International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(1s) 2026 pp. 181-199
https://doi.org/10.70917/ijcisim-2026-1894
Article

Real-Time Defect Removal and Mask-Guided Image Inpainting of Live-Captured Images Using a Pretrained LaMa-Based Inpainter with Bilevel Hypergradient Adapter Optimization

Suma N[1], Kusuma Kumari B.M[2]

1Department of Studies and Research in Computer Applications, Jnanasiri Campus, Tumkur University, Bidrakatte,Tumkuru, India; ORCID: 0009-0008-9300-4348
2Department of Studies and Research in Computer Applications, Jnanasiri Campus, Tumkur University, Bidrakatte,Tumkuru, India; ORCID : 0000-0003-2961-7752
Corresponding Author : Suma N

Abstract: Image inpainting is used when part of an image is missing or damaged. In live-captured images, this is harder because the defect shape is not always regular. The repaired part should look like it belongs to the same image. This work proposes a real-time defect removal and mask-guided image inpainting framework for such images. A pretrained Large Mask Inpainting (LaMa) model is used as the base inpainter. LaMa is first adapted using the Landscape dataset. Small adapter layers are then trained with Mask-Aware Bilevel Hypergradient Optimization (MABHO) to improve the repaired region. The output is checked with full-image scores and masked-region scores. The proposed Masked Region Restoration Quality Loss (MRQL) score is also used to measure the repair quality inside the damaged area. Training and 5-fold cross-validation use 300 Landscape images. For external validation, 150–180 Intel images are used. Real-time testing is done on 15 camera-captured images. Compared with the frozen LaMa baseline on the Landscape dataset, Peak Signal-to-Noise Ratio (PSNR) increases from 20.96 to 22.84 dB, and Structural Similarity Index Measure (SSIM) increases from 0.811 to 0.864. Learned Perceptual Image Patch Similarity (LPIPS) decreases from 0.184 to 0.129. The framework further achieves 20.72 dB Masked-PSNR, 0.823 Masked-SSIM, and 0.884 MRQL. These results show that the proposed method supports practical defect-aware restoration for live-captured images.
Keywords: Image inpainting; mask-guided inpainting; LaMa; adapter-based refinement; Mask-Aware Bilevel Hypergradient Optimization; masked-region restoration

1. Introduction

Image inpainting restores missing, damaged, or unwanted parts of an image by using nearby image content [1]. It is useful for removing scratches, stains, small defects, occlusions, and damaged surface regions. Deep learning improves this task by learning structure, context, and texture patterns from image data [2]. Still, restoration becomes difficult when the damaged part has an uneven shape, crosses object edges, or lies inside a textured region where both detail and overall appearance must be kept [3]. The task becomes harder in live-captured images. Such images may have uneven lighting, slight blur, background clutter, and defects with different shapes. Texture also changes from one scene to another. So, the model cannot just fill the missing area. The restored part must match nearby edges, local texture, and the full image appearance. Attention-based, structure-guided, and video-aware methods improve inpainting quality, but they also increase model size and computation [4]. This leaves a practical gap between offline inpainting and real-image defect correction.

Copyright: © 2026 by the authors

Full-image scores do not always show the real quality of the repaired region. PSNR and SSIM are useful, but they can look good when most of the image remains unchanged. The masked area may still have small errors. This matters in defect removal because the corrected region is the main area to check. New diffusion-guided and large inpainting models improve visual quality, but many of them are heavy for lightweight adaptation and practical inference [5]. So, this work also uses masked-region measures and the proposed MRQL score to check how well the damaged area is restored and how naturally it blends with nearby texture and structure. Here, LaMa is used as the base inpainter because it already has image completion ability. It is then adjusted for defect removal using small adapter layers. MABHO improves these adapter updates, mainly in the repaired region. The final image is checked with full-image scores, masked-region scores, and the proposed MRQL score. This idea also follows earlier work that considers both local repair and full-image consistency [6].

The restored image is checked in two parts: the full image and the damaged area. Full-image scores check whether the whole image looks stable, and masked-region scores check the repaired defect area separately. The experiments use 5-fold cross-validation, Intel image testing, and camera-captured samples to check the method in different conditions. The rest of the paper is organized as follows. Section 1.1 gives the related work. Section 1.2 explains the research gap and motivation. Section 2 presents the materials and methods. Section 3 gives the results and discussion. Section 4 concludes the paper.

1.1 Related Work

Image inpainting has been used for restoring missing, damaged, or unwanted parts of an image. It helps in cases where scratches, stains, occlusions, surface defects, or unwanted objects affect image quality. Earlier methods mostly focused on filling missing pixels, but recent work moved toward restoring structure and texture more naturally. This became important when the missing region was irregular, large, or placed in a visually important area. Because of this, many studies focused on texture, structure, context, and reconstruction quality under different mask conditions. These points are also important for real-time defect correction, where the restored image must look natural and the output must be generated quickly from live-captured inputs.

Image inpainting has changed a lot with deep learning. Earlier work made it clear that a good restored image should keep the main structure, object edges, and natural flow of the image [1]. Later studies used encoder-decoder networks, GANs, attention methods, and transformers to handle irregular holes and complex scenes [2,3]. Some surveys also discussed video inpainting, where large missing regions, frame consistency, and processing time remained difficult issues [4]. Diffusion-based editing produced more natural-looking results, but its heavy computation made it harder to use in real-time settings [5]. These works make one thing clear. A filled patch alone is not enough. The repaired part must follow the mask and match the nearby image content. It should also be produced fast enough for real use.

Later methods handled this by using refinement and mask guidance. Local and global refinement helped restore nearby texture and wider image structure [6]. Mask-guided convolution made the repair focus on the damaged region [7]. Context-adaptive learning used nearby content while filling the missing part [8]. Wider feature use helped when the missing area was large [9]. Two-stage methods first gave a rough repair and then improved it [10]. Gated convolution and mask-aware processing helped separate valid and missing regions [11]. These points show why mask handling, context, and refinement are needed for defect removal.

GAN-based and structure-focused methods tried to make the filled region look more natural. Adaptive U-Net GAN used adversarial learning to blend the repaired part with the rest of the image [12]. Dual-resolution GAN worked with features at different scales, which helped in low-resolution restoration [13]. Uncertainty-based inpainting treated some missing regions as open-ended, since one hole can have more than one possible repair [14]. Later methods gave more attention to structure and texture. Sparse self-attention helped capture wider image relations with less attention cost [15]. Bidirectional flow with joint texture and structure learning supported both fine detail and shape consistency [16]. Coarse-to-fine learning handled structure first and then refined texture [17]. Dual degradation fusion showed that restoration becomes harder when missing regions come with other image damage [18]. Hierarchical texture-aware learning used contextual attention and multiscale fusion to preserve fine texture [19]. Recent work also used transformers, guidance, and diffusion for harder inpainting cases. Mask-aware transformer methods helped the model read valid and missing regions more clearly [20]. Diffusion-based methods used extra guidance when the nearby image content was not enough for proper completion [21]. Sketch-guided inpainting gave more control over the restored structure [22]. Reference-guided transformers used related visual content to support the repair [23]. Some newer methods combined mask-aware transformers with diffusion for large missing areas [24]. Prior-based learning also used wider image context for better restoration [25]. These methods handled difficult missing regions better. Still,

many of them needed heavy models, extra guidance, or more inference time. This made them less suitable for real-time defect correction.

Overall, these studies cover texture repair, mask handling, and structure recovery [6–14]. Later work includes attention-based restoration, structure-texture learning, and multiscale features [15–19]. Transformer and diffusion-guided methods support difficult missing-region repair [20–25]. Many of these methods still need extra guidance, more stages, or longer processing time [20–25]. So, a lighter mask-guided method is needed for real images, with clear repair and lower processing load.

1.2 Research Gap and Motivation

The gap is mainly in live-captured defect correction. Real captured images can have uneven lighting, weak boundaries, and defect shapes that change from one image to another. A useful method should repair the masked area clearly, keep the surrounding image stable, and avoid heavy retraining. For this reason, the present work uses pretrained LaMa with small adapter layers. MABHO is used to update these adapters and improve the damaged region. This keeps the LaMa backbone fixed while giving task-specific correction for defect removal.

2. Materials and Methods

2.1. Overview of the proposed framework

The approach is designed for defect repair in live-captured images. First, the damaged area is marked using a mask. The repair is done only in that marked area, and the nearby clear image content is used as guidance. Before restoration, the image is resized and normalized so that the input format stays the same. The masked image is then passed to the pretrained LaMa inpainter. Since LaMa already knows image completion, the restoration does not start from the beginning. Small adapter modules are added for the defect-removal task. The LaMa backbone is kept fixed, and only the adapter weights are updated. Bilevel hypergradient optimization is used for this adapter update. The inner step improves the repair on training images. The outer step checks that update using the final restoration loss and adjusts the adapter again. This helps the model handle different defect shapes with less training. The process starts with image capture and input preparation. Then the defect mask is created, LaMa restores the masked image, and the adapter improves the restored part. The final image is checked using reconstruction and visual quality measures. Figure 1 shows the overall architecture, and Figure 2 shows the workflow. The restoration process is represented as:

$$I_r = F_{\theta,\phi}\,(I_m, M)$$

where I_m denotes the masked input image, Mdenotes the defect mask, F denotes the pretrained LaMa-based image inpainting model, θ represents the pretrained model parameters, ϕ represents the trainable adapter parameters, and I_r denotes the restored output image.

The bilevel optimization process for adapter learning can be expressed as

$$\phi^* = \arg\min_{\phi}\ \mathcal{L}_{outer}(\theta, \phi^*(\mathcal{L}_{inner}))$$

subject to

$$\phi^*(\mathcal{L}_{inner}) = \arg\min_{\phi}\ \mathcal{L}_{inner}(\theta, \phi)$$

where $\mathcal{L}_{inner}$is the task-level restoration loss used to optimize the adapter within the inpainting model, and $\mathcal{L}_{outer}$is the outer objective that guides the adapter update through hypergradient-based refinement. This way, LaMa is adjusted for defect removal without training the full model again.

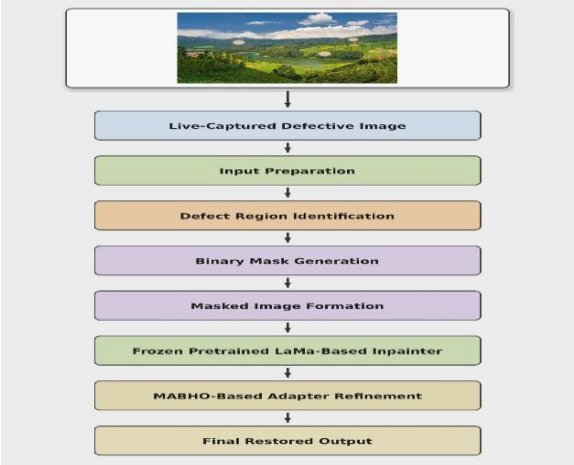

Figure 1. Overall architecture of defect removal framework

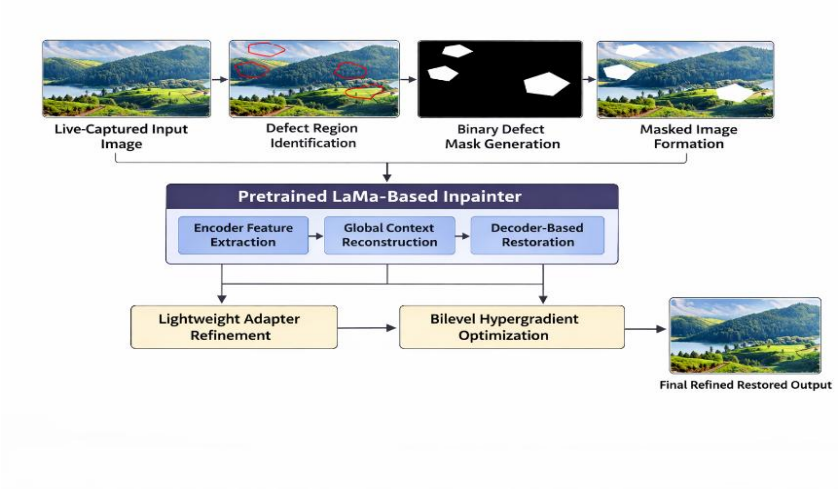

Figure. 2. Detailed operational workflow of the proposed framework

2.2 Dataset Description

This study uses three image sources. The Landscape Pictures dataset is used for training and 5-fold cross-validation. The Intel Image Dataset is used for external validation. A small set of camera-captured images is used for real-time testing. The dataset details are given in Table 1. Around 300 images are randomly selected from the Kaggle Landscape Pictures dataset. Each image is resized to 256 × 256 pixels. These images are used for training and testing the proposed model.

Table 1. Dataset configuration used in the study

Dataset source	Role in the study	Number of images	Usage
Landscape Pictures (Kaggle)	Main dataset	300	Training and 5-fold cross-validation
Intel Image Dataset (Kaggle)	External dataset	150–180	External validation / generalization testing
Camera-clicked live images	Real-time test set	15	Inference and qualitative evaluation only

Each image is combined with a defect mask to create a damaged input. The same mask method is used in training and testing to keep the setup consistent. The masked image I_m is defined as

$$I_m = I \odot (1 - M)$$

where I is the original image, M is the defect mask, and $\odot$ denotes element-wise multiplication.

150–180 images from the Kaggle Intel Image Dataset are used for external validation. These images are used only for testing. They are not used during training or 5-fold cross-validation. Since the work is on image restoration, the class labels are not considered. This dataset checks how the trained model works on unseen images. For real-time testing, 15 camera-captured images are used. The camera-captured images are kept only for inference and visual checking. This helps note how the model works on real captured inputs. Raw samples from the Landscape, Intel, and camera-captured images are shown in Figure 3. All images are resized to 256 × 256 pixels before processing. This keeps the input size the same in all stages.

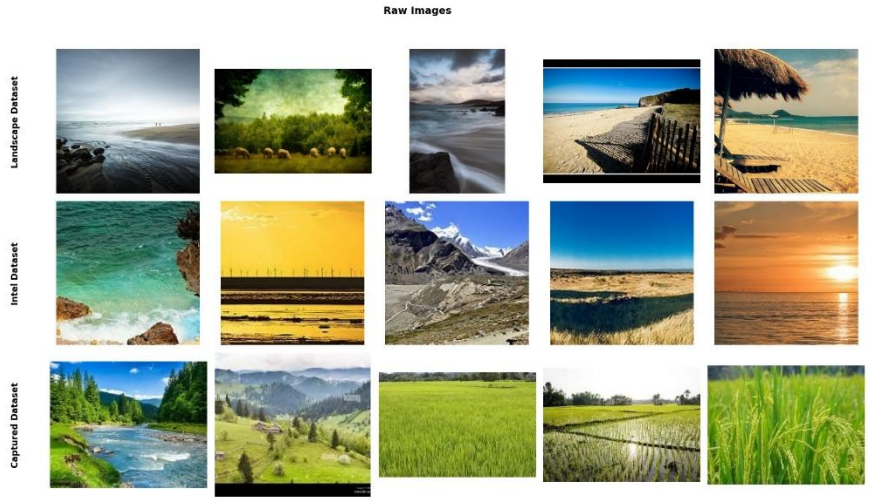

Figure 3. Representative raw images from Landscape, Intel, and real-time captured datasets

2.3 Live Image Acquisition and Input Preparation

The framework uses public dataset images and camera-captured images. For real-time testing, images are taken using a normal digital or mobile camera. Each input image is resized to 256 × 256 pixels, converted to RGB, and normalized before restoration. This prepares the input for the LaMa-based inpainting model. The preprocessed images are shown in Figure 4.

Figure 4. Corresponding preprocessed images after resizing and normalization

Let the prepared input image be represented as

$$I \in R^{256\times256\times3}$$

Here, I refers to the prepared RGB image. Once the image is ready, it moves to the next stage, where the defect mask is created and the restoration process starts. The same input format is used for training, validation, and real-time testing.

2.4 Defect Region Identification and Mask Generation

The defect area is first marked before restoration. For the public dataset, synthetic masks are used so that the experiment stays the same for all images. For real-time images, the defect region is marked from the captured input and converted into the same mask format. In both cases, the mask has the same size as the input image. The defect mask is represented as:

$$M\in\{0,1\}^{256\times256}$$

where $M(x,y)=1$ denotes the defective region and $M(x,y)=0$ denotes the valid image region. The generated mask is given to the pretrained LaMa-based inpainting model along with the prepared image. The model restores only the masked part, while the nearby valid image content guides the repair.

2.5 Pretrained Inpainter for Image Restoration

The pretrained LaMa inpainter is used for the first restoration. The damaged image and mask are given as input. The mask marks the repair area, and the clear part of the image guides the filling process. LaMa is already pretrained, so it reduces training work and gives a stable base. The initial restoration process is shown in Figure 5.

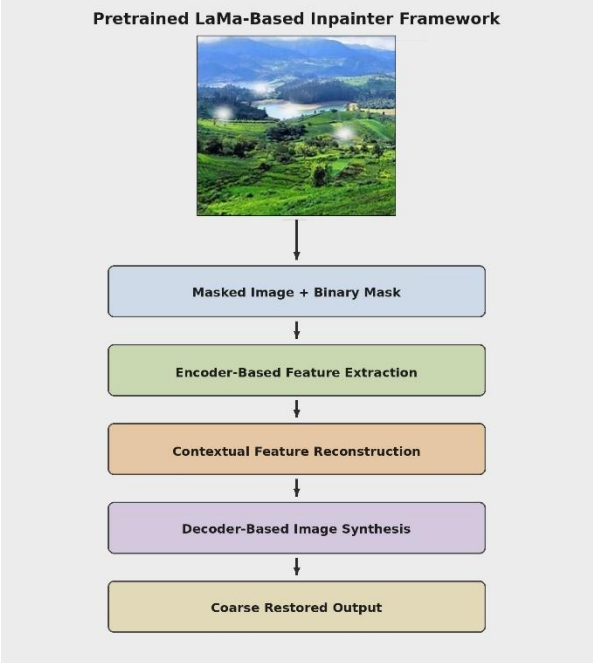

Figure 5. Pretrained LaMa-based inpainter framework

LaMa is kept as the base inpainter. Adapter learning gives the task-specific correction. Bilevel hypergradient optimization improves the adapter update. These steps help the restoration work for real-time defect removal, while the LaMa backbone stays unchanged.

2.6 MABHO: Mask-Aware Bilevel Hypergradient Optimization

Lightweight adapters are added to the pretrained LaMa inpainter. The main LaMa backbone is not trained again. Only the adapter layers are trained for defect removal. This keeps the training load low and still allows small changes

for the damaged region. The adapters mainly learn how to correct the masked part and match it with the nearby image content. MABHO is used for this adapter update. In the inner stage, the adapter learns from the restoration loss on training images. In the outer stage, the same update is checked again using hypergradient values. This helps the adapter update move in a better direction. The complete MABHO pipeline is shown in Figure 6.

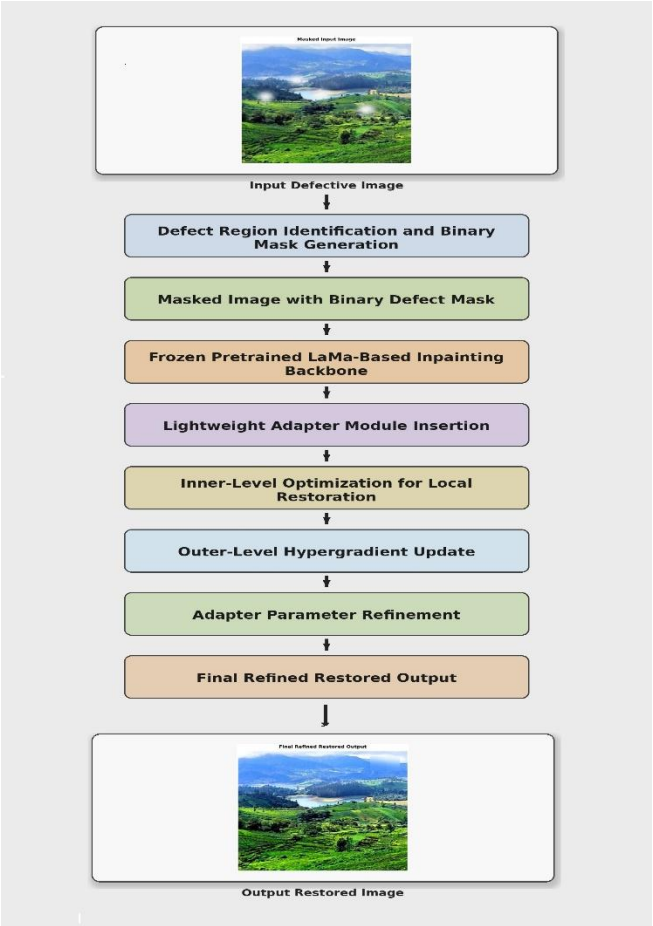

Figure 6. MABHO end-to-end restoration workflow

MABHO checks how the adapter changes after the inner update. Based on this check, the adapter parameters are adjusted again for better defect repair. The hypergradient-based update is written as:

$$\nabla_{\phi} L_{outer} = \frac{\partial L_{outer}}{\partial \phi'} \cdot \frac{\partial \phi'}{\partial \phi}$$

Here, ϕ is the adapter setting before the inner update, and Φ' is the setting after the inner update. This shows how the adapter change affects the final restoration loss. The adapter is then updated using this value.

2.7 Loss Functions and Optimization Objectives

The adapters are trained using restoration loss. This loss checks the difference between the restored image and the clean image. Pixel loss helps recover the masked part with proper intensity and content. SSIM loss helps keep the local structure close to the original image. The total restoration loss is written as:

$$\mathcal{L}_{rest} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{ssim}$$

where $\mathcal{L}_{\text{rec}}$ denotes the reconstruction loss, $\mathcal{L}_{\text{ssim}}$ denotes the structural similarity loss, and λ_1 and λ_2 are weighting factors used to balance their contributions. The inner step trains the adapter with restoration loss. The outer step checks that update and corrects it if needed. This helps the restored image improve without training the full LaMa model again.

2.8 Proposed Algorithm

Algorithm 1 shows the restoration steps. The image is first prepared, and the defect mask is created. The masked image is restored using pretrained LaMa. Then the adapter is adjusted through MABHO for better repair.

Algorithm 1: Proposed real-time defect removal and mask-guided image inpainting framework

Algorithm 1 Proposed restoration process
Input: I, M
Output: I_r
Step 1: Read the input image I.
Step 2: Resize and normalize the image.
Step 3: Generate the defect mask M.
Step 4: Prepare the masked image.
Step 5: Feed the masked image and mask to the pretrained LaMa-based inpainter.
Step 6: Insert adapter-based refinement.
Step 7: Apply inner restoration update.
Step 8: Compute hypergradient-based outer update.
Step 9: Refine adapter parameters using MABHO.
Step 10: Produce restored image I_r.
Step 11: Evaluate the restored output

2.9 Implementation Details and Real-Time Restoration Flow

The implementation is carried out in Python with PyTorch on Google Colab. A pretrained Large Mask Inpainting (LaMa) inpainter is used, and small adapter layers are added to it. During restoration, only the masked defect area is changed, while the remaining part of the image is kept unchanged. Table 2 gives the hyperparameter settings. The restored image is checked using quality, masked-region, and speed measures. The quality measures used are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). The masked-region measures include Masked-PSNR, Masked-SSIM, and Masked Region Restoration Quality Loss (MRQL). Latency and Frames Per Second (FPS) are used to check real-time performance.

Peak Signal-to-Noise Ratio (PSNR): This metric checks the pixel-level difference between the restored image and the reference image. It is defined as:

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right)$$

Here, MAX is the highest pixel value. MSE is the mean squared error between the restored image and the reference image.

Table 2. Hyperparameter settings used in the proposed framework

Parameter	Value	Purpose
Input image size	256 × 256	Uniform model input
Number of training images	300	Main training and 5-fold cross-validation
Number of external test images	150–180	External validation

Number of real-time test images	15	Practical inference evaluation
Number of folds	5	Cross-validation
Batch size	8	Training mini-batch size
Number of epochs	50	Model training
Optimizer	Adam	Parameter update
Learning rate	0.0001	Base optimization step size
Adapter learning rate	0.0001	Adapter refinement
Reconstruction loss weight (\lambda_1)	1	Pixel-level restoration term
SSIM loss weight (\lambda_2)	0.5	Structural consistency term
Early stopping patience	10	Prevent overtraining

Structural Similarity Index Measure (SSIM): SSIM checks how close the restored image structure is to the reference image. It looks at luminance, contrast, and structure. A higher SSIM value means the restored image keeps the structure better. It is given by:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where x and y represent the reference and restored images, μ denotes mean intensity, σdenotes variance, and σ_{xy}denotes covariance.

Learned Perceptual Image Patch Similarity (LPIPS): LPIPS checks visual similarity using deep image features. Pixel scores may miss some visual differences, so LPIPS is also used. A lower LPIPS value means the restored image looks closer to the reference image. It is expressed as:

$$LPIPS = \sum_l \frac{1}{H_l W_l} \sum_{h,w} |w_l \odot (\widehat{y_{hw}^l} - y_{hw}^l)|_2^2$$

where $l$denotes the feature layer, $H_l$and $W_l$denote spatial dimensions, and $w_l$denotes the learned channel weights.

Masked Peak Signal-to-Noise Ratio (Masked-PSNR): Masked-PSNR is calculated only on the defect area. This is used because the model changes only the masked part. A higher value means the repair is better in that area. It is defined as:

$$MPSNR = 10\log_{10}\left(\frac{MAX^2}{MSE_M}\right)$$

where MSE_Mdenotes the mean squared error computed only over the masked region.

Masked Structural Similarity Index Measure (Masked-SSIM): Masked-SSIM checks structure only in the repaired defect area. It compares that part with the same region in the reference image. A higher value means the structure is preserved better. It is written as:

$$MSSIM = SSIM(x_M, y_M)$$

where x_M and y_M denote the masked-region patches from the reference and restored images.

Latency: Latency is the time taken to restore one input image. It shows how fast the framework gives the output. Lower latency means faster restoration. It is computed as:

$$Latency = t_{out} - t_{in}$$

where t_{in} is the input time and t_{out} is the output generation time.

Frames Per Second (FPS): FPS shows how many images are restored in one second. A higher FPS value means the method runs faster and is more suitable for real-time use. It is given by:

$$FPS = \frac{N}{T}$$

where N is the number of processed images and T is the total processing time.

Masked Region Restoration Quality Loss (MRQL) Score: MRQL is used to check the repaired mask region in one combined score. It includes masked-region quality, structure preservation, and visual similarity. It is defined as:

$$MRQL = \alpha \cdot MPSNR_{norm} + \beta \cdot MSSIM + \gamma \cdot (1 - LPIPS)$$

Subject to

$$\alpha + \beta + \gamma = 1$$

where $MPSNR_{norm}$ denotes the normalized Masked-PSNR, $MSSIM$ denotes the Masked-SSIM, and α, β, and γ are weighting coefficients. A higher MRQL score means better repair inside the masked region. These metrics check the output from different sides: pixel match, structure, visual quality, repaired-region quality, and real-time speed.

3. Results and Discussion

3.1 Experimental Setup and Protocol

Training is carried out for 50 epochs with batch size 8. Adam optimizer is used with a learning rate of 0.0001, and early stopping patience is set to 10. Latency and Frames Per Second (FPS) are measured using batch-size-one inference.

3.2 Quantitative performance

Full-image quality is checked using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). The values are shown in Table 3.

Table 3. Full-image restoration performance on the Landscape dataset

Method	Peak Signal-to-Noise Ratio (dB)	Structural Similarity Index Measure	Learned Perceptual Image Patch Similarity
Frozen LaMa baseline	20.96	0.811	0.184
Proposed framework	22.84	0.864	0.129

Masked-Region Evaluation: The defect area is checked separately using only the masked pixels. For this part, Masked Peak Signal-to-Noise Ratio (Masked-PSNR), Masked Structural Similarity Index Measure (Masked-SSIM), and the proposed Masked Region Restoration Quality Loss (MRQL) are used. The results are shown in Table 4.

Table 4. Masked-region restoration performance on the Landscape dataset

Method	Masked Peak Signal-to-Noise Ratio (dB)	Masked Structural Similarity Index Measure	MRQL Score
Frozen LaMa baseline	17.84	0.746	0.781
Proposed framework	20.72	0.823	0.884

The total loss starts at about 1.82 in the first epoch and reaches 0.48 by epoch 50, as shown in Figure 7. It drops fast at first. Later, the decrease becomes slow and steady. The curve does not show major jumps, so the training remains stable.

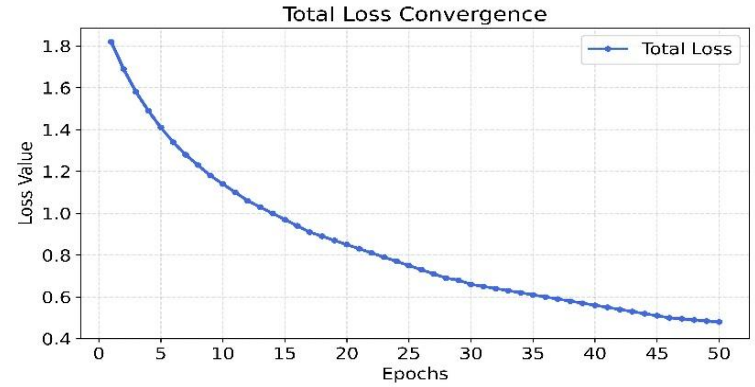

Figure 7. Total training loss convergence of the proposed MABHO framework across 50 epochs

Pixel Loss and Structural Loss Behaviour: The training pixel loss starts at about 1.08 in the first epoch and reaches 0.18 by epoch 50, as shown in Figure 8. It drops quickly in the early epochs. Later, the curve becomes slow and steady. This shows stable pixel-level reconstruction without large changes in the loss.

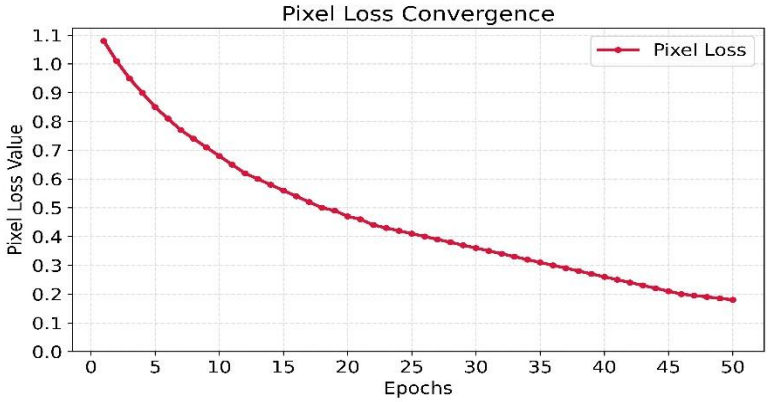

Figure 8. Training pixel loss convergence of the proposed MABHO framework across 50 epochs.

The training structural loss starts at about 0.74 in the first epoch and reaches 0.22 by epoch 50, as shown in Figure 9. The curve decreases slowly and stays stable during training. This shows steady improvement in structural restoration.

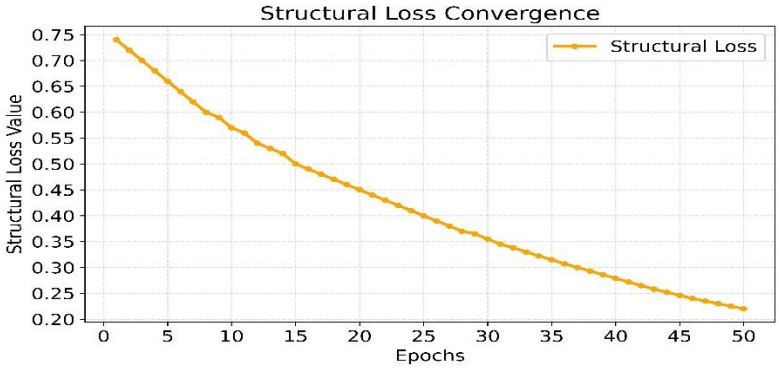

Figure 9. Training structural loss convergence of the proposed

MABHO framework across 50 epochs

3.3 Qualitative Performance

Figure 10 shows the restoration steps. It starts with the masked input and defect mask. The frozen LaMa model gives the first restored output. Adapter insertion and inner-outer optimization are then applied. After the adapter parameters are updated, the final restored image is generated. This stepwise view shows how MABHO improves the initial LaMa output.

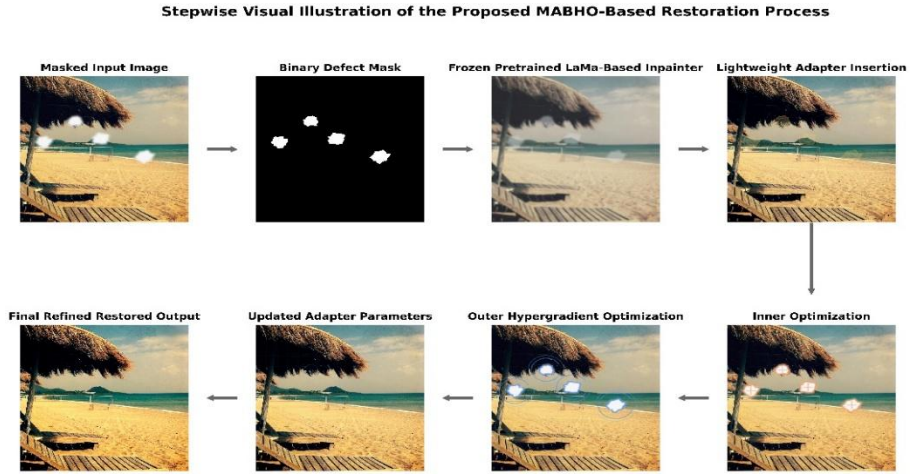

Figure 10. Stepwise MABHO-based restoration from masked input to final output

Multi-Sample Visual Comparison: Figure 11 shows sample Landscape results. Each row shows the masked image, mask, frozen LaMa result, and final output. The frozen LaMa output looks rough in some places and too smooth in others. The proposed output keeps the texture closer to the nearby area. The colour also matches better, and the repaired part looks less visible.

Failure Cases and Error Analysis: Some samples still do not restore perfectly, as shown in Figure 12. This happens mainly when the mask is large, the texture repeats, the lighting changes, or the boundary is weak. In Case 1, the mask covers a large structural part, so the model has less nearby context to use. In Case 2, repeated texture makes the repair confusing. In Case 3, lighting change and weak boundary detail make blending harder. These cases show that the result depends on mask size, texture pattern, and local image appearance.

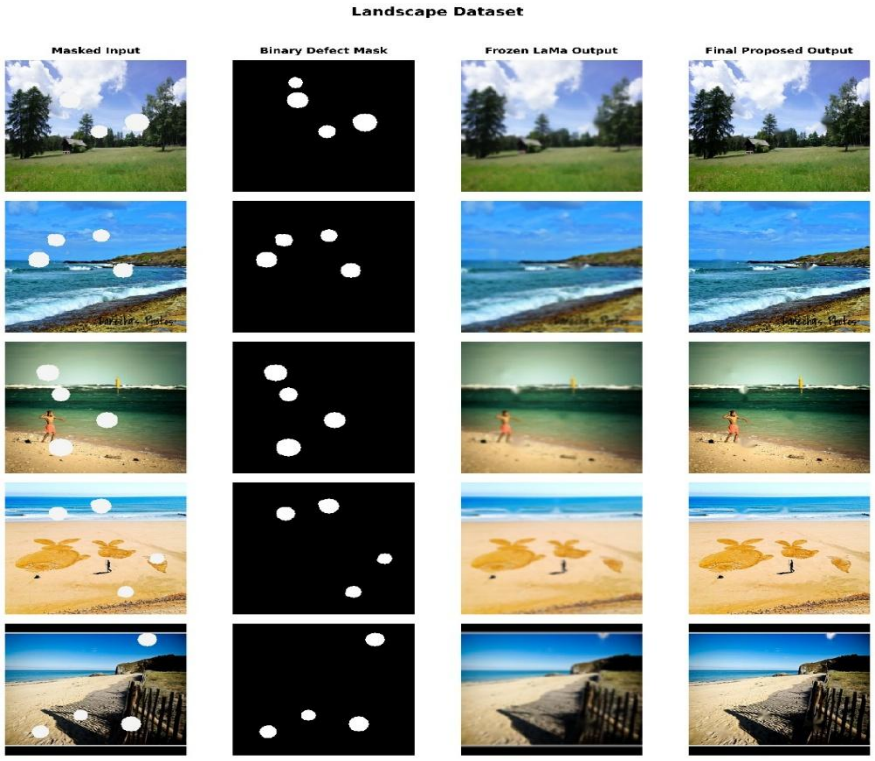

Figure 11. Multi-sample qualitative comparison on Landscape dataset images

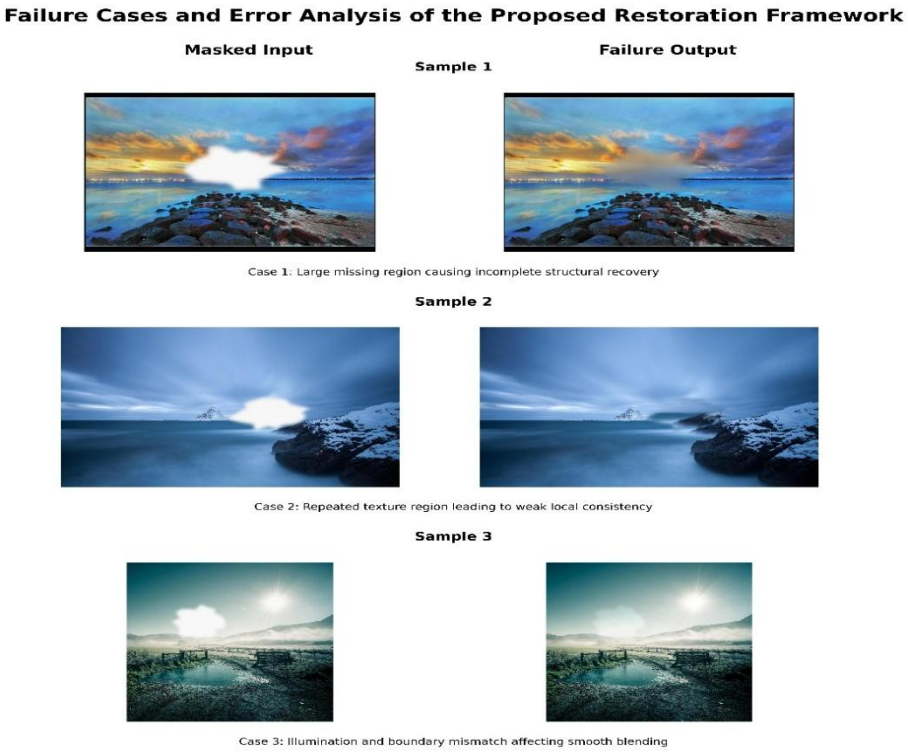

Figure 12. Failure cases of the proposed restoration framework

3.4 Ablation Study

The ablation study checks how each part affects the result. Table 5 compares the frozen LaMa baseline, adapter use, bilevel optimization, and the full framework. The baseline gives weaker repair in the masked region. The adapter improves the local repair, and bilevel optimization improves it further. The full framework gives the best masked-region result with only a small latency increase. Figure 13 shows this improvement step by step.

Table 5. Ablation study of the proposed framework on the Landscape dataset

Variant	Masked-PSNR (dB)	Masked-SSIM	LPIPS	MRQL	Latency (ms)	FPS
Frozen LaMa baseline	17.84	0.746	0.184	0.781	38.4	26
LaMa + Adapter	18.91	0.782	0.161	0.828	42.7	23.4
LaMa + Adapter + Bilevel Optimization	20.11	0.811	0.141	0.866	47.9	20.9
Proposed full framework	20.72	0.823	0.129	0.884	49.6	20.2

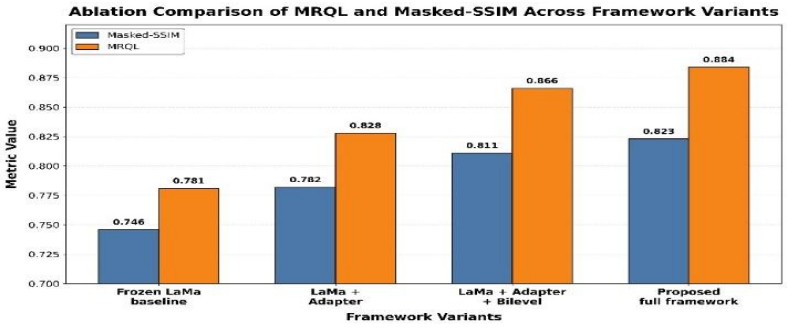

Figure 13. Ablation comparison of MRQL and Masked-SSIM across framework variants

3.5 Cross-Validation and External Validation

5-Fold Cross-Validation: The framework is tested using 5-fold cross-validation on the Landscape dataset. Table 6 shows that the scores are almost stable across the five folds. The average values are 22.84 ± 0.13 dB PSNR, 0.864 ± 0.005 SSIM, 0.129 ± 0.004 LPIPS, 20.72 ± 0.17 dB Masked-PSNR, 0.823 ± 0.007 Masked-SSIM, and 0.884 ± 0.008 MRQL. Figure 14 shows that MRQL and stays steady across the five folds.

Table 6. Fold-wise 5-fold cross-validation results on the Landscape dataset

Fold	PSNR (dB)	SSIM	LPIPS	Masked-PSNR (dB)	Masked-SSIM	MRQL
Fold 1	22.66	0.857	0.135	20.5	0.814	0.874
Fold 2	22.78	0.861	0.131	20.64	0.819	0.88
Fold 3	22.84	0.864	0.129	20.72	0.823	0.884
Fold 4	22.9	0.867	0.127	20.8	0.827	0.888
Fold 5	23.02	0.871	0.123	20.94	0.832	0.894

Mean ± Std	**22.84 ± 0.13**	**0.864 ± 0.005**	**0.129 ± 0.004**	**20.72 ± 0.17**	**0.823 ± 0.007**	**0.884 ± 0.008**

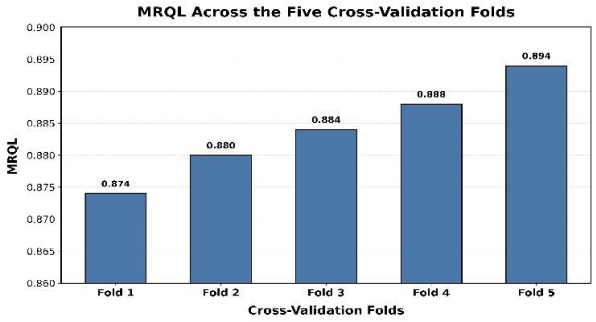

Figure 14. MRQL across five cross-validation folds

External Validation: External validation is done on 150–180 unseen Intel images from Kaggle. These images are not used for training or cross-validation. Table 7 shows that the proposed method performs better than the frozen LaMa baseline. It gets 21.94 dB PSNR, 0.848 SSIM, and 0.138 LPIPS. The baseline gets 20.21 dB PSNR, 0.796 SSIM, and 0.192 LPIPS. For the masked region, the scores improve from 17.08 dB to 19.86 dB in Masked-PSNR, from 0.731 to 0.807 in Masked-SSIM, and from 0.768 to 0.867 in MRQL. This shows that the method also works on unseen images.

Table 7. External validation results on the Intel dataset

Method	PSNR (dB)	SSIM	LPIPS	Masked-PSNR (dB)	Masked-SSIM	MRQL
Frozen LaMa baseline	20.21	0.796	0.192	17.08	0.731	0.768
Proposed framework	21.94	0.848	0.138	19.86	0.807	0.867

Figure 15 shows the visual results on unseen Intel images. The frozen LaMa output is too smooth in some repaired areas. The proposed output keeps the scene structure better. The repaired part also blends more naturally with the nearby image content.

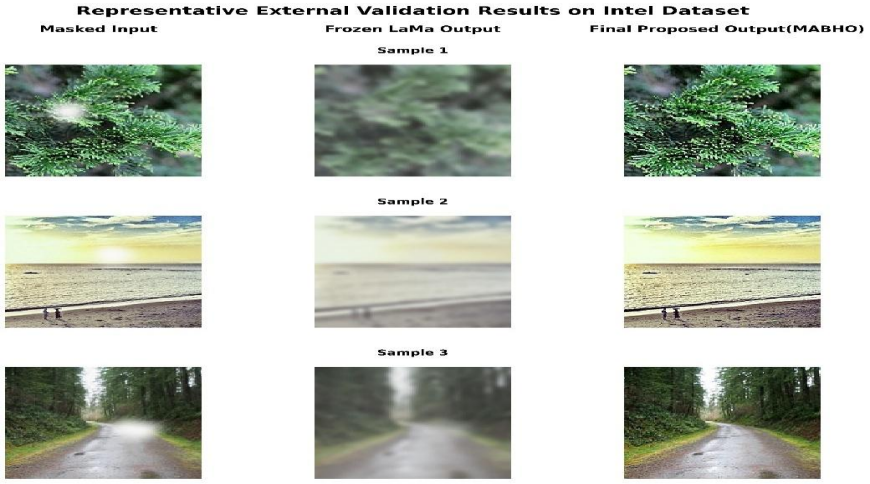

Figure 15. Representative qualitative results on unseen Intel images.

Real-time testing uses 15 camera-captured images. These images are used only for inference with batch size one. Table 8 shows that the proposed framework takes 49.6 ± 2.4 ms and gives 20.2 ± 1.0 FPS. The frozen LaMa baseline takes 36.8 ± 1.9 ms and gives 27.2 ± 1.4 FPS. The proposed method is a little slower, but the repaired area looks better in camera-captured images, as shown in Figure 16.

Table 8. Real-time inference performance on 15 camera-captured images

Method	Latency (ms)	FPS
Frozen LaMa baseline	36.8 ± 1.9	27.2 ± 1.4
Proposed framework	49.6 ± 2.4	20.2 ± 1.0

Figure 16. Representative real-time qualitative results on camera-captured images

Comparative Benchmark Analysis: The proposed framework is compared with five recent inpainting methods: HINT [20], MMGInpainting [21], MATdiff [24], adaptive prior and long-range dependency-based learning [25], and hierarchical texture-aware inpainting [19]. These methods use different ideas such as transformers, diffusion guidance, long-range context, and texture-aware repair. This comparison is used only as a reference because the datasets, masks, and protocols are different. Table 9 gives the reference comparison. On the Landscape dataset, the proposed framework records 22.84 dB PSNR, 0.864 SSIM, 0.129 LPIPS, 20.72 dB Masked-PSNR, 0.823 Masked-SSIM, and 0.884 MRQL. These scores show that the proposed method gives good repair quality, mainly for the damaged region and real-time defect correction.

Table 9. Benchmark comparison with recent image inpainting methods

Ref.	Method	Year	Dataset / setting	PSNR	SSIM	LPIPS	Other reported result	Runtime
[20]	HINT	2024	CelebA-HQ, large masks (40–60%)	24.1287	0.8241	0.1449	FID = 5.6179	125 ms/image
[21]	MMG Inpainting	2024	Multi-modality guided diffusion inpainting	—	—	—	Diffusion-based multimodal guidance for semantic completion	Reported as faster than prior diffusion-style designs in accessible summary
[24]	MATdiff	2026	CelebA-HQ, large-mask inpainting	—	—	—	Improves over MAT by 5.94% in FID, 3.13% in LPIPS, 18.75% in P-IDS, and 5.14% in U-IDS	—
[25]	Adaptive prior and long-range dependency-based learners	2025	Image inpainting benchmarks	—	—	—	Adaptive prior learning with long-range dependency modeling	—
[19]	Hierarchical texture-aware image	2026	Texture-aware image inpainting	—	—	—	Texture-aware restoration with	—

	inpainting via contextual attention and multiscale feature fusion						contextual attention and multiscale fusion	
Proposed	Pretrained LaMa + adapters + bilevel hypergradient optimization	2026	Landscape dataset	22.84	0.864	0.129	Masked-PSNR = 20.72, Masked-SSIM = 0.823, MRQL = 0.884	49.6 ms, 20.2 FPS

3.6 Discussion

The analysis shows a few limits in the present framework. The result depends on the mask boundary. If the mask is too narrow, small defect traces may remain. If the mask is too large, the repaired area may become slightly smooth. Similar mask-related issues are seen in MagConv [7] and HINT [20]. Large damaged regions, repeated textures, and thin structures are also difficult to restore. These cases may cause texture mismatch, weak edges, or less natural blending. Similar problems are seen in texture-aware and transformer-diffusion methods [19], [24]. MABHO needs a proper balance between pixel loss and structural loss. A poor balance may make the output too smooth or create small artifacts near irregular masks. Related local and global trade-offs are discussed in context-adaptive and dependency-aware methods [8], [25]. The study also has dataset limits. Training uses a limited number of Landscape images, and the camera-captured test set is small. Intel and camera-captured images also differ from Landscape images in lighting, texture, and defect patterns. This makes cross-dataset testing harder. Similar domain sensitivity is noted in multimodal and diffusion-guided methods [21]. These limits show that the framework is mainly affected by mask boundaries, difficult scene content, loss balance, and dataset variation.

4. Conclusions

This work proposes a real-time defect removal framework for live-captured images. It uses mask-guided inpainting with pretrained Large Mask Inpainting (LaMa), adapter refinement, and Mask-Aware Bilevel Hypergradient Optimization (MABHO). The method repairs the damaged region and keeps the remaining image content unchanged. On the Landscape dataset, the method gives 22.84 dB PSNR, 0.864 SSIM, and 0.129 LPIPS. For the masked region, it gives 20.72 dB Masked-PSNR, 0.823 Masked-SSIM, and 0.884 MRQL. The 5-fold cross-validation results stay close across the folds. The Intel image test shows that the method works on unseen images. The camera-captured test also shows that the method can run with batch-size-one inference. The method gives better repair in the defect region and keeps the image stable. It still has difficulty with mask quality, large missing areas, texture changes, and computation time. Future work can improve these parts and make the method more useful for wider real-image use.

5. Acknowledgements

The authors thank the Department of Studies and Research in Computer Applications, Tumkur University, for providing the necessary support for this work. The first author acknowledges guidance and support received during the course of the Ph.D. research.

Author Contributions: Suma N carried out the conceptualization, methodology, software development, experimentation, data curation, analysis, visualization, and original draft writing. Kusuma Kumari B.M. contributed to the validation, supervision, review, and editing of the manuscript, as well as overall project guidance. Both authors read and approved the final manuscript.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest

References

1. Elharrouss, S.; Almaadeed, N.; Al-Maadeed, S.; Akbari, Y. Image inpainting: A review. Neural Processing Letters 2020, 50, 2003–2028. https://doi.org/10.1007/s11063-019-10163-0.
2. Zhang, J.; et al. Image inpainting based on deep learning: A review. Information Fusion 2023, 97, 101860. https://doi.org/10.1016/j.inffus.2022.08.033.
3. Xiang, Y.; et al. Deep learning for image inpainting: A survey. Pattern Recognition 2023, 140, 109537. https://doi.org/10.1016/j.patcog.2022.109537.
4. Quan, W.; et al. Deep learning-based image and video inpainting: A survey. International Journal of Computer Vision 2024, 132, 1580–1615. https://doi.org/10.1007/s11263-023-01977-6.
5. Huang, Y.; et al. Diffusion model-based image editing: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 2025, 47, 4409–4437. https://doi.org/10.1109/TPAMI.2025.3541625.
6. Quan, W.; Tan, R.; Huang, H.; Liao, J.; Yi, J.-H.; Zhang, L. Image inpainting with local and global refinement. IEEE Transactions on Image Processing 2022, 31, 2405–2420. https://doi.org/10.1109/TIP.2022.3152624.
7. Yu, Z.; Hou, J.; Chen, M.; Wang, J. MagConv: Mask-guided convolution for image inpainting. IEEE Transactions on Image Processing 2023, 32, 4716–4727. https://doi.org/10.1109/TIP.2023.3298536.
8. Deng, J.; et al. Context adaptive network for image inpainting. IEEE Transactions on Image Processing 2023, 32, 6332–6345. https://doi.org/10.1109/TIP.2023.3298560.
9. Wang, X.; et al. Image inpainting based on panoramic feature aggregation and convolution optimization. Knowledge-Based Systems 2024, 286, 111382. https://doi.org/10.1016/j.knosys.2024.111382.
10. Chen, X.; et al. A two-stage network with inference attention module for image inpainting. Engineering Applications of Artificial Intelligence 2024, 138, 109181. https://doi.org/10.1016/j.engappai.2024.109181.
11. Zhao, Y.; et al. An image inpainting method based on contextual attention gated convolution and mask-aware module. Displays 2024, 92, 102945. https://doi.org/10.1016/j.displa.2024.102945.
12. Dong, S.; Dou, Y. AU-GAN: An adaptive U-Net GAN model for image inpainting. Multimedia Tools and Applications 2024, 83, 41827–41849. https://doi.org/10.1007/s00530-024-01290-3.
13. Huang, J.; Huang, J. DRGAN: Dual-resolution guided generative adversarial network for low-resolution image inpainting. Knowledge-Based Systems 2023, 264, 110346. https://doi.org/10.1016/j.knosys.2023.110346.
14. Wang, S.; et al. Diverse image inpainting with disentangled uncertainty estimation. Pattern Recognition 2023, 137, 109243. https://doi.org/10.1016/j.patcog.2022.109243.
15. Huang, X.; et al. Sparse self-attention transformer for image inpainting. Pattern Recognition 2024, 145, 109897. https://doi.org/10.1016/j.patcog.2023.109897.
16. Yao, W.; et al. Image inpainting based on bidirectional information flow and multi-dimensional feature learning of texture and structure. Signal Processing 2025, 226, 109672. https://doi.org/10.1016/j.sigpro.2024.109672.
17. Chen, C.; et al. CTSTNet: Coarse-to-fine structure and texture learning for image inpainting based on neural network. Applied Soft Computing 2024, 170, 112671. https://doi.org/10.1016/j.asoc.2024.112671.
18. Chen, Z.; et al. Image inpainting based on dual degradation feature and adaptive feature fusion U-net. Applied Soft Computing 2025, 177, 113010. https://doi.org/10.1016/j.asoc.2025.113010.
19. Li, Y.; et al. Hierarchical texture-aware image inpainting via contextual attention and multiscale feature fusion. Image and Vision Computing 2026, 159, 105875. https://doi.org/10.1016/j.imavis.2025.105875.
20. Chen, S.; et al. HINT: High-quality INpainting Transformer with mask-aware encoding and enhanced attention. IEEE Transactions on Multimedia 2024, 26, 7649–7660. https://doi.org/10.1109/TMM.2024.3369897.
21. Zhang, C.; et al. MMGInpainting: Multi-modality guided image inpainting based on diffusion models. IEEE Transactions on Multimedia 2024, 26, 6548–6562. https://doi.org/10.1109/TMM.2024.3382484.
22. Liu, C.; Xu, S.; Peng, J.; Zhang, K.; Liu, D. Toward interactive image inpainting via robust sketch refinement. IEEE Transactions on Multimedia 2024, 26, 9973–9987. https://doi.org/10.1109/TMM.2024.3402620.
23. Liu, T.; et al. Transref: Multi-scale reference embedding transformer for reference-guided image inpainting. Neurocomputing 2025, 632, 129749. https://doi.org/10.1016/j.neucom.2025.129749.
24. Wang, S.; et al. MATdiff: Mask-aware transformer with diffusion model for large-mask image inpainting. Neurocomputing 2026, in press, 133042. https://doi.org/10.1016/j.neucom.2026.133042.
25. Cao, F.; Xu, Q.; Ye, H. Adaptive prior and long-range dependency-based learners for image inpainting. IEEE Transactions on Circuits and Systems for Video Technology 2025, 35, 10742–10755. https://doi.org/10.1109/TCSVT.2025.3574529.