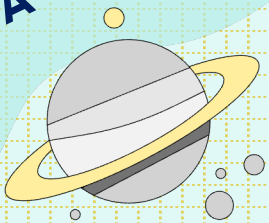
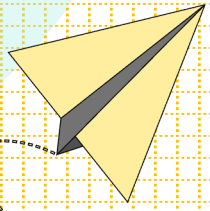
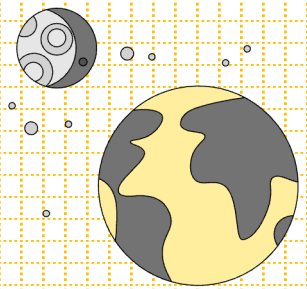
COMPUTATIONAL IMAGING



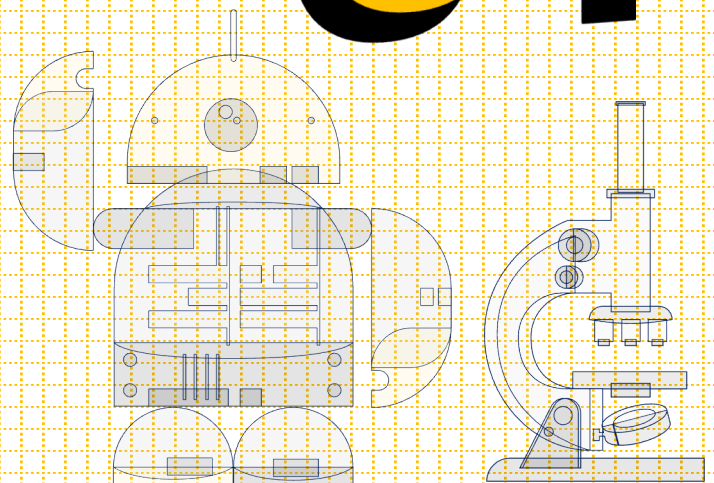




ACTIVITY

01

MOTION deblurring

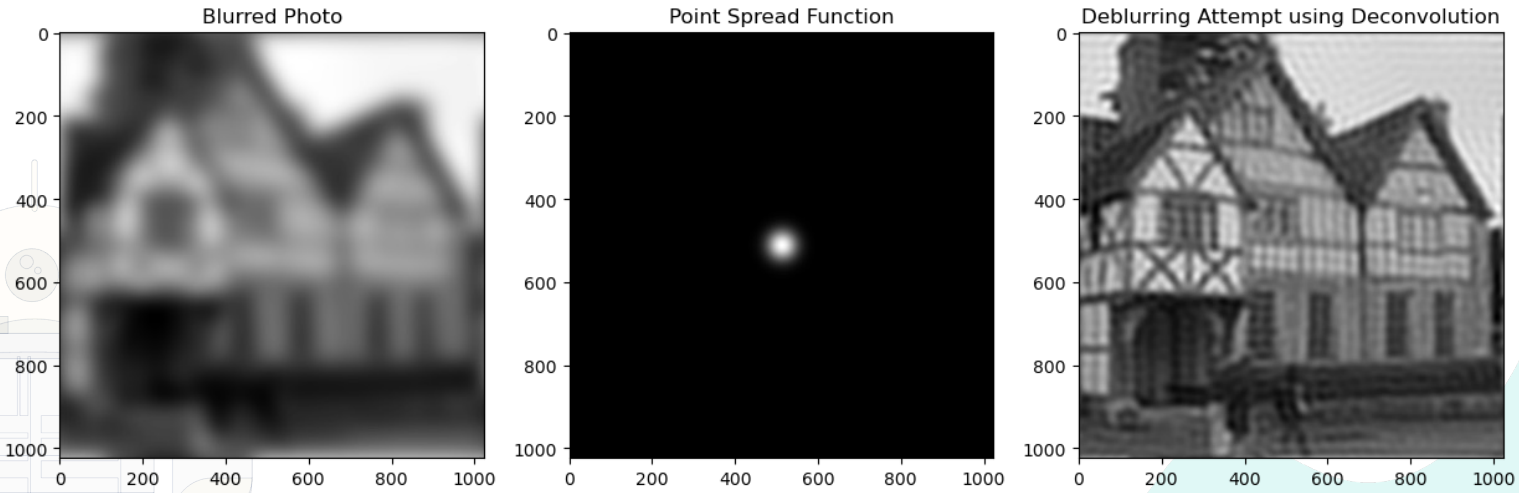


SUBMITTED BY:
RENE L.PRINCIPE JR.

SUBMITTED TO:
DR. MARICOR N. SORIANO

Background

Photo deblurring may seem like magic, but it involves complex physics. Shown below is an image deconvolution technique that leverages a known smearing effect called the point spread function (PSF). The original image appears blurry, as if smeared in all directions. By deciphering the blurred photo's PSF as a Gaussian distribution function, we can apply deconvolution in the Fourier space to reconstruct a much clearer and sharper image. Deblurring using this technique can be valuable in improving low-quality images, with practical applications in identifying faces or letterings from license plates.



Super-Resolution

As mentioned, a captured image is a combination of the original image or object, f(x,y), a PSF that causes smearing h(x,y) and noise n(x,y) given by

$$g(x,y) = h(x,y) * f(x,y) + n(x,y).$$

where (x,y) refers to the coordinates in real-space. In fourier space, the coordinates represents the frequencies (u,v), hence,

$$G(u,v) = H(u,v) * F(u,v) + N(u,v).$$

To recover the F(u,v), wiener filtering allows estimation by minimizing the mean square error, and the solution becomes

$$F(u,v) = \left[\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right] G(u,v),$$

where K is the noise to signal ration(NSR). The ideally pristine original image can then be recovered by taking the inverse Fourier transform of F(u,v).

Method Overview

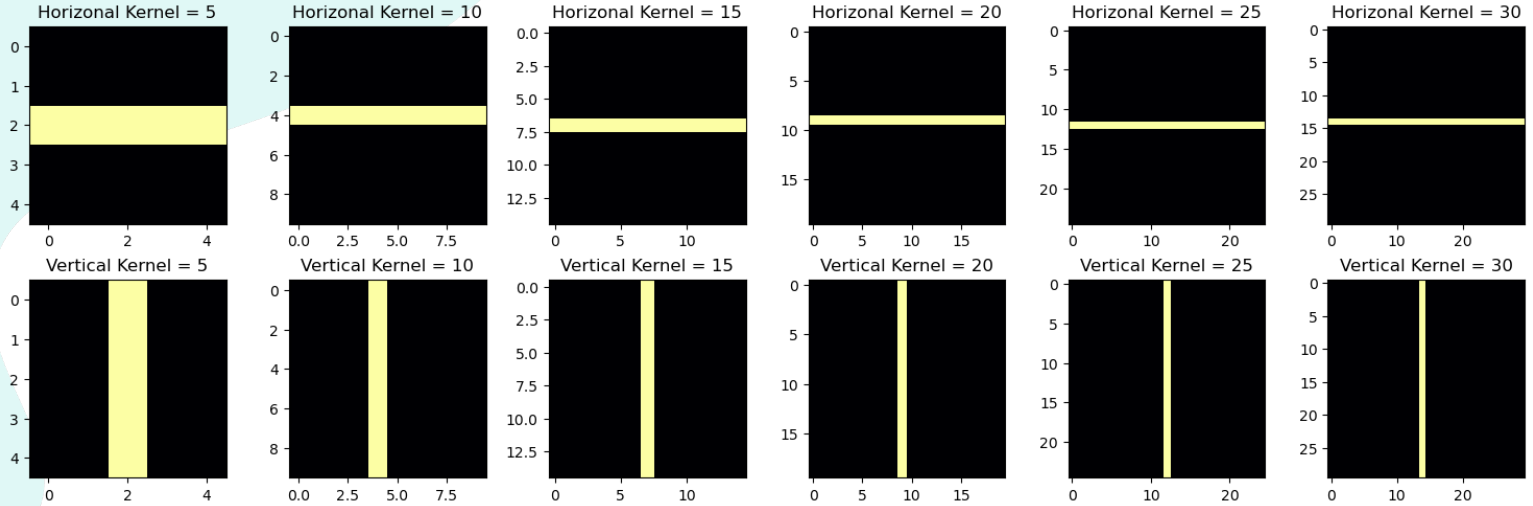
In this activity, we applied blurring kernels of varying orientation and extent sizes, along with increasing degrees of NSR, synthetically, to a test image.

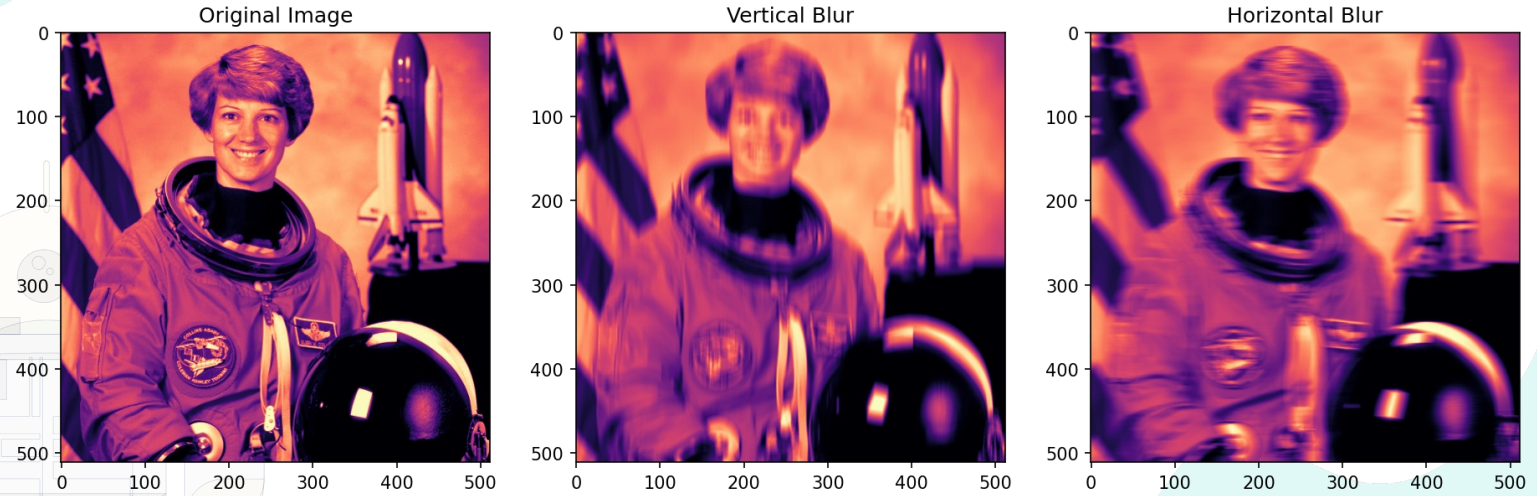
Given the blurred and noisy images, the restoration of the original image was performed by coding the using (1) packaged Weiner estimation via **skimage.restoration** and (1)estimation using Fourier transforms via **numpy.fft** in Python.

The accuracy of reconstruction, given the blurring parameters, was quantified using the Structure Similarity Index (SSIM).

Lastly, we attempted to estimate the blurring parameters, such as motion direction and pixel extent, of a moving car to super-resolve its quality and decipher the text.

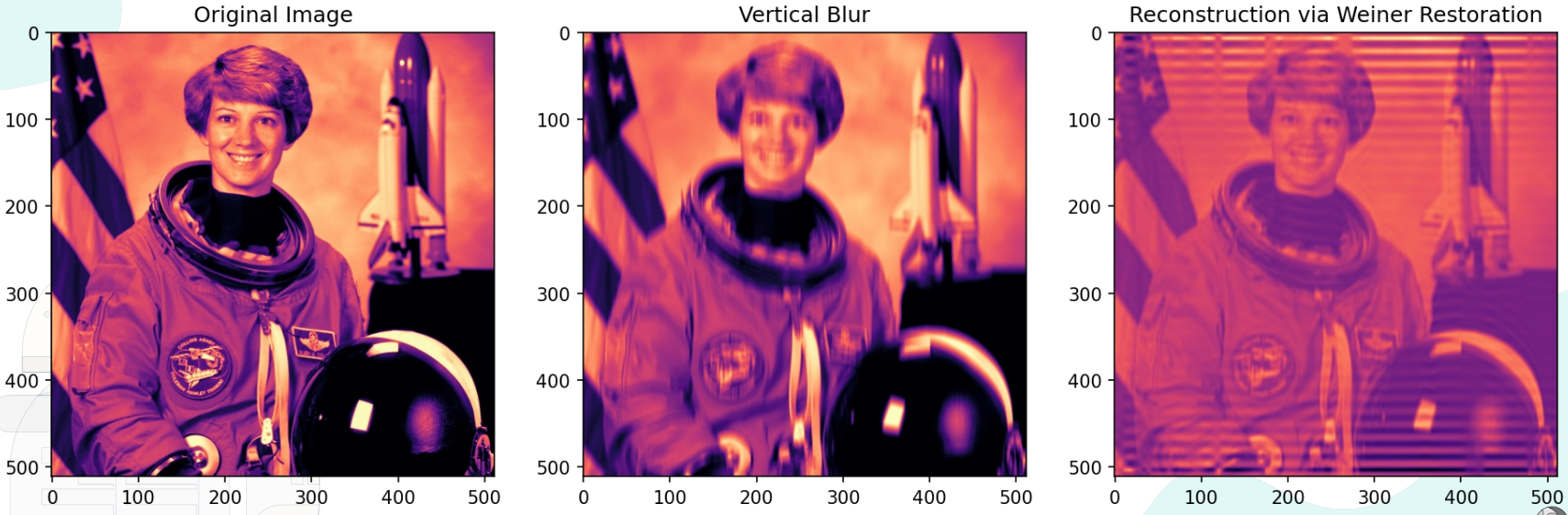
Synthetic kernel blurring





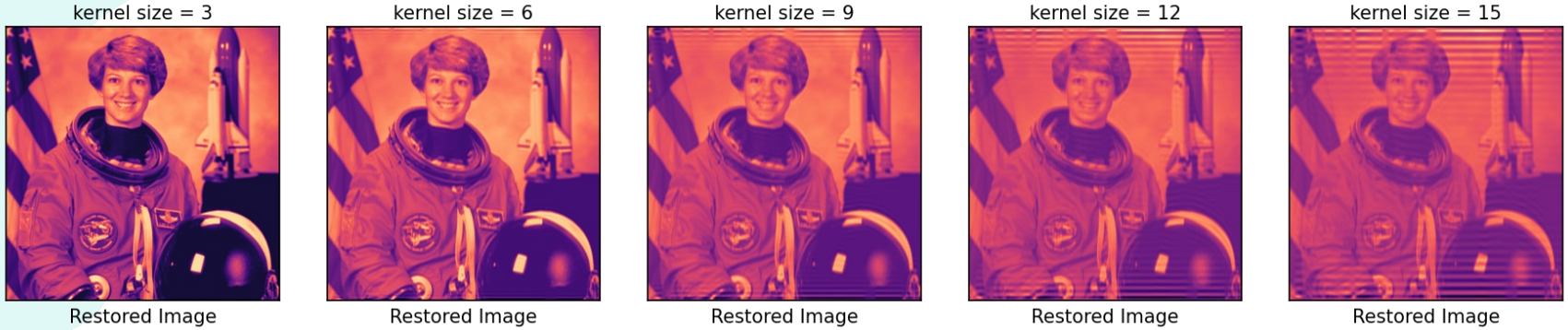
Skimage – Weiner Restoration

After applying synthetic blur, we restored the image using the automated Wiener estimation technique from **skimage.restoration**. A vertical blur smears the original image upwards and downwards, leading to evident artifacts on the image's extremes. This type of blur also compromises the recovered image's contrast, although the astronaut's face becomes more perceivable and potentially usable for identification.



Skimage – Weiner Restoration

By increasing the kernel size for both horizontal and motion blurs, we observed a degradation in restoration quality, with reduced contrast and more prominent artifacts.

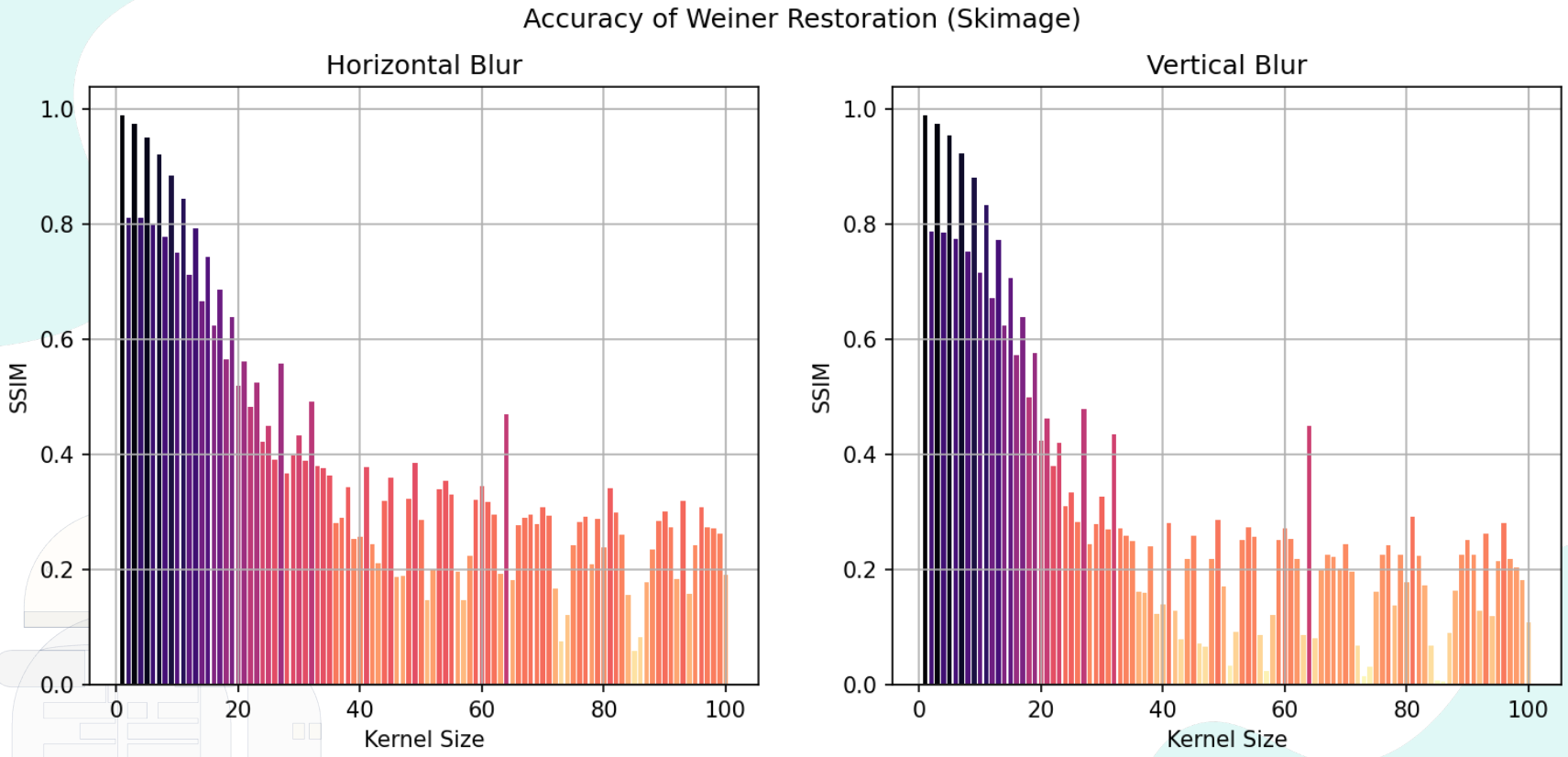


Vertical Blurring Kernel

Horizontal Blurring Kernel

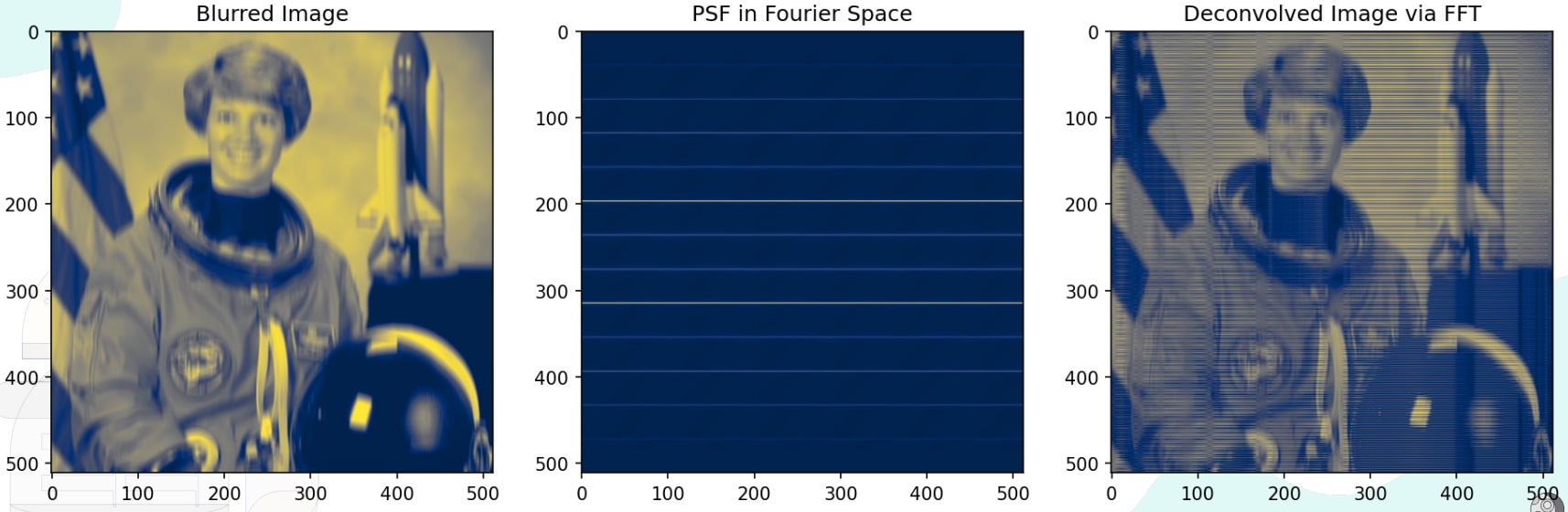
Skimage – Weiner Restoration

Large-scale analysis for small kernel sizes revealed that SSIM fluctuates between odd and even kernel sizes and that vertical blurs tend to degrade the image more than horizontal blur.

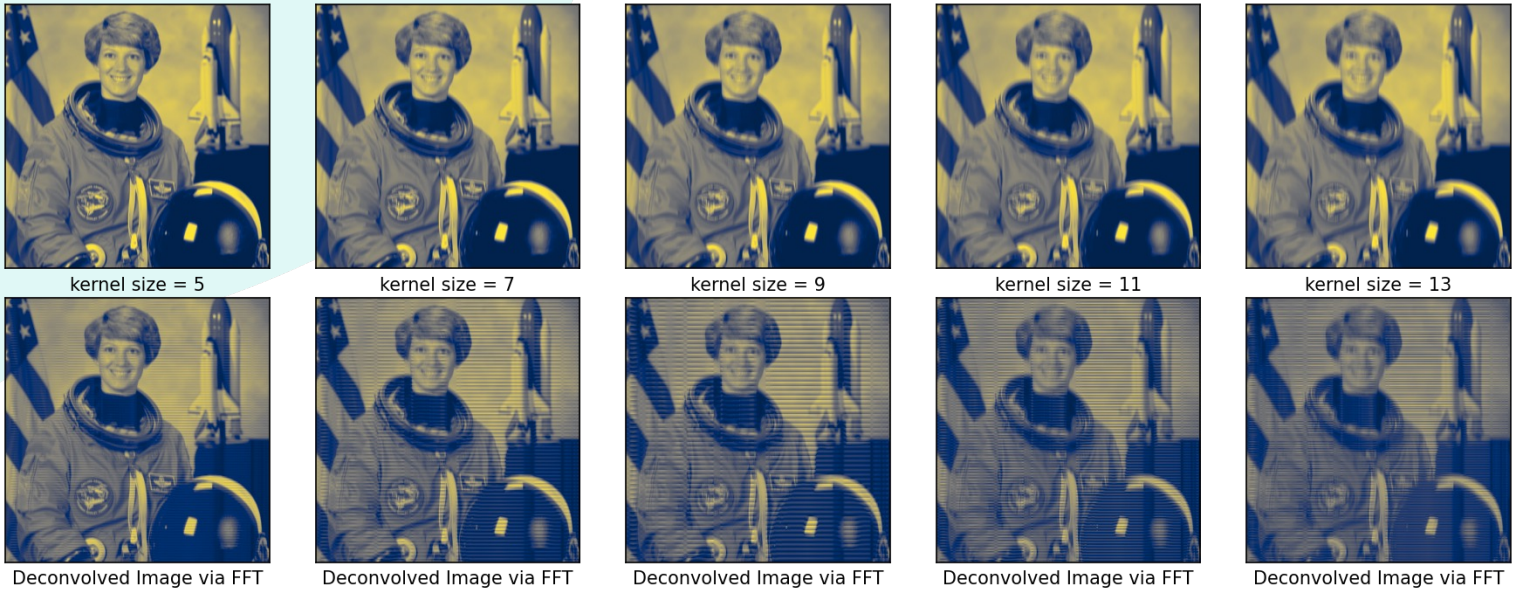


NP.FFT – Weiner Restoration

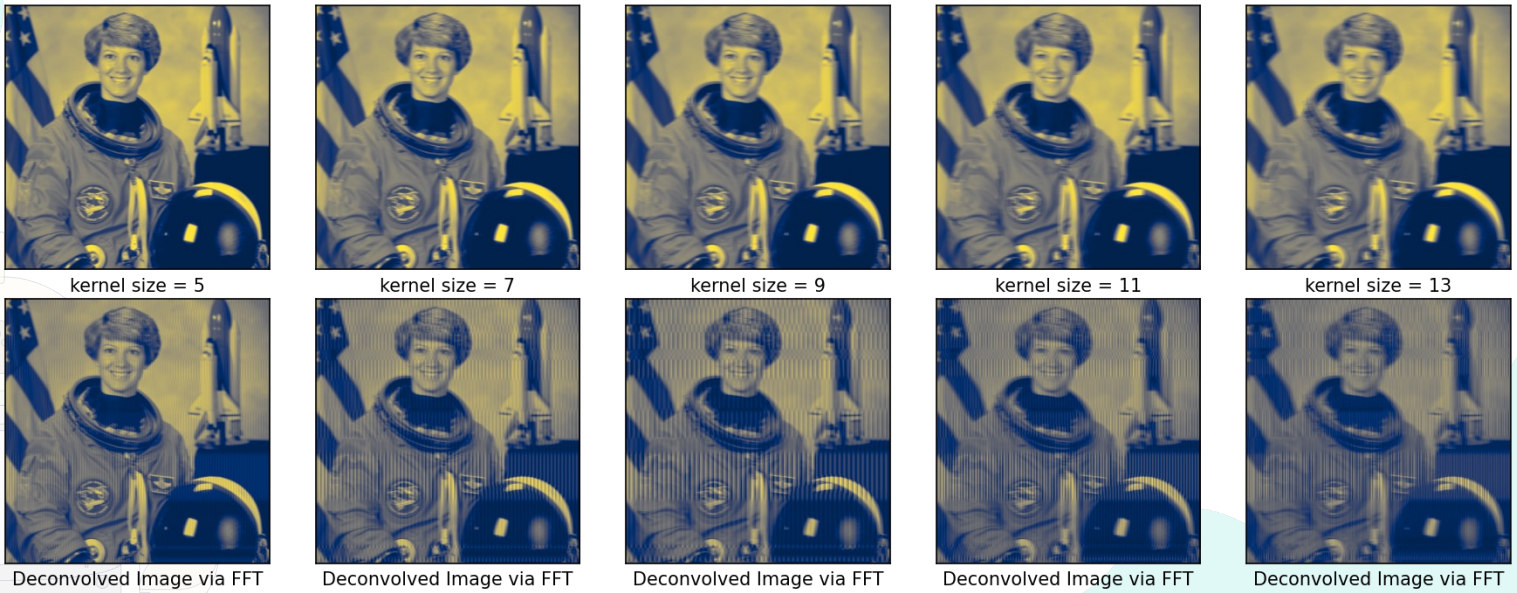
Since a package was used previously, I implemented the equations for restoration step by step on the fourier space via **numpy.fft**. Evidenced by FFT reconstruction is the reduction of contrast. The emergence of artifacts was also observed; however, they are more uniformly distributed compared to **skimage.restoration**. As a result, the face of the astronaut is less perceivable. This observation is true for the horizontal and vertical motion blurs alike.



Weiner Filtering via FFT (Vertical Kernel)

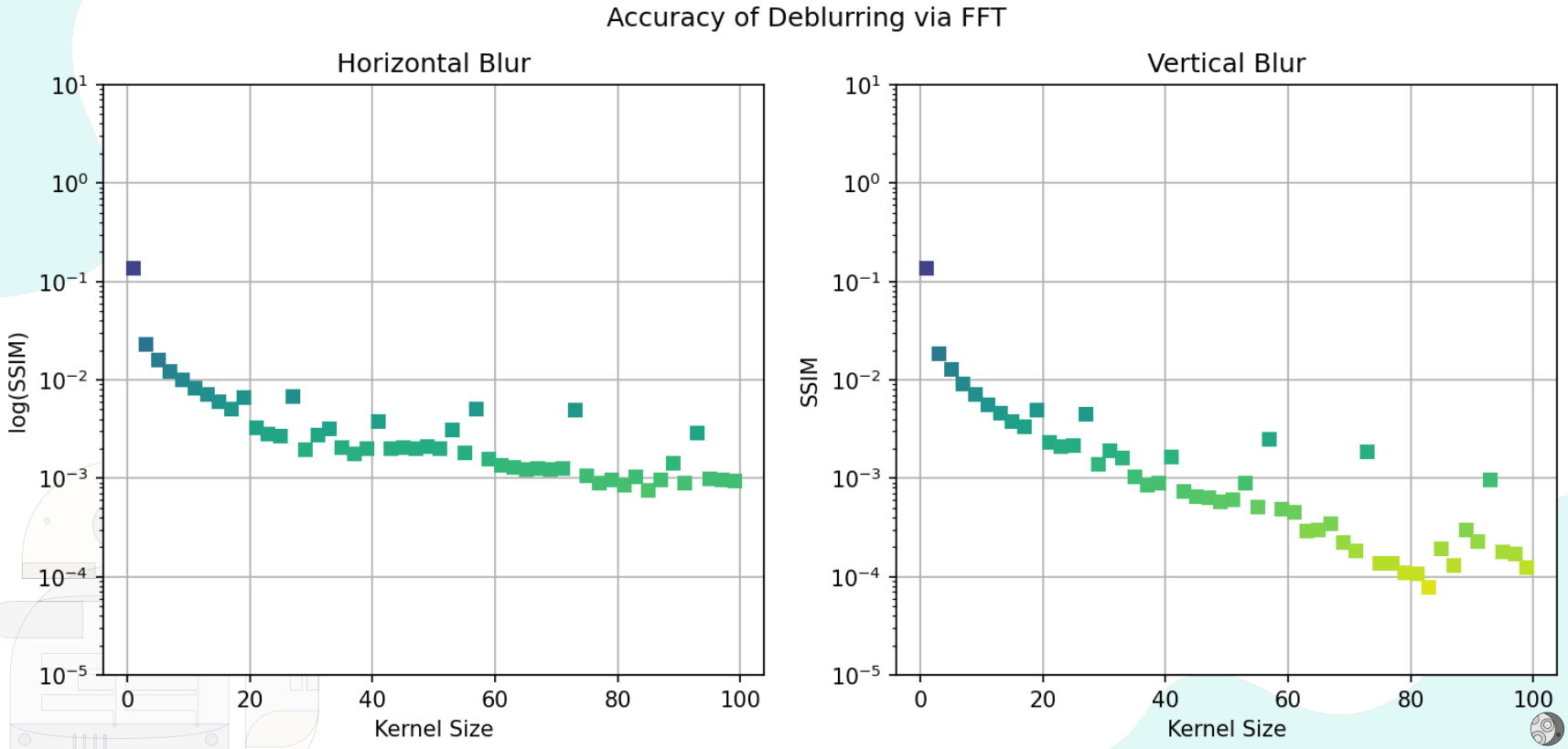


Weiner Filtering via FFT (Horizontal Kernel)



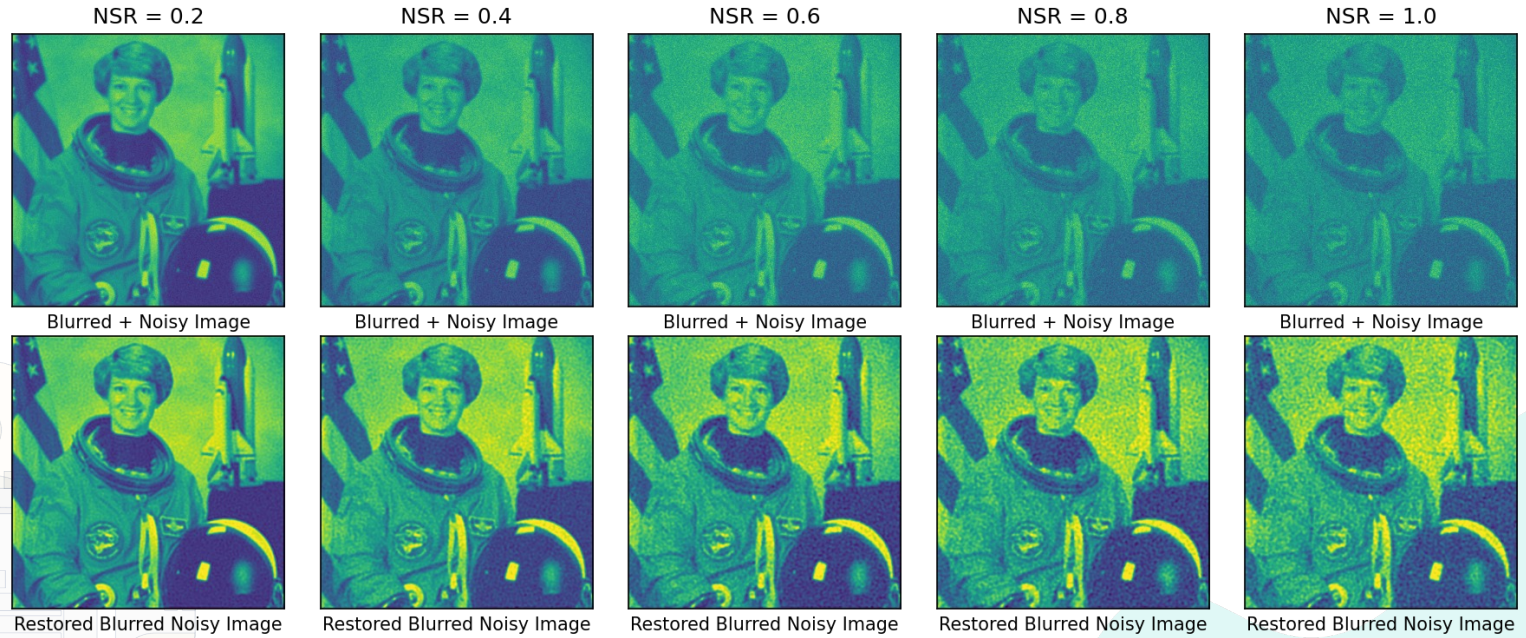
NP.FFT – Weiner Restoration

Large-scale analysis of FFT restoration showed a generally downward trend of SSIM as the kernel size is increased, which means quality is degraded. However, similar to the results using a package, vertical blurs tend to degrade the image more than horizontal blur.

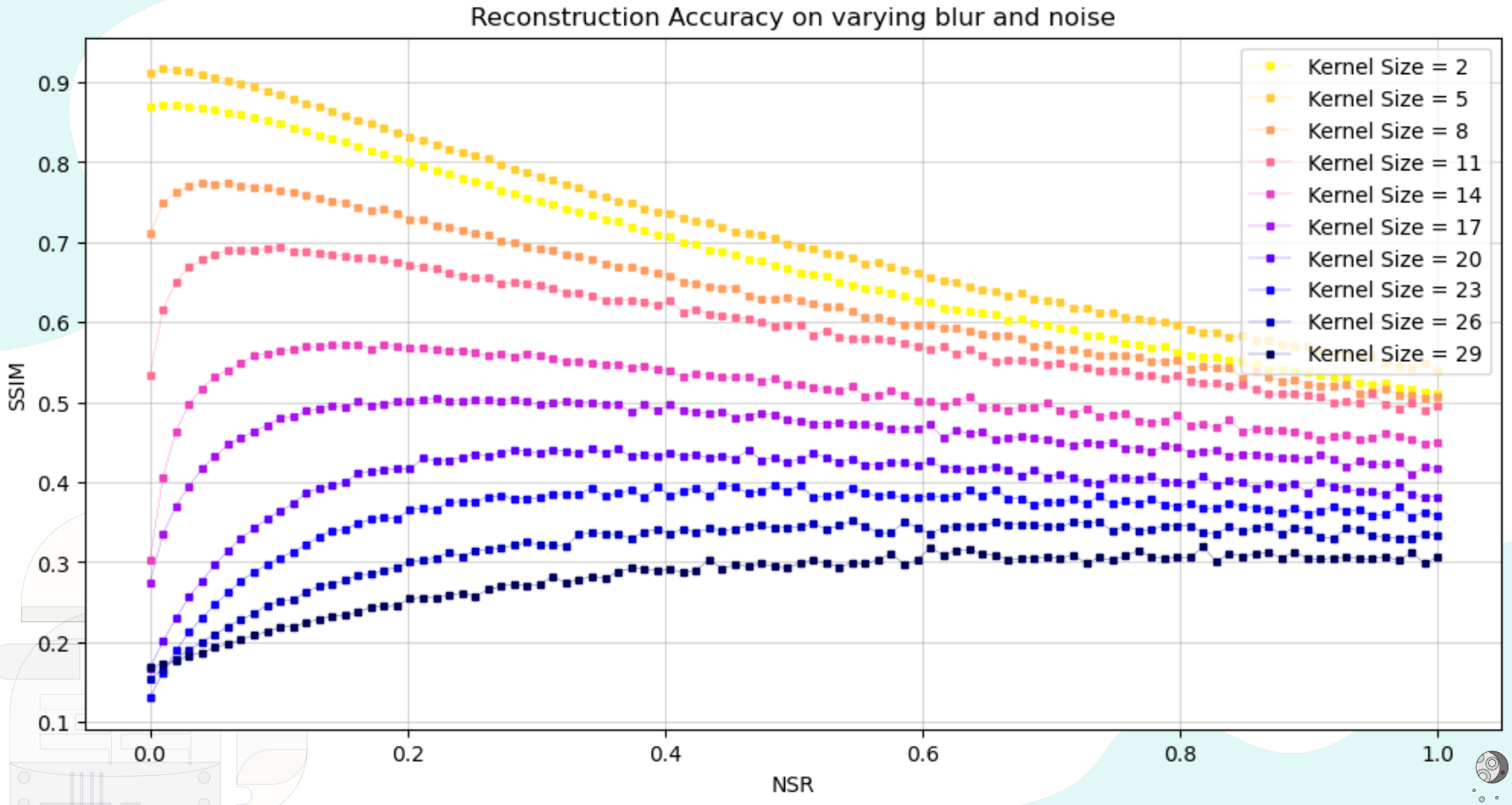


Effects of varying NSR

For a set square blur kernel, the effect of varying NSR was also explored. It was then reconstructed using **skimage.restoration**. Increasing NSR corresponds to increasing the noise, which reduced the contrast of the original image. Restoration results showed the dominant effect of noise but unlike normal blurring, the contrast remained high.

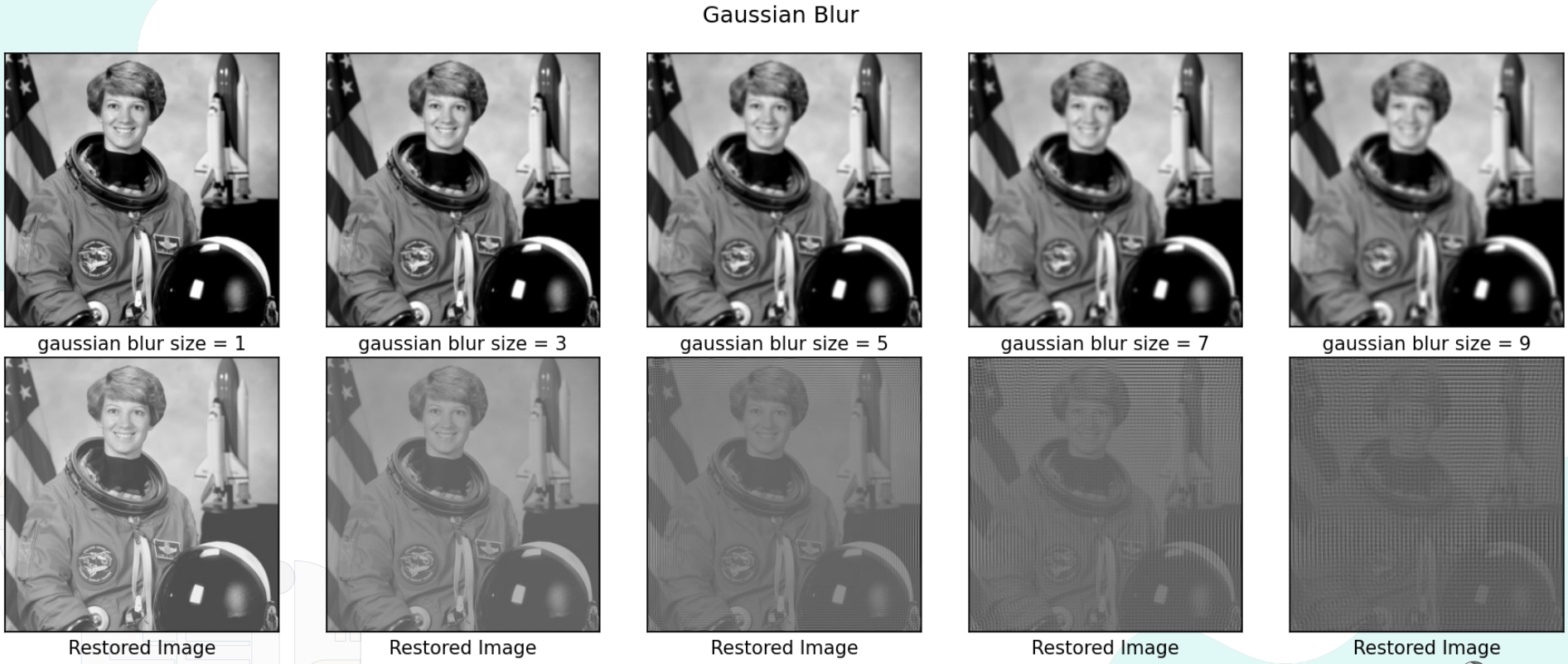


From the blur and noise analysis, the restored image is usable if the motion blur extent is not that drastic, and loss of contrast can be circumvented by adding a sufficient NSR. On increasing kernel size, the restoration is improved by adding a non-zero NSR, until such values that the quality degrades altogether.



Gaussian Blur

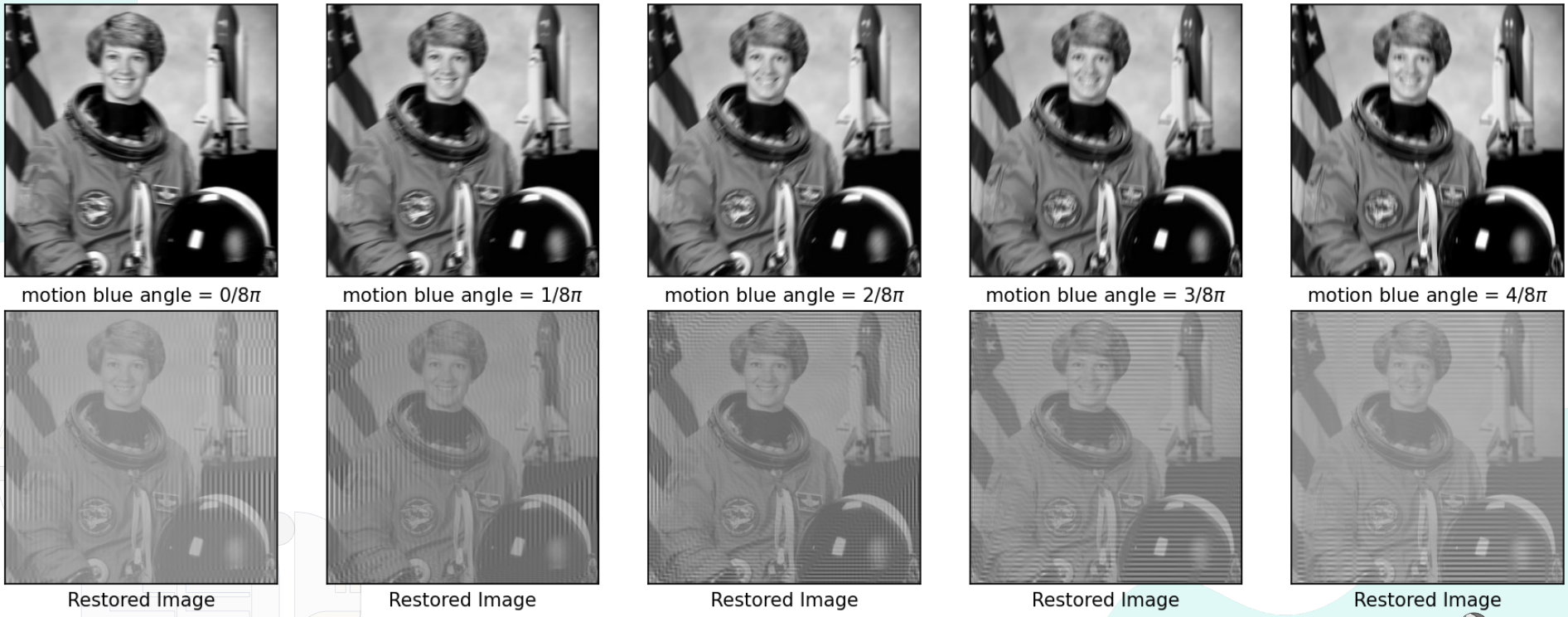
Gaussian blurs drastically degrades the image compared to linear blurs, implying that it might be easier to correct a motion blur than a blur that is a result of an out of focused imaging system.



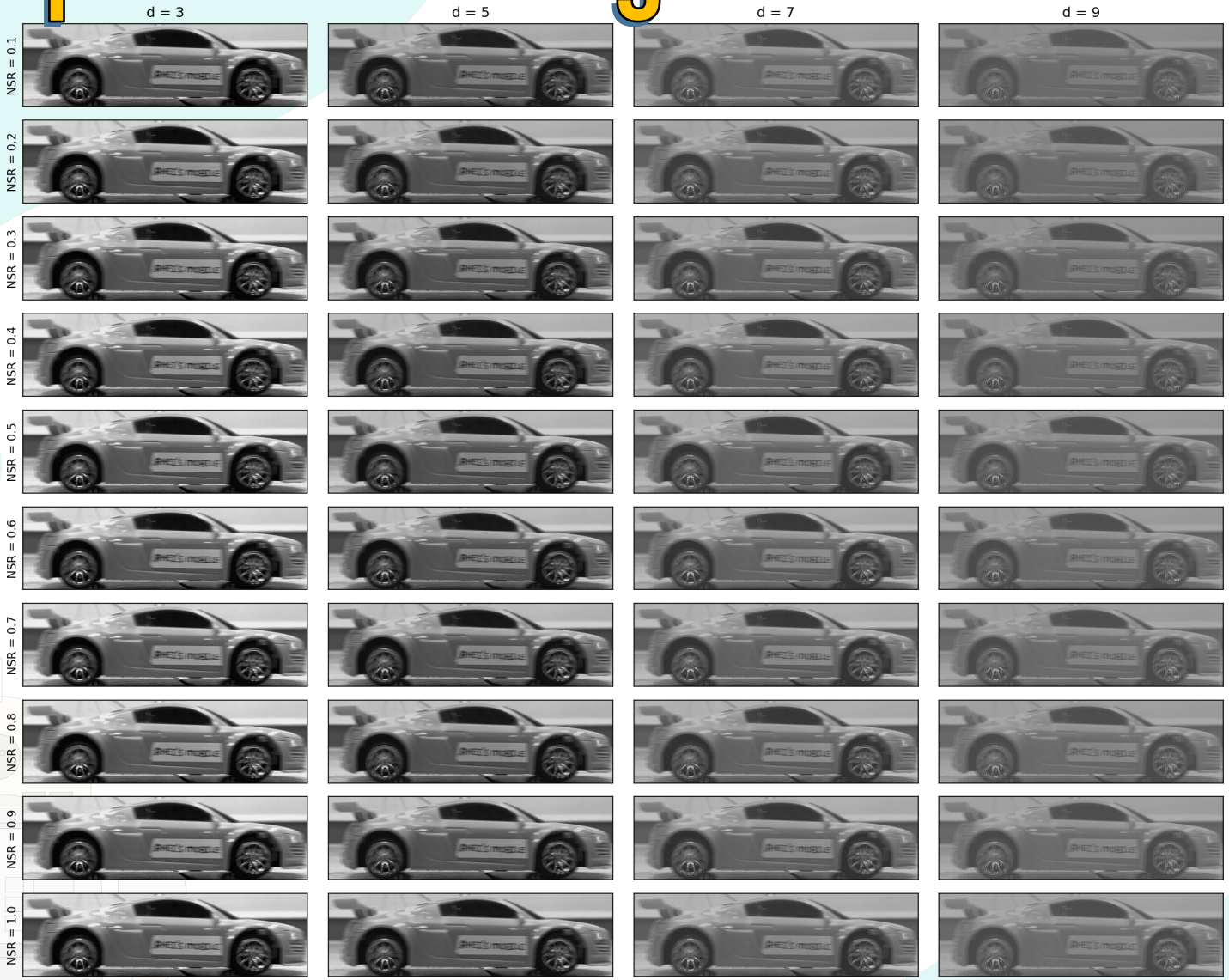
Angular Motion Blur

Meanwhile, angled motion blurs intuitively introduced both horizontal and vertical artifacts.

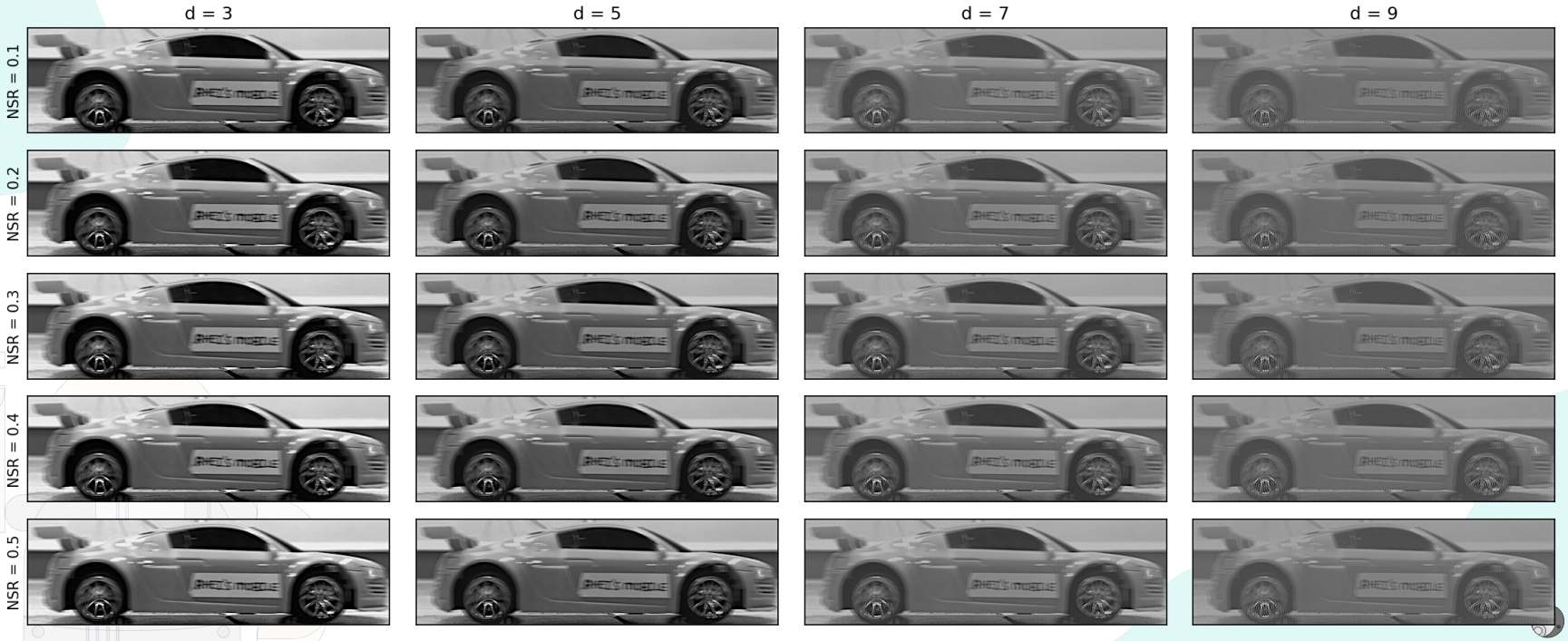
Angled Motion Blur



Super-resolving a video frame



Using a frame from a video of a moving toy car, kernel size d and NSR levels for reconstruction is estimated. As observed, underestimation of the extent of motion blur doesn't resolve the horizontally blur of the image, while overestimation reduced the contrast, degrading the quality. At d = 7, text becomes more apparent as NSR is increased, further reinforcing the simulation findings. Hence, optimal parameters for kernel extent and NSR should vary depending on the conditions of the captured image.



reflection

I have ventured the realm of spectral super-resolution in my undergraduate thesis and back in my App Physics 155 class, we deconvolved an image using a gaussian PSF. This activity further allowed me to explore the various effects of blurring kernel's direction and extent, as well as NSR introduction to simulate the the formation of actual captures. I used two methods: hard coding the process and using a package, and believe I was able to deliver a sufficient output. I further explained the large-scale nuances of the parameters in the results by calculating SSIM reconstruction accuracy, revealing interesting findings. The parameter variations in the simulation was also applied on a real-life moving object to deblur it, hence the spatial super-resolution. It was a fun and fulfilling activity. I'd like to acknowledge Jeremy Narag for giving me a usable and resolvable image frame from the video of a moving toy car.

With that said, I'd give myself a score of **105/100.**

references

- [1] M. Soriano, Physics 305 – Super-Resolution Motion Deblurring, (2023).
- [2] M. Newman, Computational Physics with Python

SOURCE CODE

https://github.com/reneprincipejr/Physics-305/tree/main/Activity%201%20-%20Super-resolution%20Motion%20Deblurring