all noise levels. A similar trend is observed under Gaussian Filter, where the baselines show lower reconstruction quality at larger kernel sizes, while ReImage continues to achieve higher PSNR and SSIM. In the case of JPEG Compression, even at the lowest quality factor ($\text{QF} = 60$), ReImage attains superior PSNR, SSIM, and LPIPS compared to baselines, and its results at $\text{QF} = 60$ already surpass those of the baselines at $\text{QF} = 90$. These results indicate that although performance inevitably declines with stronger distortions, ReImage retains a consistent advantage over the baselines, achieving higher fidelity and perceptual quality across all tested settings.

B.2. Geometric Common Degradation

To further demonstrate the robustness of ReImage under diverse degradations, we conduct experiments on three geometric distortions: Cropping, Rotation and Resizing, evaluated using PSNR. For this setting, both ReImage and Imuge+ are additionally fine-tuned with geometric distortion types, where Imuge+ is chosen as the strongest-performing baseline. As shown in Tab. 6, ReImage consistently outperforms Imuge+ across all geometric distortions, with relative improvements of 5.4% for cropping, 12.4% for rotation, and 21.8% for resizing. Although the performance under geometric distortions is generally lower than that observed under valuemetric distortions such as noise or compression, ReImage still achieves clear gains over the baseline. Notably, under the Resizing, ReImage attains a performance level comparable to that achieved under valuemetric distortions.

B.3. Ablation Study on Various Dataset

We further conduct additional experiments on diverse datasets to assess the generalization ability of ReImage. To ensure that all models are trained on an identical dataset, we train every model only on MS-COCO2017. We then evaluate the trained models separately on three datasets: MS-COCO2017, CelebA-HQ [39], and ILSVRC [27]. Tampering is simulated using Stable Diffusion Inpaint (SD Inpaint). For CelebA-HQ, we create attribute-based facial masks, and for ILSVRC, we employ SAM [16] to obtain object-level masks. Additionally, we randomly sample the same number of images as reported in Imuge+ (520 for CelebA and 1,047 for ILSVRC).

As shown in Tab. 7, ReImage achieves the highest performance across all datasets, consistently outperforming Imuge+ and W-RAE. On CelebA-HQ, ILSVRC, and MS-COCO, our model achieves an average improvement of over 32.6% in PSNR and 14.4% in M-PSNR compared to the strongest baseline (W-RAE). Notably, the performance is consistent across datasets, indicating no significant gap between them. These results demonstrate that our model generalizes robustly and effectively to diverse datasets used in prior studies.

Table 5. **Comparison of Self-Recovery Methods under Various Levels of Noise.** We evaluate model robustness by measuring the quality of recovered images across varying levels of Gaussian Noise, JPEG Compression and Gaussian Filter. Note that † denotes models retrained with these degradations.

Metrics	Methods	Gaussian Noise			JPEG Compression				Gaussian Filter	
		$\sigma = 1$	$\sigma = 3$	$\sigma = 5$	QF $= 60$	QF $= 70$	QF $= 80$	QF $= 90$	$k = 3$	$k = 5$
PNSR↑	Imuge+	24.02	23.42	22.85	22.87	23.19	23.53	23.88	23.80	23.53
	W-RAE †	26.60	26.30	25.76	16.10	17.13	17.95	18.44	10.81	11.63
	ReImage	**31.85**	**31.29**	**30.13**	**27.10**	**27.81**	**28.61**	**29.07**	**28.94**	**27.36**
SSIM↑	Imuge+	0.69	0.66	0.62	0.56	0.58	0.60	0.62	0.59	0.56
	W-RAE †	0.72	0.70	0.67	0.33	0.39	0.44	0.5	0.21	0.25
	ReImage	**0.91**	**0.88**	**0.82**	**0.83**	**0.85**	**0.86**	**0.88**	**0.87**	**0.84**
LPIPS↓	Imuge+	0.20	0.22	0.26	0.29	0.27	0.26	0.24	0.26	0.29
	W-RAE †	0.24	0.25	0.27	0.61	0.59	0.57	0.53	0.66	0.66
	ReImage	**0.13**	**0.15**	**0.19**	**0.23**	**0.22**	**0.20**	**0.16**	**0.20**	**0.24**

Table 6. **Comparison of Self-Recovery Methods under Common Geometric Distortions.** We evaluate the robustness of ReImage against three types of geometric distortions: Cropping, Rotation, and Resizing. Across all cases, ReImage consistently achieves higher PSNR performance compared to Imuge+.

Methods	Crop	Rotate	Resize
Imuge+	22.58	20.12	24.81
ReImage	23.81	22.61	30.21

B.4. Tamper Localization

To evaluate the effectiveness of the Tamper Localization (TL) module in ReImage, we compare it against recent self-recovery methods, including Imuge [33], Imuge+ [34], and W-RAE [5]. We simulate tampering using Stable Diffusion Inpainting [25], Stable Diffusion XL [24], and splicing attacks, guided by mask annotations from the valAGE-Set. Localization accuracy is measured using Intersection over Union (IoU), F1 score, and Area Under the ROC Curve (AUC).

As shown in Tab. 8, ReImage consistently achieves the highest tamper localization performance across various tampering types. Compared to Imuge [33], the strongest baseline among existing methods, ReImage achieves relative improvements of 34%, 30%, and 10% in IoU, F1, and AUC, respectively. Furthermore, while Imuge+ [34] performs poorly on unseen attacks such as SD Inpaint and SDXL, ReImage remains effective. This demonstrates that ReImage can accurately localize tampered regions even under unseen or diverse attack types. Qualitative localization results can be seen in Fig. 8. The figure shows that ReImage achieves higher performance than other baseline models.

B.5. Recovery Quality in Tampered Regions

We conduct additional experiments to evaluate image self-recovery performance on the valAGE-Set, comparing ReImage with recent state-of-the-art methods, including Imuge [33], Imuge+ [34], and W-RAE [5]. Specifically, we evaluate the M-PSNR across the same tampering types as in Tab. 8. As shown in Tab. 9, ReImage consistently achieves the highest performance, with relative improvements of 14%, 14%, and 14% over W-RAE. These results demonstrate the effectiveness of our approach in recovering tampered regions across diverse types of attacks.

B.6. Recovery Quality under Various Masking Ratios

To evaluate the robustness of our method under varying tampering intensities, we conduct experiments across different masking ratios. Specifically, we compare our approach with recent state-of-the-art self-recovery methods. Tampering is simulated via splicing attacks, where each mask contains 1 to 3 randomly placed geometric shapes (rectangles or circles). The masking ratio is varied across 10%, 20%, 30%, 40%, 50%, and 60% of the total image area to reflect different levels of tampering. We report PSNR and Mask PSNR (M-PSNR) as evaluation metrics.

As shown in Fig. 6, our method consistently achieves strong performance across all evaluation metrics and masking ratios. In particular, ReImage outperforms the best-performing baselines by more than 5dB in both PSNR and M-PSNR. Even at a masking ratio of 60%, ReImage achieves a higher PSNR than Imuge at 10%. Furthermore, while Imuge and W-RAE degrade significantly as the masking ratios increases, ReImage shows only a slight decrease of approximately 4dB and 0.39dB in PSNR and M-PSNR, respectively, between the 10% and 60% masking ratios. demonstrating robustness to tampering intensity.

Table 7. **Comparison of Self-Recovery Methods across Different Datasets.** We further evaluate the methods on additional datasets, including MS-COCO [17], CelebA-HQ [39], and ILSVRC [27]. M-PSNR denotes the PSNR of the recovered image computed only over the tampered regions.

	MS-COCO		CelebA-HQ		ILSVRC	
Methods	PSNR	M-PSNR	PSNR	M-PSNR	PSNR	M-PSNR
Imuge+	22.61	16.02	21.00	15.61	21.95	15.75
W-RAE†	23.61	20.98	22.99	20.22	21.28	21.36
ReImage	30.57	23.93	29.33	24.08	29.93	23.50

Table 8. **Comparison of Tamper Localization Performance across Self-Recovery Methods** We simulate tampering using three methods: SD Inpaint [25], SDXL [24], and splicing. Tamper localization performance is evaluated on the valAGE-Set using IoU, F1, and AUC metrics.

Models	Tamper Localization								
	SD Inpaint			SDXL			Splicing		
	IoU↑	F1↑	AUC↑	IoU↑	F1↑	AUC↑	IoU↑	F1↑	AUC↑
Imuge	0.61	0.69	0.90	0.60	0.68	0.90	0.61	0.69	0.90
Imuge+	0.20	0.25	0.89	0.24	0.25	0.89	0.62	0.69	0.97
W-RAE†	0.49	0.61	0.71	0.49	0.61	0.72	0.50	0.61	0.73
ReImage	**0.82**	**0.90**	**0.99**	**0.82**	**0.91**	**0.99**	**0.84**	**0.93**	**0.99**

Table 9. **Additional Comparison of Self-Recovery Methods on valAGE-Set** We evaluate all baselines using M-PSNR to assess recovery quality specifically within tampered regions. "M-PSNR" denotes the PSNR of the recovered image, computed only over the tampered regions.

Models	Recovered Image $\hat{I}_{\text{rec}}$		
	SD Inpaint	SDXL	Splicing
	M-PSNR↑	M-PSNR↑	M-PSNR↑
Imuge	10.66	10.59	10.61
Imuge+	14.91	14.86	16.54
W-RAE†	20.98	21.00	19.86
ReImage	**23.93**	**23.96**	**22.59**

Table 10. **Additional Comparison of Self-Recovery Methods under Photoshop Editing.** We evaluate recovery quality of all baselines under Photoshop editing operations to assess robustness against realistic manipulations. "M-PSNR" denotes the PSNR of the recovered image, computed only over the tampered regions.

Models	PSNR	SSIM	LPIPS	M-PSNR
Imuge+	21.13	0.62	0.24	13.50
W-RAED	24.94	0.68	0.28	20.70
ReImage	**28.85**	**0.88**	**0.15**	**22.75**

B.7. Ablation Study on Photoshop Editing

We evaluate ReImage under realistic tampering scenarios. To simulate such manipulations, we employ Adobe Photoshop with the MS-COCO [17] dataset. Specifically, we select objects within images and apply removal or inpainting operations using Photoshop's integrated generative model. A total of 100 attacked images are collected for evaluation. As shown in Tab. 10, ReImage achieves the highest recovery performance compared to the baseline models Imuge+ and W-RAE. Compared with the best-performing baseline, ReImage improves PSNR and M-PSNR by 3.91 dB and 2.05 dB, respectively. While the results under Photoshop editing are slightly lower than those in Table 2 of the main paper, our method still exhibits strong recovery performance. Furthermore, as shown in Fig. 10, qualitative results in clearly show that ReImage successfully restores tampered regions with high visual quality.

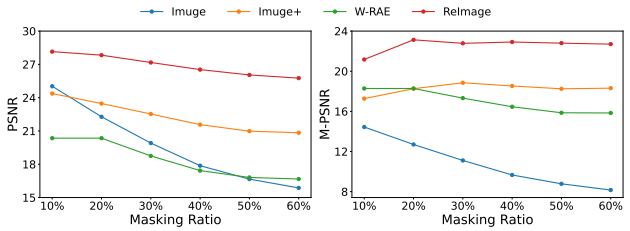

Figure 6. **Comparison of Self-Recovery Methods Across Various Masking Ratios.** We evaluate the recovery performance of all baseline methods under different masking ratios, ranging from 10% to 60%. Tampering is simulated via splicing attacks, where tampered regions are randomly generated using a combination of circles and rectangles.

B.8. Recovery Quality under the GT Mask Setting

We evaluate all methods under the ground-truth (GT) mask setting, where the ground-truth tampering masks are pro-

Table 11. **Impact of WG Loss in ReImage.** We evaluate the contribution of WG Loss ($\mathcal{L}_{\text{WG}}$). While using $\mathcal{L}_{\text{IE}}$ alone provides lower recovery quality, combining it with $\mathcal{L}_{\text{WG}}$ significantly enhances performance, demonstrating the importance of WG Loss in guiding more accurate image reconstruction.

		Container Image I_{con}			Recovered Image $\hat{I}_{\text{rec}}$			
$\mathcal{L}_{\text{WG}}$	$\mathcal{L}_{\text{IE}}$	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	M-PSNR ↑
✗	✓	36.56	0.94	0.11	25.00	0.85	0.20	15.78
✓	✓	36.10	0.93	0.10	30.57	0.87	0.17	23.93

Table 12. **Additional Study on the Impact of Core Components in ReImage** We examine the contribution of the core components-Pixel Shuffling (PS), Image Enhancement (IE), and the Watermark Generator (WG)-by evaluating all possible combinations. Using the full set of components consistently yields the best performance, both in terms of container and recovered image quality. M-PSNR denotes the PSNR of the recovered image computed only over the tampered regions.

			Container Image	Recovered Image	
PS	IE	WG	PSNR↑	PSNR↑	M-PSNR↑
✗	✗	✗	35.93	25.82	16.44
✓	✗	✗	25.78	22.95	18.32
✗	✓	✗	41.28	24.21	13.80
✗	✗	✓	42.24	23.53	13.61
✓	✓	✗	26.19	24.25	18.91
✗	✓	✓	39.40	26.47	16.96
✓	✗	✓	33.17	29.27	23.17
✓	✓	✓	36.10	30.57	23.93

vided instead of predicted ones. This setting isolates the recovery performance from the influence of localization accuracy. We compare Imuge+ [34], W-RAE [5], and ReImage. Tampering is simulated using Stable Diffusion Inpainting [25], guided by the mask annotations from the valAGE-Set.

As shown in Tab. 13, ReImage achieves the highest performance, outperforming Imuge+ by a relative margin of 16% in M-PSNR. Notably, ReImage outperforms both Imuge+ and W-RAE, even though they are provided with GT masks, whereas ReImage operates without them. These results highlight the effectiveness of our pipeline not only in self-recovery but also in tamper localization. Moreover, these results suggest that ReImage could achieve even greater performance with improved tamper localization accuracy.

B.9. Experiments with High Resolution and Additional Domain

We evaluate self-recovery performance at high resolutions. To assess our method across various resolutions, we first downsample the original image to 256×256, generate the watermarked image, and compute the residual between the watermarked and original images. The residual is then upsampled and added back to the original image. As shown in Tab. 14, ReImage achieves higher recovery quality at high resolutions (512×512: +15.9%, 1024×1024: +16.4% over the best baseline). Furthermore, we evaluate ReImage and the baselines on domain-specific datasets, including document and medical images. As shown in Tab. 15, ReImage maintains the best performance across different domains without additional retraining, using a model trained on COCO (Document (NL-DIR): +48.4%, Medical (ToP-Cow): +13.8%).

Table 13. **Comparison of self-recovery under GT mask localization settings.** We evaluate the recovery quality under GT-mask setting, where the module accurately identifies the tampered regions. "M-PSNR" denotes the PSNR of the recovered image, computed only over the tampered areas.

	GT Mask			
Models	PSNR	SSIM	LPIPS	M-PSNR
Imuge+	27.92	0.66	0.22	22.37
W-RAE†	24.40	0.73	0.30	21.64
ReImage	**31.70**	**0.88**	**0.16**	**25.95**

Table 14. **Comparison of Self-Recovery Performance under Various Resolutions.** We evaluate the quality of the container image and the recovered image at different resolutions using PSNR and M-PSNR.

	512×512			1024×1024		
Model	I_{con}	$\hat{I}_{\text{rec}}$		I_{con}	$\hat{I}_{\text{rec}}$	
Imuge+	33.61	25.31	18.47	34.01	25.68	19.80
W-RAE†	31.22	25.36	17.00	31.74	25.74	17.11
ReImage	**38.33**	**28.34**	**22.27**	**38.11**	**29.96**	**23.17**

Table 15. **Comparison of Self-Recovery Performance across Various Datasets.** We evaluate the quality of the container image and the recovered image across different domains (e.g., document, medical).

	NL-DIR (Dcoument)			TopCoW (Medical)		
Model	I_{con}	$\hat{I}_{\text{rec}}$		I_{con}	$\hat{I}_{\text{rec}}$	
Imuge+	28.91	19.13	15.51	34.01	26.98	18.84
ReImage	**36.88**	**28.39**	**21.94**	**37.40**	**30.69**	**23.46**

C. Discussion

C.1. Effect of WG Loss

The loss in $\mathcal{L}_{\text{WG}}$ is additionally introduced to encourage the Watermark Generator (WG) to retain some capability to recover the original image. To assess its necessity, we compare the performance of using only $\mathcal{L}_{\text{IE}}$ versus using both $\mathcal{L}_{\text{WG}}$ and $\mathcal{L}_{\text{IE}}$. As shown in Tab. 11, we observe a clear improvement in the quality of the recovered image when both $\mathcal{L}_{\text{WG}}$ and $\mathcal{L}_{\text{IE}}$ are used. Specifically, $\mathcal{L}_{\text{WG}}$ encourages the inverse process of WG to retain a coarsely recovered version of the original image, allowing the Image Enhancement (IE) module to focus only on restoring the remaining missing details. Without $\mathcal{L}_{\text{WG}}$, however, the combined WG and IE behave more like a single large watermark decoder, lacking the structural separation between coarse recovery and refinement. This may prevent the model from leveraging the inductive bias of performing a rough watermark recovery before enhancement, leading to suboptimal performance.

C.2. Effect of Each Component

Tab. 12 includes settings beyond those presented in the main paper, enabling a comprehensive comparison across all combinations of PS, WG, and IE components. This enables a more comprehensive analysis of each component's individual contribution. As shown in the Tab. 12, each component of our model contributes to improved image self-recovery performance. We observe consistent performance gains as more components are combined, with the best results achieved when all three (PS, WG, and IE) are used together. This indicates that each module contributes complementary benefits, rather than interfering with one another. Additionally, the relatively high quality of the container image without PS can be attributed to the model embedding less information, which in turn makes recovery more difficult.

C.3. Tamper Localization

In this section, we discuss why ReImage achieves higher performance compared to the baselines [5, 34]. Both ReImage and W-RAE [5] leverage the locality and fragility properties of the INN block described in Sec. 3.2. Specifically, the corrupted regions in the container image tend to align with the missing content in the extracted secret image. This property consistently holds regardless of the type of attack model or method. However, Imuge+ [34] does not leverage any attack-agnostic property for localization, and thus fails to operate reliably when exposed to unseen attack patterns. As shown in Tab. 8, Imuge+ performs well under the splicing attack used during training, but its performance drops significantly under SD Inpainting [25] and SDXL [24] attacks, which are not included in the training phase.

By comparison, W-RAE performs worse than ReImage and even under the splicing attack, it underperforms compared to Imuge+. W-RAE employs two watermarks for self-recovery: a secret image for reconstructing the original image, and a separate watermark for localization. The localization watermark is first embedded into the secret image, which is then embedded into the cover image. Since W-RAE must store and perfectly recover both both watermarks, the localization process becomes noisy, making it difficult to precisely identify the attacked regions. In contrast, ReImage only uses the shuffled secret image for both localization and reconstruction, which enables it to achieve high localization performance.

C.4. Pixel Shuffling

In this section, we discuss the importance of Pixel Shuffling (PS). As described in Sec. 3.3.1, PS induces spatial misalignment between the corrupted region of the container image and the corresponding region of the secret image. Additionally, during the inverse transformation, the shuffled pixels are mapped back to their original positions, dispersing the bulk corruption throughout the entire image and providing sparse cues for reconstruction across the whole image. These two properties make PS a crucial mechanism for enabling reliable self-recovery even under malicious attacks.

However, PS inevitably introduces an inherent drawback: its pixel-level permutation increases the entropy of the secret image. To mitigate this effect, we incorporate a Watermark Generator (WG) designed to suppress high-frequency components. While WG effectively reduces much of the high-frequency noise introduced by PS, the remaining frequency level is still higher than that of the original secret image.

However, alternative approaches that attempt to replace PS exhibit clear limitations. As discussed in Sec. 3.3.1, simple shifting operations fail to sufficiently disperse corrupted pixels; even after the inversion, the corrupted region remains spatially concentrated, making reconstruction considerably more challenging. Likewise, applying PS on a coarse grid (e.g., W-RAE) provides insufficient spatial dispersion, again leaving the corruption clustered and limiting recovery performance.

We further examine whether explicit pixel permutation can be removed entirely by replacing the Invertible Watermarking (IW) module with a Transformer-based architecture. Since Transformers can capture long-range spatial dependencies, one might expect information at a given location to be propagated to distant pixel positions, allowing those positions to also incorporate that information. Such long-range interactions could, in principle, induce a form of learned spatial misalignment that might help distribute localized corruption more broadly even without explicit pixel permutation. To test this, we construct a 12-layer Transformer-based IW model using the same training

Table 16. **Additional Study on the Impact of Pixel Shuffling in ReImage.** We compare ReImage with a variant where the Invertible Watermarking (IW) module is replaced with a Transformer-based architecture and Pixel Shuffling (PS) is removed. Although the Transformer can capture long-range spatial dependencies, it still falls short of the explicit spatial misalignment provided by PS. M-PSNR denotes the PSNR of the recovered image computed only over the tampered regions.

	Container Image I_{con}			Recovered Image $\hat{I}_{\text{rec}}$			
Model	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	M-PSNR↑
Variant	39.96	0.97	0.07	27.91	0.87	0.17	18.71
ReImage	36.10	0.93	0.10	30.57	0.87	0.17	23.93

Table 17. **Comparison with additional watermarking methods**

	Container	Recovered	
Model	PSNR	PSNR	M-PSNR
EditGuard-D	30.37	22.20	18.81
MaskWM	41.96	19.53	14.82
ReImage	36.10	**30.57**	**23.93**

and evaluation pipeline. As shown in Tab. 16, removing PS increases the container quality due to the reduced entropy of the embedded representation, which aligns with the analysis in Sec. C.2. However, the recovered quality in the attacked regions substantially degrades (M-PSNR drops by approximately 27%). This suggests that the Transformer-based variant does not naturally converge toward inducing the spatial misalignment between the container image and the secret image. The explicit misalignment introduced by PS provides the necessary inductive bias to guide the model toward recovering the attacked regions.

C.5. Comparison with Robust Watermarking Methods

In this section, we compare the conceptual positioning of ReImage with robust image watermarking and evaluate its performance against existing methods. ReImage shares key characteristics with robust image watermarking in that a watermark is embedded, survives common degradations, and exhibits locality and fragility under tampering. However, the key difference lies in its adaptation for self-recovery. Existing robust image watermarking methods rely on spatial alignment during the embedding process, making them unsuitable for self-recovery.

Furthermore, we evaluate additional baselines with available code, including EditGuard and MaskWM. Since these methods are not designed for self-recovery, we retrain them by replacing the watermark with the cover image. As shown in Tab. 17, EditGuard and MaskWM perform 37% and 56% worse than ReImage in terms of recovered image quality, respectively. This indicates that strong cover–watermark alignment remains in existing methods, highlighting the importance of cover–watermark misalignment.

C.6. Intermediate Result

In this section, we visualize the intermediate results of ReImage after passing each module (I_{org}, I_{sec}, $\mathcal{S}(I_{\text{sec}})$, I_{con}, I_{att}, $\mathcal{S}(\hat{I}_{\text{sec}})$, $\hat{I}_{\text{sec}}$, $\hat{I}_{\text{org}}$, $\hat{I}_{\text{enh}}$, and $\hat{I}_{\text{rec}}$). As shown in Fig. 7, the secret image processed by the Watermark Generator (I_{sec}) exhibits suppressed high-frequency components after the shuffling step ($\mathcal{S}(I_{\text{sec}})$). This design helps maintain both high container image quality and high recovered image fidelity.

The reconstructed original image $\hat{I}_{\text{org}}$ successfully restores the original image to a meaningful extent, but—as described in Sec. C.2—still contains globally distributed degradations introduced by shuffling process. Passing this result through the Image Enhancement (IE) module mitigates these degradations and produces a cleaner output ($\hat{I}_{\text{enh}}$). Finally, regions that were not attacked are replaced with higher-quality container pixels identified by the localization module, further improving the quality of the final recovered result. Overall, each stage progressively refines the signal and contributes to the final reconstruction quality.

D. Limitations

ReImage consistently demonstrates high fidelity in both container and recovered images across various tampering methods, as verified through both qualitative and quantitative evaluations. Nonetheless, it exhibits limitations in certain scenarios. In particular, ReImage often fails to reconstruct tampered regions when geometric transformations such as rotation and cropping are applied, as these distort the embedded secret image and degrade the information necessary for accurate recovery.

In addition, tamper localization can impose constraints on performance. When the predicted region is smaller than the actual tampered area, parts of the attacked content may remain unrecovered, since the corresponding region from the container image—including tampered parts—is used to overwrite the recovered image. This may lead to an underestimation of ReImage's true capability.

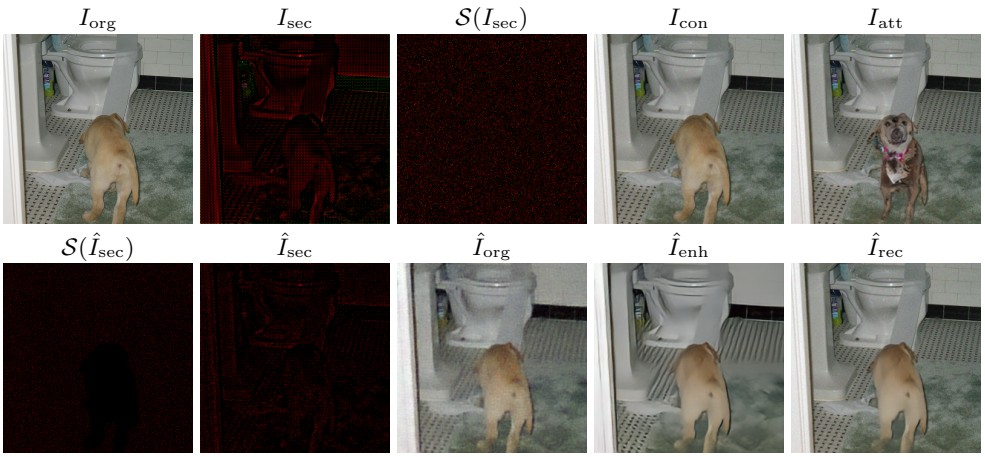

Figure 7. **Visualization of Intermediate Results in ReImage.** We visualize the intermediate outputs after each module (I_{org}, I_{sec}, $\mathcal{S}(I_{\text{sec}})$, I_{con}, I_{att}, $\mathcal{S}(\hat{I}_{\text{sec}})$, $\hat{I}_{\text{sec}}$, $\hat{I}_{\text{org}}$, $\hat{I}_{\text{enh}}$, and $\hat{I}_{\text{rec}}$). Each stage progressively refines the signal and contributes to the final reconstruction quality.

E. Additional Visual Results

In this section, we extend our evaluation by visualizing the robustness of ReImage against various common degradations (*i.e.*, Gaussian Noise (G.N.), JPEG Compression (JPEG), Gaussian Filter (G.F.), Median Filter (M.F.), Poisson Noise (P.N.), Hue Adjustment, Brightness Adjustment and Contrast Adjustment), as shown in Fig. 9. These experiments show that ReImage remains effective under various content-preserving modifications. Additionally, we present further qualitative comparisons on the valAGE-Set between ReImage and other baseline methods. As shown in Fig. 11 and Fig. 12, ReImage consistently generates high-fidelity container and recovered images, even under tampering attacks and degradations. In contrast, other methods either fail in both aspects or struggle to balance imperceptibility and accurate recovery.

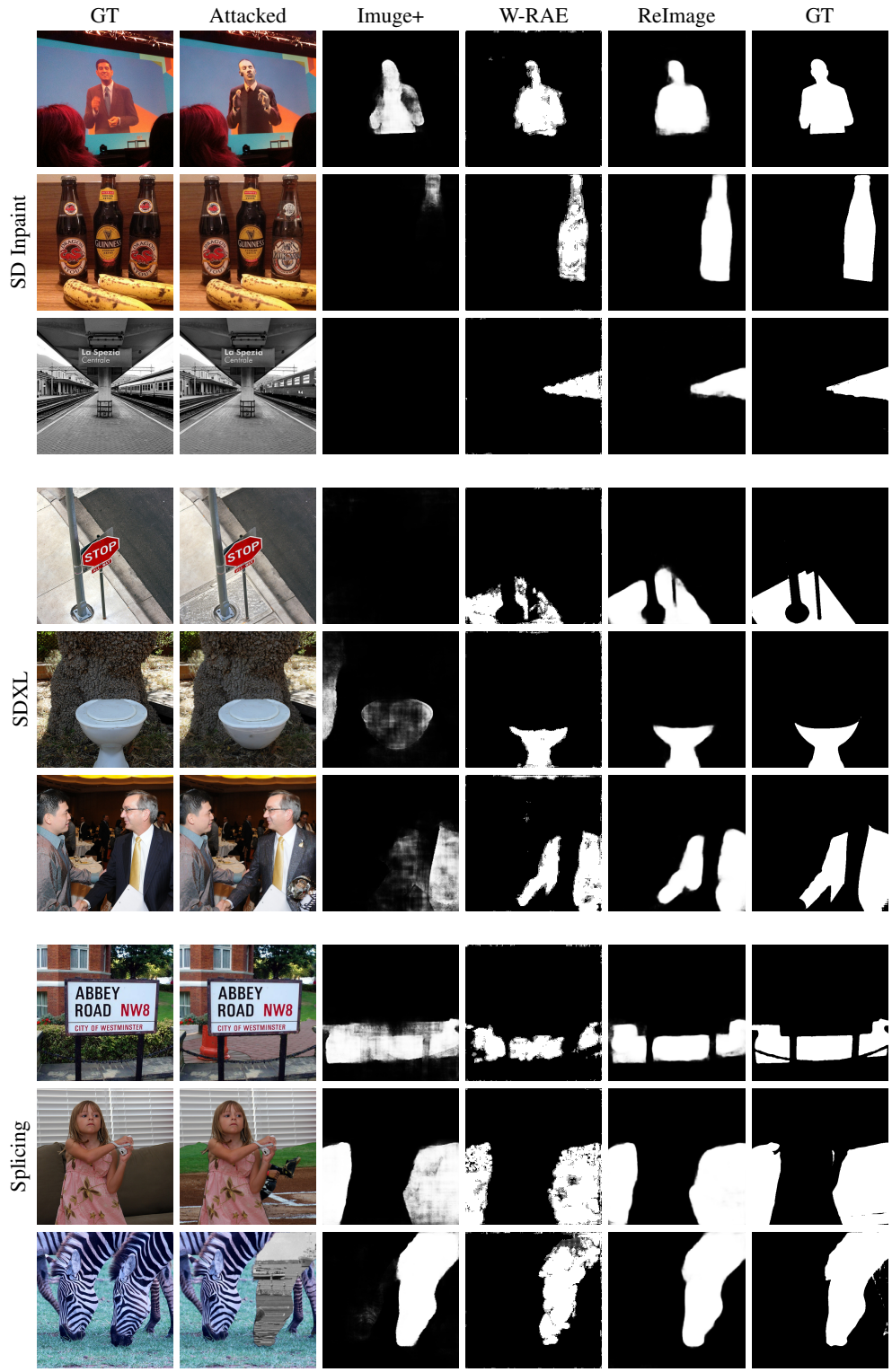

Figure 8. **Visual Comparison of Localization Modules from Different Self-Recovery Methods.** We qualitatively evaluate the performance of the localization module. Tampering is simulated using SD Inpaint [25], SDXL [24], and splicing. The proposed ReImage demonstrates high localization accuracy.

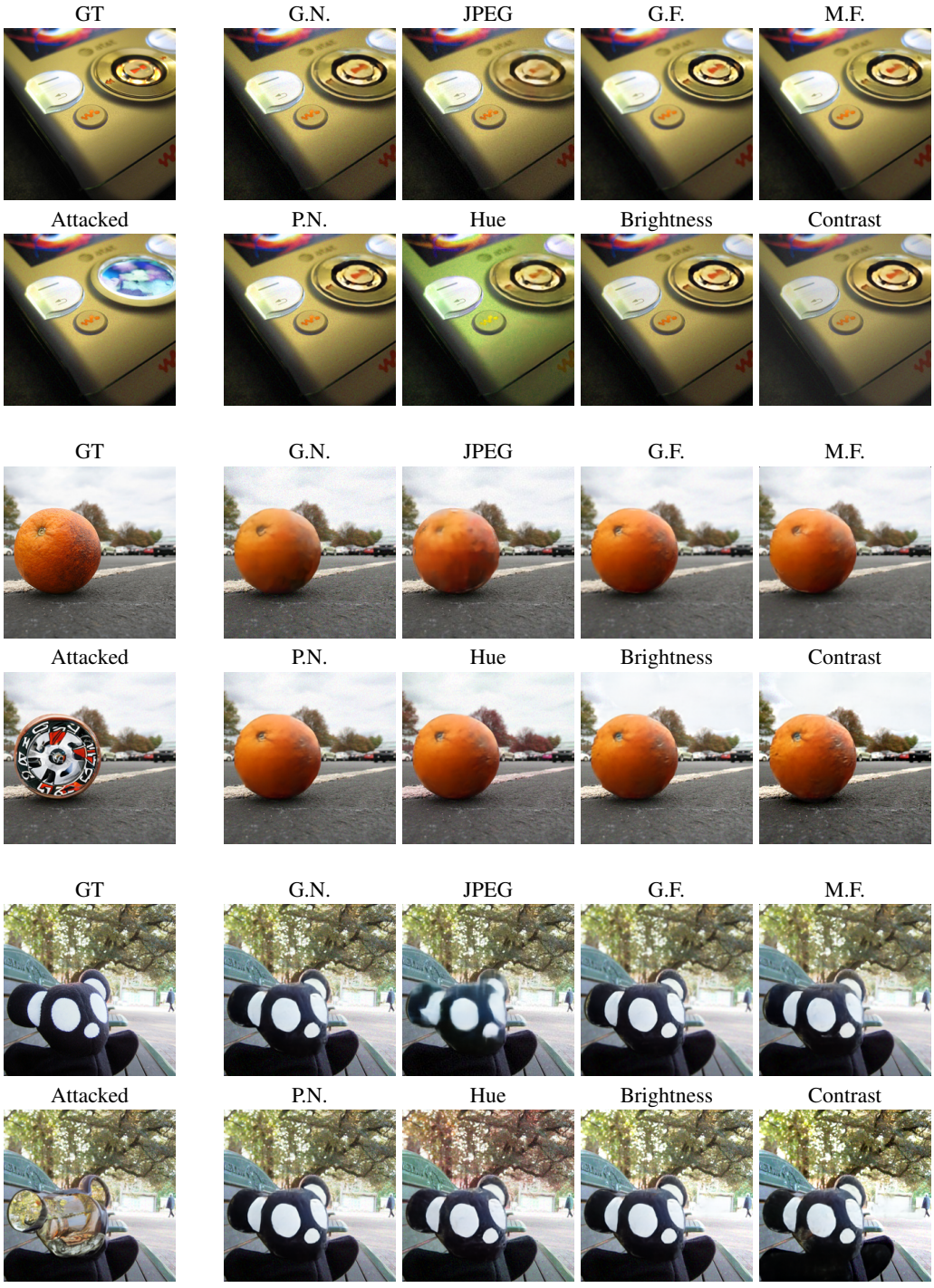

Figure 9. **Qualitative Results under Various Common Degradation Types** We qualitatively evaluate the robustness of ReImage under eight common degradation types—Gaussian Noise (G.N.), JPEG Compression (JPEG), Gaussian Filter (G.F.), Median Filter (M.F.), Poisson Noise (P.N.), Hue Adjustment, Brightness Adjustment and Contrast Adjustment.

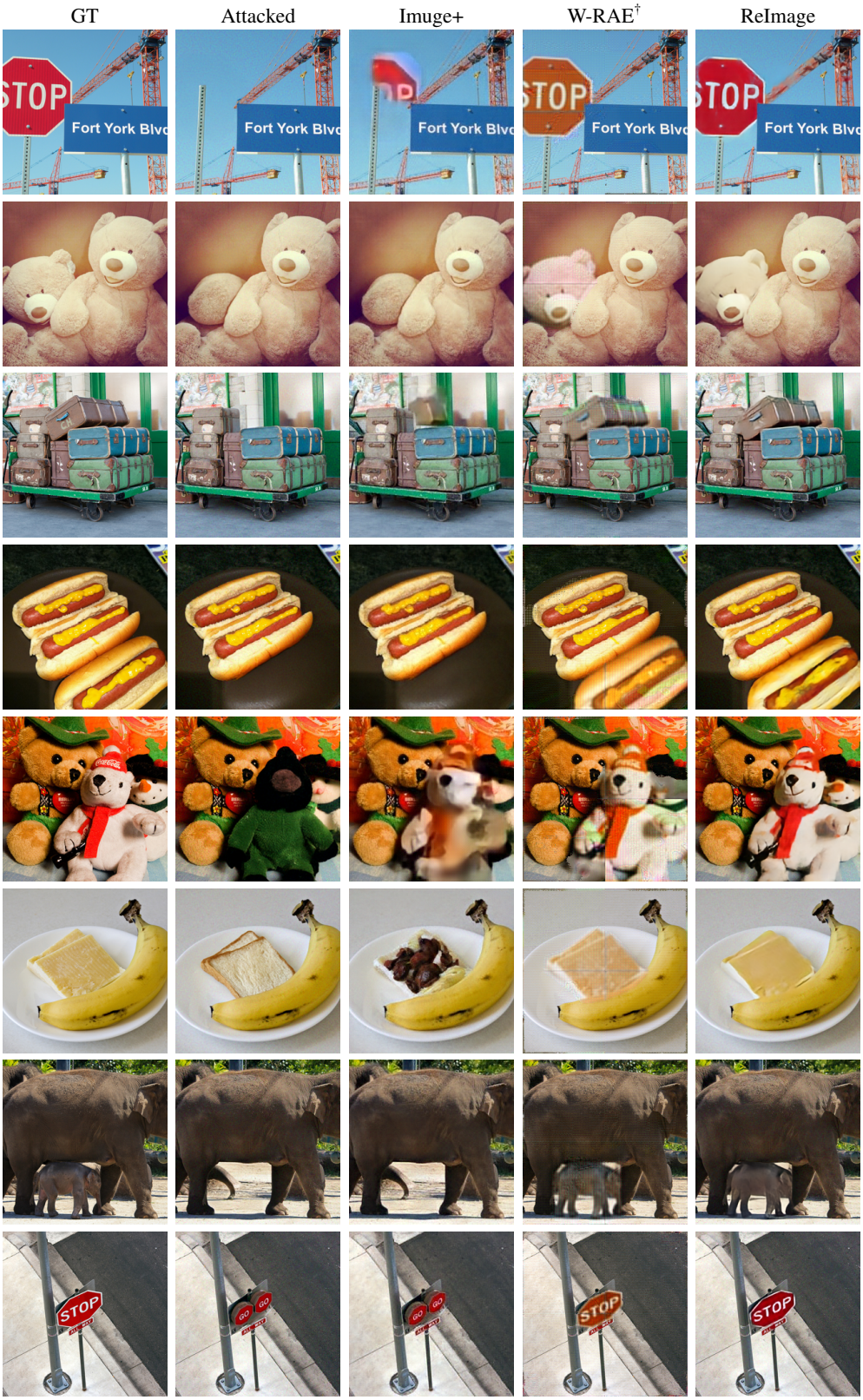

Figure 10. **Visual Comparison of Recovered Images from Different Self-Recovery Methods under Photoshop Editing** Tampering is simulated using Adobe Photoshop. ReImage demonstrates high-fidelity reconstruction of the original image under realistic manipulation

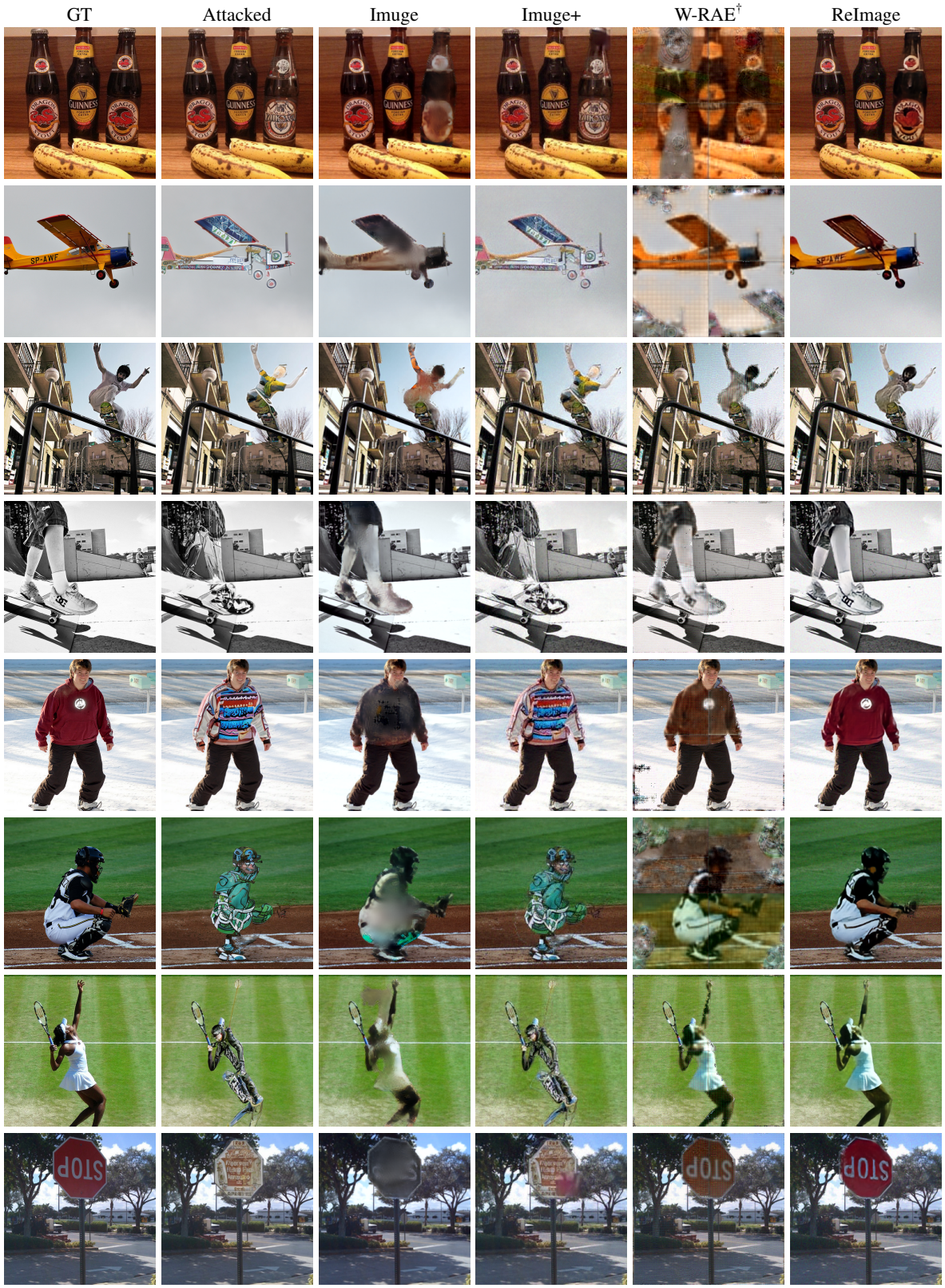

Figure 11. **Visual Comparison of Recovered Images from Different Self-Recovery Methods on valAGE-Set** Tampering is simulated using SD Inpaint [25]. ReImage demonstrates high-fidelity reconstruction of the original image.

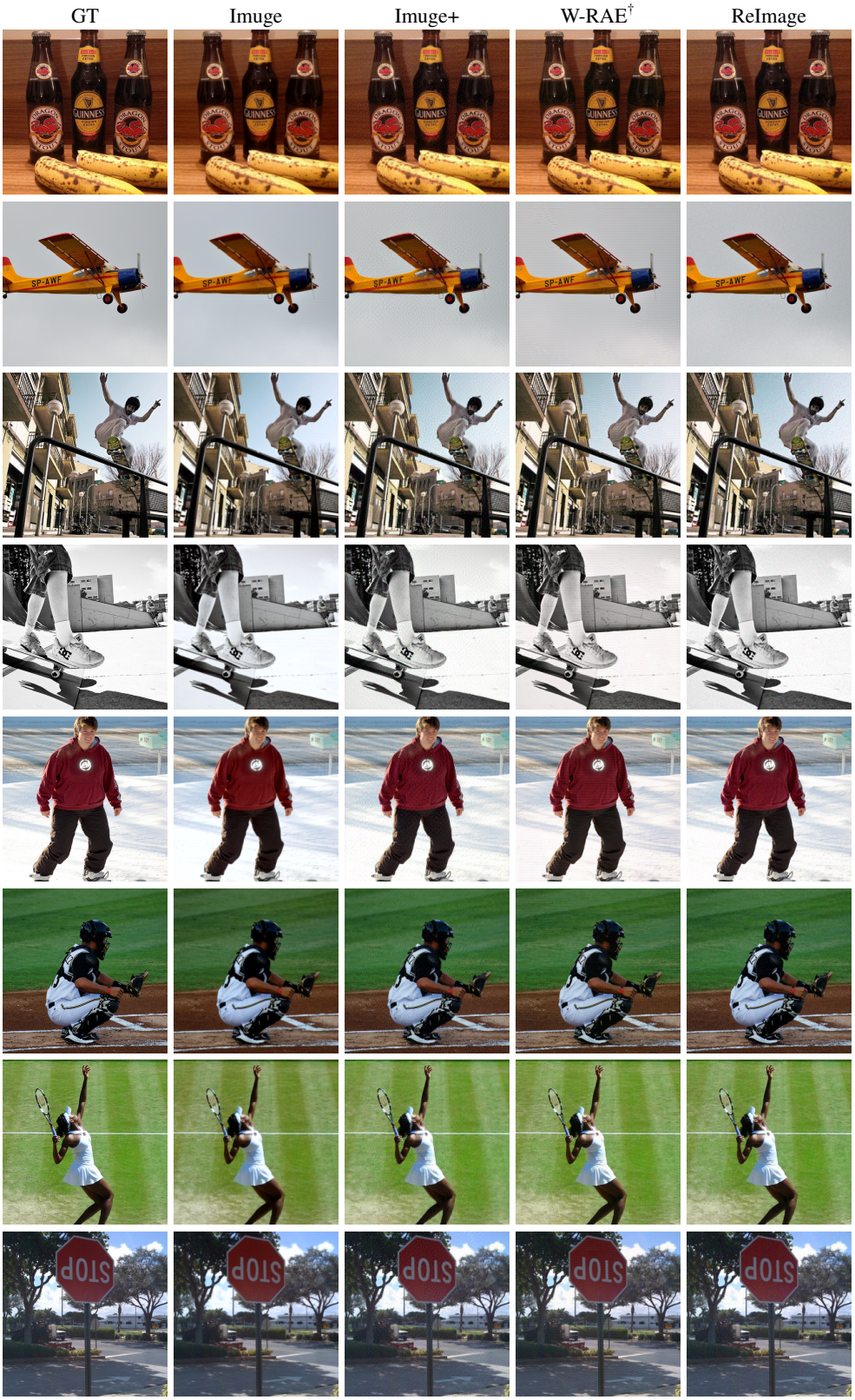

Figure 12. **Visual Comparison of Container Images from Different Self-Recovery Methods on valAGE-Set** Tampering is simulated using SD Inpaint [25]. ReImage demonstrates the ability to embed watermarks into the target image with minimal visual distortion.