Image Transformation, Encryption, and Recovery

Student Name and ID: [REDACTED]

Abstract

Image blur and noise are common degradations in imaging systems, and recovering a sharp image from a blurred one is a classical unstable inverse problem. As, in practice, perfect images do not exist and all imaging systems cause degradation, oftentimes we require these techniques to be able to perform image interpretation. Other times, with the increased valuing of security and privacy in this digital age, we want to make sure we can obfuscate sensitive details in an image, so those details can only be recovered if one has the private key. In this project, we study RGB image recovery under both standard convolution-based "transformations" (Gaussian, motion, box blur) and "private-key" transformations (also known as "encryptions" in this project), where the blur kernel is generated from a secret random seed or derived from a key image. We implement the forward model as a color channel-wise convolution computed using the FFT, add Gaussian noise, and reconstruct using Tikhonov-regularized deconvolution in the frequency domain.

Across three images with different structures (vector-like graphics, a comic scan, and a real-life photo with artificial vector-like annotation), correct-kernel recovery achieves significantly higher PSNR and lower MSE than mismatched-kernel recovery for private-key kernels—with PSNR roughly 2.5dB higher for our random seed kernel, and 5dB higher for our key image kernel. This demonstrates strong sensitivity to kernel knowledge. Finally, we observe that metric scores such as MSE and PSNR do not always align with perceptual importance: small high-frequency text can be visually unrecoverable even when global error remains low, which highlights the limitations of purely quantitative evaluation.

Introduction

Problem Statement and Context

Image degradation is an unavoidable physical phenomenon when operating real-world imaging systems. Motion during exposure, imperfect focus, lens aberrations, and sensor noise, all contribute to blurring and distortion of captured images. Hence, recovering an original sharp image from a degraded observation is key in image processing and is thus essential for real-life applications such as computer vision, photography, medical imaging, and scientific analysis.

On the other hand, with the increased value for privacy, we require more and more ways to store data securely, with storing images being no exception [Luo+24]. In this project, we focus on the most general form of tackling with this issue, which is what we will call "image encryption". We use this term to refer to image transformations with a private key, such that without the private key, it is impossible to recover (which we use interchangably with "decrypt" for encrypted images) the image, but with the private key, it could recover a sharp image. In this case, the goal is focused on maximizing the decryption error, which we shall call "privacy", between those without the private key and those with the private key.

Mathematically, image degradation is often modeled as a convolution of an image with a blur kernel, followed by adding noise. Although this forward process is well-defined, the inverse process is highly unstable especially with noise, since small perturbations in the observed image can lead to large errors in reconstruction [GW18]. In this project, we leverage this instability both as a challenge for reconstruction and as a mechanism for privacy.

Background and Theoretical Framework

As previously mentioned, image degradation is usually modeled as a linear convolution of an underlying image with a blur kernel, followed by additive noise. Let x be the original image and y be the observed degraded image. A standard forward model is

$$y = Kx + \eta,$$

where K is a convolution operator and η represents noise.

For a spatially invariant blur, the operator K acts via convolution with a kernel k like so:

$$(Kx)(i,j) = \sum_{u,v} k(u,v)\, x(i-u, j-v).$$

Such models capture physical effects, such as motion blur.

If we include a third parameter to refer to the color channel c (i.e. Red, Green, or Blue), then the 2D convolution of an image channel x_c with kernel k_c is

$$(K_c x_c)(i,j) = \sum_{u=-r}^{r} \sum_{v=-r}^{r} k_c(u,v)\, x_c(i-u,\ j-v),$$

with indices handled by padding or circular wrapping in the code.

For a kernel that is the same across channels, we can write $k(u,v)$ and

$$(Kx)(i,j,c) = \sum_{u,v} k(u,v)\, x(i-u, j-v, c).$$

Thus, recovering x from y is an inverse problem. In the presence of noise, this problem is unstable—direct inversion of K amplifies noise and leads to unstable reconstructions. To address this instability, we must consider regularization techniques.

Tikhonov Regularization

In this project, we use Tikhonov regularization and solve

$$\hat{x} = \arg\min_x \|Kx - y\|^2 + \lambda \|x\|^2,$$

where λ controls the trade-off between reconstruction accuracy and stability [TA77].

We choose Tikhonov regularization specifically because it is deterministic, easy to implement, yet allows for significant tunability with λ. For convolution operators, the convolution theorem allows for both the forward and inverse problems to be efficiently computed in the frequency domain, using the Fast Fourier Transform (FFT). Numerically, there is a method proposed in [CT65], which allows for such computation to be done efficiently. Thus, the combination of Tikhonov regularization and FFT, renders a viable approach towards image recovery via numerical computation, for our project.

Key Insight for Image Encryption

Moreover, in this project, this theoretical framework can also be used to study image encryption. By treating the blur kernel as a private key, the same convolutional model can be seen as both a degradation and reversible transformation. When the correct kernel is known, regularized inversion allows for partial recovery of the original image; when the kernel is unknown or incorrect, inversion fails catastrophically due to the instability of the problem from noise. **This dual interpretation allows us to study image recovery and privacy within one unified mathematical framework, where instability becomes a duality for us, both as a challenge for reconstruction and a mechanism for enforcing privacy.**

Review of Existing Literature

As referenced throughout previous sections, image degradation and recovery have been extensively studied under the context of inverse problems in image processing. A large body of work focuses on stabilizing unstable deconvolution problems via regularization, with classical approaches emphasizing mathematical well-posedness, numerical stability, and reconstruction fidelity [GW18; TA77]. These works establish a strong theoretical foundation for image recovery under known degradation models.

Classical regularization-based approaches, such as Tikhonov regularization, remain widely used due to their mathematical simplicity and analytical robustness [TA77]. When combined with frequency-domain methods by the FFT, these techniques allow for efficient computation of both forward degradation and inverse reconstruction [CT65]. Thus, FFT-based deconvolution has become a standard tool in large-scale image processing, especially in scenarios where we emphasize computational efficiency.

Metrics for image reconstruction quality have also gradually changed over the years. Traditional metrics such as Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) are commonly used due to their simplicity and interpretability [GW18]. However, prior studies have shown these pixel-wise error metrics do not always correlate with human visual perception. This motivated perceptually informed measures such as the Structural Similarity Index (SSIM), which better capture structural distortions in images [Wan+04]. These findings highlight an important gap between numerical reconstruction accuracy and perceptual accuracy. In this project, we would still mainly focus on MSE and PSNR, to emphasize quantitative sensitivity to kernel mismatch, rather than perceptual similarity, since it is the more objective measure for privacy.

Alongside image restoration research, recently there is also increased focus on privacy-preserving image processing and image encryption techniques. Many of these methods rely on key-dependent transformations that aim to protect sensitive visual information, while still allowing controlled recovery [Luo+24]. These methods mostly focus on security guarantees and privacy preservation, yet often treat encryption and recovery as completely separate objectives rather than as the duality of a shared inverse problem.

Research Gap and Rationale

Yet, despite extensive research on image deblurring, regularization, and image encryption individually, little work has explored these topics under a unified mathematical framework. In particular, the framework of convolution-based degradation models as private-key transformations, where instability enables both recoverability for authorized users and failure for unauthorized ones, has received limited systematic study. All in all, this motivates the present work for this project.

Research Objectives and Hypotheses

Research Objectives

The primary objectives of this project are as follows:

1. **Implement a convolution-based image degradation and recovery pipeline** using FFT-based forward modeling and Tikhonov-regularized deconvolution.
2. **Compare image recovery performance** under standard blur kernels and private-key kernels.
3. **Quantitatively evaluate the sensitivity of image recovery** to kernel mismatch using MSE and PSNR.
4. Investigate **how instability in inverse problems can simultaneously enable recoverability for authorized users and privacy against unauthorized recovery.**

Hypotheses

Based on inverse problem theory and prior literature, we hypothesize that:

1. Correct-kernel deconvolution will yield significantly lower MSE and higher PSNR than deconvolution performed with an incorrect kernel.
2. In our experiments, kernel mismatch will lead to substantially worse reconstruction for private-key transformations, with the key-image kernel exhibiting the largest degradation among the tested kernels. This is because, it is the kernel with the most unpredictable pattern.
3. Quantitative metrics such as MSE and PSNR will not always reflect perceptual recoverability, particularly for small, high-frequency structures.

Methods

Research Design

This project studies how image reconstruction behaves under different convolution-based transformations, including both standard (public) kernels and encrypted private-key kernels. We evaluate reconstruction accuracy when the correct kernel is known, and investigate the sensitivity of the inverse problem by testing recovery with intentionally incorrect kernels. We do not perform blind kernel estimation; all recovery assumes access to either the true kernel or a deliberately mismatched one.

Repeat the following kernels for all images. We consider Red, Green, and Blue as three separate channels needed for kernels and do not do grayscale, for better visual differentiation.

Baseline: “Transformations” (No Private-Key Convolutions)

- Gaussian Blur
- Motion Blur
- Box Blur

New: "Encryptions" (Private-Key Convolutions)

- Random Kernel from a Seed (seed = private-key), of course fix the seed across all three images
- Private Key Image-derived Kernel via `private_key.png`

Data Sources

Images Used

The data used in this project consists of three RGB images with distinct structural characteristics:

- `antimeme.png`: A vector-like, clean image with large flat color regions and strong contrast. This makes it easier to visually assess blur and regularization artifacts, although the small "THIS IS FINE" text is very sensitive to blur.
- `save_me.png`: A real-life photo with natural lighting, surface texture, sensor noise, and overlaid Chinese text plus artificial purple scribble. These factors introduce mixed frequencies and uneven illumination.
- `this_is_fine.png`: A comic drawing that is not vectorized and contains bold ink lines, but also visible grain and scan noise as mixed textures. It has more texture than `antimeme.png` but cleaner than the real photo `save_me.png`.

Finally, our private key (`private_key.png`) is a real-life photo, which means the natural lighting may create noise. As the photo subject is mainly green, this will ensure the kernels for Red, Green, and Blue will be different. All in all, these factors should make it a less predictable kernel. However, it shouldn't be too unpredictable to the point where recovery is hard.

All images are resized to 512×512 pixels and normalized to the range $[0, 1]$. Convolution and recovery are applied independently to each RGB channel.

For each image and kernel configuration, both quantitative and qualitative data are recorded, with details for each type of data shown below.

Figure 1: All three data source images and the private key image

Quantitative Data

- **Image ID**: `antimeme`, `this_is_fine`, `save_me`
- **Kernel type**: `Gaussian`, `Motion`, `Box`, `RandomSeed`, `KeyImage`
- **Kernel parameters**: kernel size; σ for Gaussian blur; length and angle for motion blur; random seed; or key image used
- **Noise level** σ for added Gaussian noise
- **Regularization parameter** λ
- **Mean Squared Error (MSE)** between the original and recovered image (aggregated over RGB channels)
- **Peak Signal-to-Noise Ratio (PSNR)** derived from MSE

Qualitative Data

- **Image comparisons**: original image, transformed/encrypted image, noisy observation, and recovered image
- Observed **visual artifacts**, including ringing around edges, halo artifacts, oversmoothing, color shifts, ghosting effects, or complete reconstruction failure

Procedures

1. Preprocessing

Each image (`antimeme.png`, `this_is_fine.png`, `save_me.png`) is loaded as an RGB array of size $512 \times 512 \times 3$ and normalized to $[0, 1]$. No grayscale conversion is performed; convolution is applied independently to each color channel.

2. Forward Model

For each image and kernel type, convolution is applied channel-wise to form

$$y = Kx.$$

Additive Gaussian noise is then applied:

$$z = y + \text{noise},$$

to make the inverse problem realistic.

3. Recovery (Inverse Problem)

For each baseline or private-key kernel, non-blind deconvolution is performed using a regularized inverse formulation. To demonstrate key-dependence for encryption experiments, recovery is also attempted using an intentionally incorrect kernel (here, a Gaussian kernel instead of the private kernel). No blind kernel estimation is performed.

Data Analysis

Data analysis is performed by comparing reconstruction outcomes across kernel types, image types, and recovery conditions. Quantitative evaluation is conducted using Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), for the original image and the recovered image over all RGB channels.

For each image, reconstruction performance is analyzed by comparing:

- Correct-kernel recovery versus incorrect-kernel recovery,
- Standard blur kernels versus private-key kernels,
- Differences in sensitivity across kernel types and image structures.

Quantitative results are aggregated and visualized using summary plots to highlight performance gaps between kernel-matched and kernel-mismatched reconstructions. On the other hand, qualitative analysis is performed through visual comparison of recovered images to identify artifacts such as ringing, oversmoothing, color distortion, and complete reconstruction failure. These qualitative observations are used to contextualize quantitative metrics and to highlight cases where numerical error does not fully reflect perceptual degradation.

Ethical Considerations

All the images used in this project except for `this_is_fine.png` are solely my own, and `this_is_fine.png` is used under educational fair use for non-commercial analysis.

Results

The raw results of all the images' transformation/encryption and recovery/decryption are available in **Appendix A** for reference.

Quantitative Results

We summarize recovery quality using Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR), computed between the original image and the recovered image (aggregated over RGB channels). For private-key experiments, we report both **correct-key decryption** and **wrong-key decryption**.

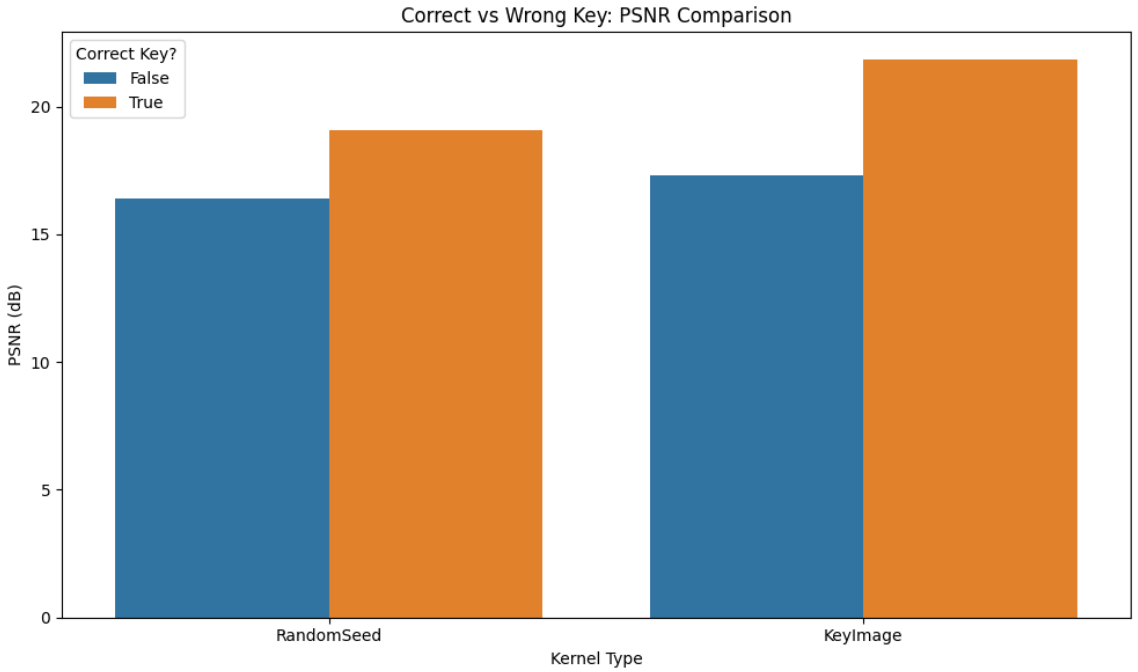


Figure 2: Average PSNR for private-key kernels under correct-key vs. wrong-key decryption. Correct-key recovery is consistently higher.

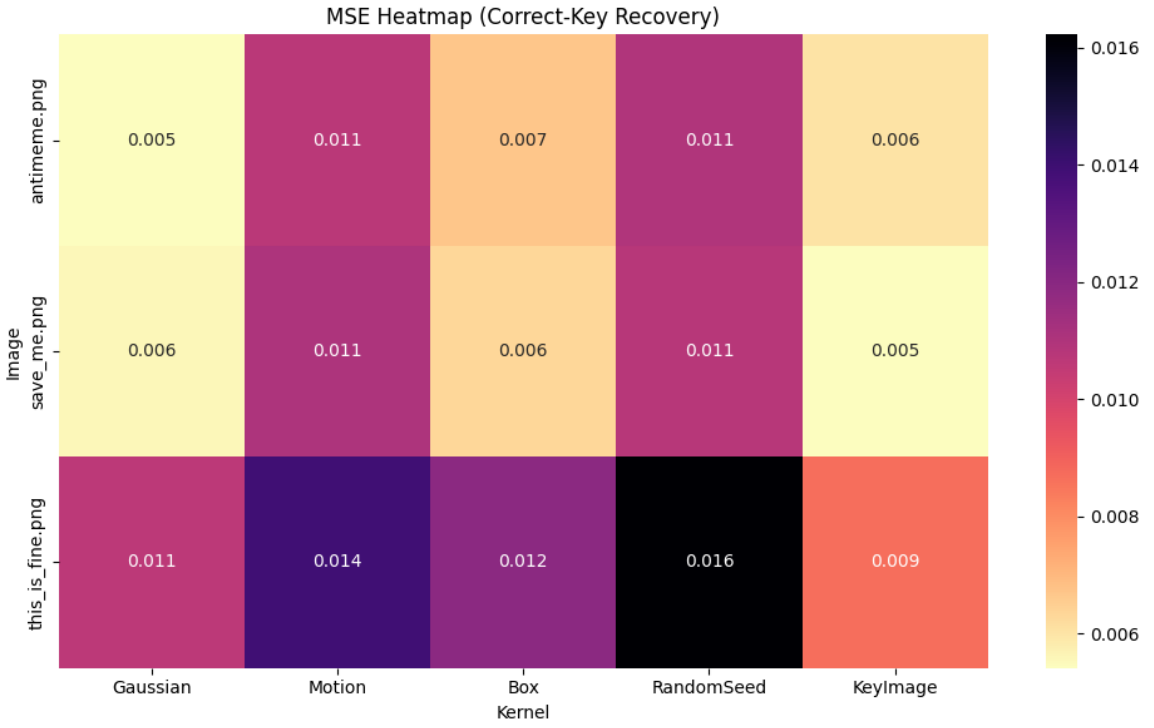


Figure 3: MSE heatmap for correct-kernel recovery across images and kernels. Lower is better.

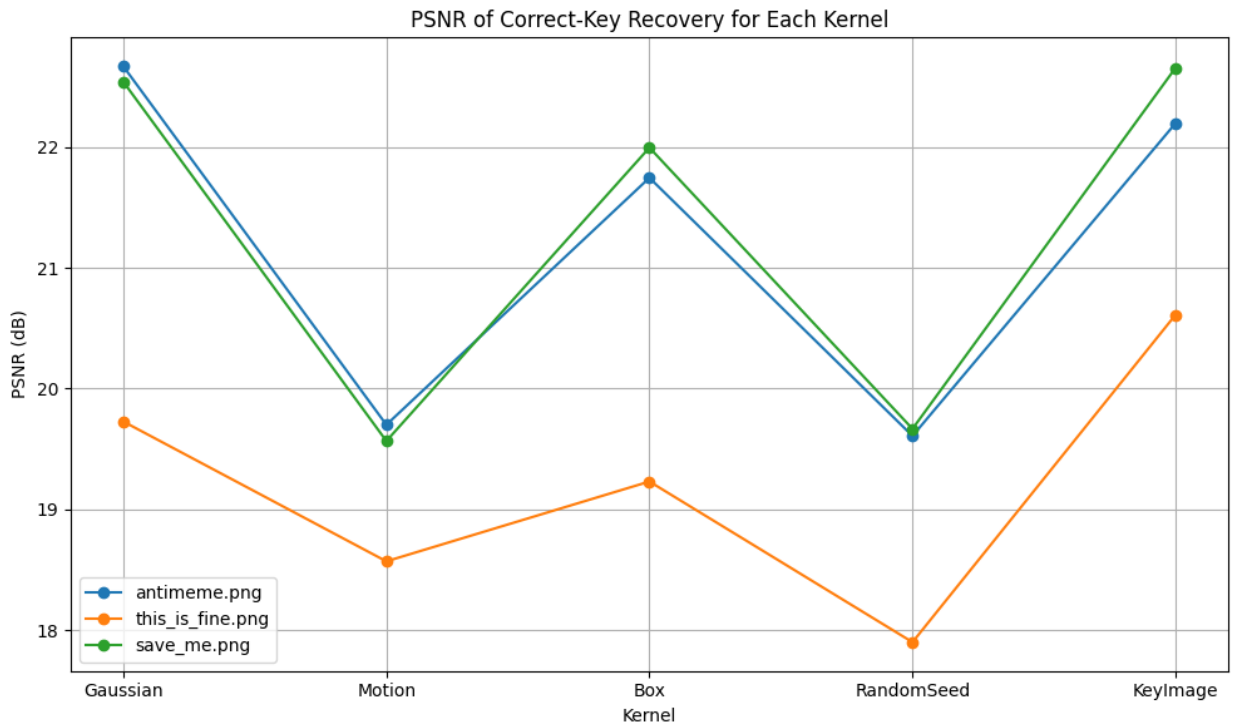


Figure 4: PSNR (correct-kernel recovery) per image across kernel types.

Key-mismatch gap (Privacy effect). Figure 2 shows a clear separation between correct-key and wrong-key recovery for both private kernels. For the `RandomSeed` kernel, correct-key PSNR is roughly 2.5 dB higher than wrong-key recovery. For the `KeyImage` kernel, the gap increases to roughly 5 dB, indicating stronger sensitivity to key knowledge.

Across-kernel difficulty. From Figure 4 (correct-kernel recovery), Gaussian blur is typically easier to invert than box blur, which is easier to invert than motion blur, and the `KeyImage` kernel achieves the strongest PSNR among the tested kernels for most images. Motion blur tends to yield lower PSNR and higher MSE, consistent with its strong directional attenuation of frequencies.

Image-dependent behavior. Figure 3 highlights that textured images (e.g. `this_is_fine.png`) can have noticeably higher MSE under the same kernel and recovery procedure. In particular, under `RandomSeed`, `this_is_fine.png` exhibits the highest MSE among the shown settings, indicating that fine textures and scan-like grain are more sensitive to deconvolution noise amplification.

Qualitative Results

We complement MSE/PSNR with qualitative inspection of recovered images (Appendix A). Across all kernels, a consistent trade-off appears: deconvolution sharpens edges, but also **amplifies noise** and typically introduces **ringing/halo artifacts** around high-contrast boundaries.

- **`antimeme.png` (vector-like, flat regions).** Recovery restores the main character silhouette and the circular background boundary relatively well. Although `antimeme.png` often attains relatively low MSE due to its large flat regions, the loss of small, high-frequency text causes disproportionate perceptual degradation that is not well captured by global error metrics.
- **`save_me.png` (real photo + text + scribble).** Recovery tends to sharpen the Chinese text block and edges of the subject, but also introduces visible grain. As a lot of the image is nearly black from

the face of the sheep, it turns out recovery is somewhat unexciting, and it is harder to examine the true effects of encryption and decryption visually.

- **`this_is_fine.png` (comic scan texture).** Recovery improves line sharpness and speech-bubble boundaries, but scan-like texture and noise are strongly amplified. This is most noticeable under more "irregular" kernels (e.g. `RandomSeed`) where inversion is effectively harsher on high-frequency components.

Encryption-specific qualitative behavior. In the private-key setting, correct-key decryption visibly restores structure with sharper edges and clearer shapes. By contrast, wrong-key decryption remains similar to the encrypted observation, being blurred and detail-suppressed. This visual separation supports the quantitative PSNR gaps shown in Figure 2.

Discussion

Interpretation of Results

The experiments support the central "duality" claim of this project: the same ill-posed inverse problem that makes deblurring difficult can also be leveraged to create a key-dependent transformation. In the frequency-domain solution

$$\hat{X}(\omega) = \frac{\overline{\hat{K}(\omega)}\,\hat{Y}(\omega)}{|\hat{K}(\omega)|^2 + \lambda},$$

recovery relies critically on the correct $\hat{K}(\omega)$. If the wrong kernel is used, the filter effectively divides by the wrong spectrum and mis-weights frequencies, producing systematic distortion rather than true inversion. With additive noise, this mismatch is amplified, which explains the strong PSNR gap between correct-key and wrong-key decryption in Figure 2.

Differences across kernels are also consistent with spectral intuition. Gaussian blur attenuates high frequencies smoothly and isotropically, making regularized inversion relatively stable. Motion blur is more challenging because it suppresses frequency content directionally; frequencies that are heavily attenuated cannot be reliably recovered, and the inverse filter tends to amplify noise along those directions, lowering PSNR and increasing visible artifacts.

For private kernels, the `KeyImage` kernel tends to cause stronger correct-key recovery than `RandomSeed` in our setup. One plausible reason is that the key-image-derived kernel may produce a spectrum that is less degenerate than the random seed kernel we sampled, making the regularized inverse less aggressive. Meanwhile, wrong-key decryption remains poor for both, since mismatched inversion does not reconstruct the true frequency content of the original image.

Comparison to Literature

Classical inverse problem literature emphasizes that deconvolution is ill-posed under noise and requires regularization for stability. Our results align with this view: sharper reconstructions come with noise amplification and ringing artifacts. Additionally, prior work has noted that MSE/PSNR do not perfectly capture human perception. In our experiments, this appears clearly in small, high-frequency text regions: they can be visually unrecoverable even when global PSNR remains acceptable, because global metrics dilute errors over the entire image such as in `antimeme.png`. On the other hand, global PSNR could be very high, but

have its recovery be visually accurate, in pictures with a lot of texture but not many minute details such as in `this_is_fine.png`.

Limitations

- **Non-blind setting.** We assume access to the true kernel (or intentionally mismatched kernels). Real scenarios may require blind kernel estimation.
- **Model mismatch.** FFT-based convolution typically implies circular boundary conditions unless special padding/handling is used; boundary artifacts can influence both recovery and metrics.
- **Single regularizer family.** We only use Tikhonov regularization with a chosen λ, but different images and kernels may prefer different regularization strengths or priors.
- **Limited metrics.** We emphasize PSNR/MSE for objective comparison, but perceptual quality could be better evaluated with SSIM or perceptual metrics.

Implications

From a privacy perspective, the private-kernel experiments illustrate a simple but concrete mechanism: authorized users with the correct key can recover meaningful structure, while unauthorized users using an incorrect key obtain a degraded output that remains blurred and detail-suppressed. Although this is not a cryptographic guarantee, it exemplifies how instability of inverse problems can be repurposed into a practical separation between "with-key" and "without-key" recovery quality.

Further Research

Several natural extensions follow from this project:

- Evaluate perceptual similarity (e.g. SSIM) and compare against PSNR/MSE-based conclusions.
- Explore alternative regularization (e.g. gradient-based / TV priors) to reduce ringing and noise amplification.
- Study robustness to stronger noise and to partial key mismatch (e.g. slightly wrong seed or perturbed key image).
- Consider blind deconvolution or kernel estimation attacks to assess privacy strength under adversarial settings.

Conclusion

Summary of Key Findings

This project implemented an FFT-based convolution forward model with additive Gaussian noise and performed image recovery using Tikhonov-regularized deconvolution. Across three images and five kernel types, correct-kernel recovery consistently outperformed mismatched recovery. In the private-key setting, correct-key decryption achieved substantially higher PSNR than wrong-key decryption, demonstrating

strong sensitivity to kernel knowledge.

Research Contribution

The main contribution of this project is a unified experimental framing: convolutional degradation and key-based "encryption" can be studied under the same inverse-problem model. Under this framing, ill-posedness is not only a reconstruction challenge but also a mechanism that creates a measurable gap between authorized and unauthorized recovery.

Broader Implications

More broadly, these experiments highlight that numerical metrics such as MSE and PSNR capture global fidelity but may miss perceptually critical details (e.g. small text). This reinforces the need to pair quantitative evaluation with qualitative inspection when judging recovery performance and privacy outcomes.

Final Thoughts

Overall, the results support the hypothesis that inverse-problem instability can simultaneously limit deblurring under noise and enable key-dependence for privacy-oriented transformations, motivating further study with stronger priors and perceptual evaluation.

Acknowledgements on LLM Usage

Acknowledgements

I have used ChatGPT to help me proofread when writing this report, and also helped me generate some of my code after giving it relevant mathematical formulae.

References

[CT65] J. W. Cooley and J. W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19 (1965), pp. 297–301. URL: https://api.semanticscholar.org/CorpusID:121744946.

[TA77] A. N. Tikhonov and V. Y. Arsenin. "Solutions of ill-posed problems". In: 1977. URL: https://api.semanticscholar.org/CorpusID:122072756.

[Wan+04] Z. Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861. URL: https://ieeexplore.ieee.org/document/1284395.

[GW18] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 4th ed. New York, NY: Pearson, 2018.

[Luo+24] Y. Luo et al. "Enhancing privacy management protection through secure and efficient processing of image information based on the fine-grained thumbnail-preserving encryption". In: *Information Processing & Management* 61.5 (2024), p. 103789. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2024.103789. URL: https://www.sciencedirect.com/science/article/pii/S0306457324001493.

Appendix A: All Generated Images

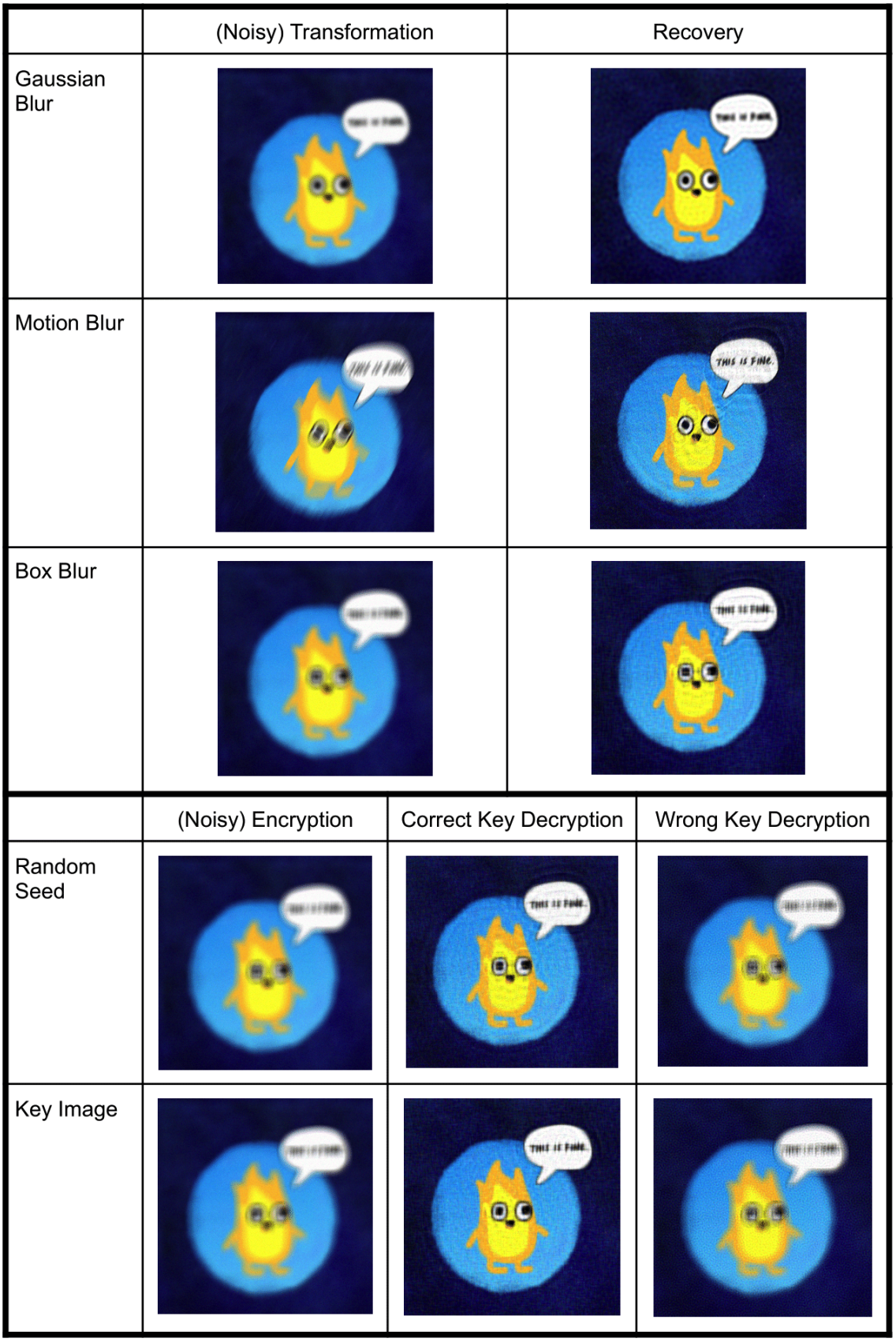


Figure 5: `antimeme.png` results

	(Noisy) Transformation	Recovery
Gaussian Blur		
Motion Blur		
Box Blur		

	(Noisy) Encryption	Correct Key Decryption	Wrong Key Decryption
Random Seed			
Key Image			

Figure 6: `save_me.png` results

	(Noisy) Transformation	Recovery
Gaussian Blur		
Motion Blur		
Box Blur		

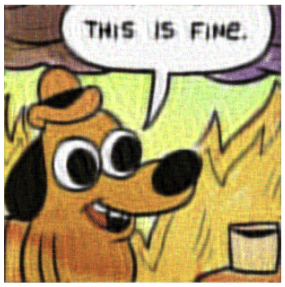
	(Noisy) Encryption	Correct Key Decryption	Wrong Key Decryption
Random Seed			
Key Image			

Figure 7: `this_is_fine.png` results

Appendix B: Code

For more information and to see all of my code, please reference my GitHub repo for this project at https://github.com/shun4midx/ICM-Final-Project. I will include the most important parts of the full code here.

main.py

```
import numpy as np
from utils import load_image, save_image
from kernels import gaussian_kernel, motion_kernel, box_kernel,
   random_kernel, key_image_kernel_rgb
from fft_deconv import apply_kernel_fft, deconv_fft
from metrics import mse, psnr
import csv
import os

IMAGES = ["antimeme.png", "this_is_fine.png", "save_me.png"]
IMAGE_DIR = "../images/"
RESULT_DIR = "../results/images/"
CSV_PATH = "../results/metrics.csv"

for img_name in IMAGES:
    img_folder = f"{RESULT_DIR}/{img_name[:-4]}"
    os.makedirs(img_folder, exist_ok=True)

def run_experiment():
    kernel_functions = {
        "Gaussian": lambda: gaussian_kernel(size=25, sigma=5),
        "Motion": lambda: motion_kernel(length=31, angle=60),
        "Box": lambda: box_kernel(size=20),
        "RandomSeed": lambda: random_kernel(size=21, seed=123),
        "KeyImage": lambda: key_image_kernel_rgb("../images/private_key.
           png", size=21)
    }

    noise_sigma = 0.03
    lam = 1e-2

    with open(CSV_PATH, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["image", "kernel", "correct_key", "mse", "psnr"
           ])

        for img_name in IMAGES:
            x = load_image(IMAGE_DIR + img_name)

            for kernel_name, kernel_fn in kernel_functions.items():
                K = kernel_fn()

                # FORWARD OPERATOR
```

```
                y = apply_kernel_fft(x, K)  # blurred/encrypted image
                z = y + noise_sigma * np.random.randn(*y.shape) # noisy
                    observed image

                # Save the FORWARD IMAGES (new)
                save_image(y, f"{RESULT_DIR}/{img_name[:-4]}/{kernel_name
                    }_forward_clean.png")
                save_image(z, f"{RESULT_DIR}/{img_name[:-4]}/{kernel_name
                    }_forward_noisy.png")

                # CORRECT-KEY RECOVERY
                x_hat = deconv_fft(z, K, lam)
                m = mse(x, x_hat)
                p = psnr(m)

                save_image(x_hat, f"{RESULT_DIR}/{img_name[:-4]}/{
                    kernel_name}_recovered_correct.png")
                writer.writerow([img_name, kernel_name, "True", m, p])

                # WRONG-KEY ONLY FOR ENCRYPTION KERNELS
                if kernel_name in ["RandomSeed", "KeyImage"]:
                    wrong_K = gaussian_kernel(size=11, sigma=2)
                    x_wrong = deconv_fft(z, wrong_K, lam)
                    m2 = mse(x, x_wrong)
                    p2 = psnr(m2)

                    save_image(x_wrong, f"{RESULT_DIR}/{img_name[:-4]}/{
                        kernel_name}_recovered_wrong.png")
                    writer.writerow([img_name, kernel_name, "False", m2,
                        p2])

if __name__ == "__main__":
    run_experiment()
```

fft_deconv.py

```
import numpy as np

def pad_kernel(K, shape):
    pad = np.zeros(shape)
    kh, kw = K.shape
    pad[:kh, :kw] = K
    return pad

def apply_kernel_fft(x, K):
    # x: H x W x 3
    # K: either (kh, kw)  or (3, kh, kw)

    H, W, _ = x.shape

    # Case 1: shared kernel across RGB
```

```
    if K.ndim == 2:
        Khat = np.fft.fft2(pad_kernel(K, (H, W)))
        y = np.zeros_like(x)
        for c in range(3):
            Xhat = np.fft.fft2(x[:, :, c])
            Yhat = Khat * Xhat
            y[:, :, c] = np.real(np.fft.ifft2(Yhat))
        return y

    # Case 2: per-channel RGB kernel
    elif K.ndim == 3:
        y = np.zeros_like(x)
        for c in range(3):
            Kc = K[c]
            Khat = np.fft.fft2(pad_kernel(Kc, (H, W)))
            Xhat = np.fft.fft2(x[:, :, c])
            Yhat = Khat * Xhat
            y[:, :, c] = np.real(np.fft.ifft2(Yhat))
        return y

    else:
        raise ValueError("Kernel K must be 2D or 3D")

def apply_kernel_fft_rgb(x, K):
    # K has shape (3, kH, kW)
    out = np.zeros_like(x)

    for c in range(3):
        out[:, :, c] = apply_kernel_fft(x[:, :, c], K[c])

    return out

def deconv_fft(z, K, lam):
    H, W, _ = z.shape
    xhat = np.zeros_like(z)

    # Shared kernel
    if K.ndim == 2:
        Khat = np.fft.fft2(pad_kernel(K, (H,W)))
        denom = np.abs(Khat)**2 + lam
        for c in range(3):
            Zhat = np.fft.fft2(z[:, :, c])
            Xhat = (np.conj(Khat) * Zhat) / denom
            xhat[:, :, c] = np.real(np.fft.ifft2(Xhat))
        return xhat

    # RGB kernel
    elif K.ndim == 3:
        for c in range(3):
            Kc = K[c]
            Khat = np.fft.fft2(pad_kernel(Kc, (H,W)))
            denom = np.abs(Khat)**2 + lam
```

```
            Zhat = np.fft.fft2(z[:, :, c])
            Xhat = (np.conj(Khat) * Zhat) / denom
            xhat[:, :, c] = np.real(np.fft.ifft2(Xhat))
        return xhat

    else:
        raise ValueError("Kernel K must be 2D or 3D")

def deconv_fft_rgb(z, K, lam):
    out = np.zeros_like(z)

    for c in range(3):
        out[:, :, c] = deconv_fft(z[:, :, c], K[c], lam)

    return out
```

kernels.py

```
import numpy as np
from scipy.ndimage import rotate
from utils import load_image, resize_np

def gaussian_kernel(size=11, sigma=2):
    ax = np.arange(-(size//2), size//2+1)
    xx, yy = np.meshgrid(ax, ax)
    g = np.exp(-(xx**2 + yy**2) / (2*sigma*sigma))
    return g / g.sum()

def box_kernel(size=9):
    return np.ones((size, size)) / (size*size)

def motion_kernel(length=15, angle=0):
    k = np.zeros((length, length))
    k[length//2, :] = 1
    k = rotate(k, angle, reshape=False)
    return k / k.sum()

def random_kernel(size=15, seed=123):
    np.random.seed(seed)
    k = np.random.rand(size, size)
    return k / k.sum()

def key_image_kernel_rgb(path, size=15):
    # Load RGB key image as float array
    img = load_image(path)  # returns 512x512x3 normalized

    # Extract RGB channels
    R = img[:, :, 0]
    G = img[:, :, 1]
    B = img[:, :, 2]
```

```
    # Resize channels
    KR = resize_np(R, size)
    KG = resize_np(G, size)
    KB = resize_np(B, size)

    # Normalize each channel to sum = 1
    KR = KR / np.sum(KR)
    KG = KG / np.sum(KG)
    KB = KB / np.sum(KB)

    # Return RGB kernel (3, size, size)
    return np.stack([KR, KG, KB], axis=0)
```

metrics.py

```
import numpy as np

def mse(a, b):
    return np.mean((a - b)**2)

def psnr(mse):
    return 10 * np.log10(1.0 / mse)
```

Appendix C: Mathematical Formulae

The formulations below are included for completeness and reference. Conceptual explanations and interpretations required for this project are provided in the "Background and Theoretical Framework" subsection.

C.1 Image and Forward Model

A color image is modeled as a discrete RGB function

$$x(i, j, c), \quad c \in \{R, G, B\},$$

with pixel values normalized to the range $[0, 1]$.

Image degradation is modeled as a linear convolution followed by additive noise:

$$y = Kx + \eta,$$

where K is a convolution operator and η is additive Gaussian noise with variance σ^2.

For a spatially invariant blur kernel $k(u, v)$, the convolution operator acts as

$$(Kx)(i, j, c) = \sum_{u,v} k(u, v)\, x(i - u, j - v, c).$$

C.2 Tikhonov-Regularized Inverse Problem

Recovering x from y is an ill-posed inverse problem, which manifests numerically as instability under noise. To stabilize the inversion, we solve the Tikhonov-regularized least squares problem

$$\hat{x} = \arg\min_x \|Kx - y\|^2 + \lambda \|x\|^2,$$

where $\lambda > 0$ is the regularization parameter controlling the trade-off between faithfulness to the original image and stability [TA77].

C.3 Frequency-Domain Solution

Since convolution operators are diagonalized by the Fourier transform, the solution can be computed efficiently in the frequency domain. Let $\hat{K}$ and $\hat{Y}$ denote the Fourier transforms of the kernel and observed image, respectively. The regularized solution is given pointwise in frequency by

$$\hat{X}(\omega) = \frac{\overline{\hat{K}(\omega)}\, \hat{Y}(\omega)}{|\hat{K}(\omega)|^2 + \lambda}.$$

The recovered image $\hat{x}$ is obtained by applying the inverse Fourier transform to $\hat{X}$. This computation is performed independently for each color channel.

C.4 Error Metrics

Reconstruction quality is evaluated using Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR).

For two RGB images x and $\hat{x}$ with N total pixel values,

$$\mathrm{MSE}(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2.$$

Assuming pixel values lie in $[0, 1]$, PSNR is defined as

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{1}{\mathrm{MSE}} \right).$$

Higher PSNR and lower MSE indicate better reconstruction quality.