2. (a)

Spatial domain에서 Laplacian kernel을 적용한 결과는 아래와 같다.

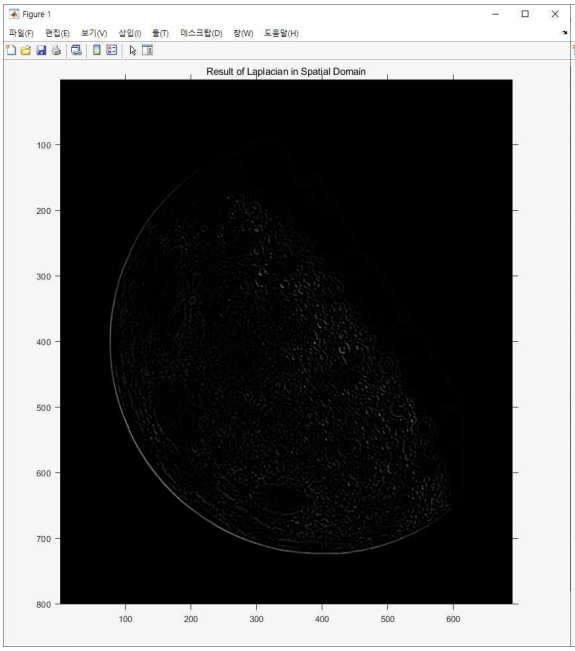

Fig 1. Result of Laplacian in Spatial Domain

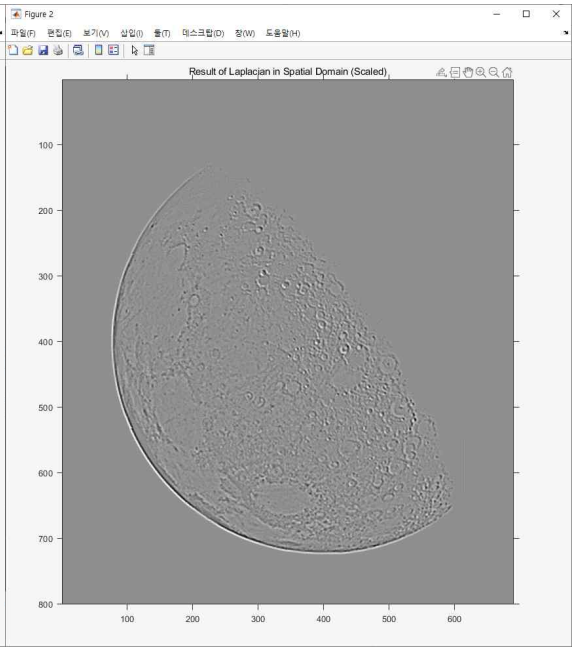

Fig 2. Result of Laplacian in Spatial Domain (scaled)

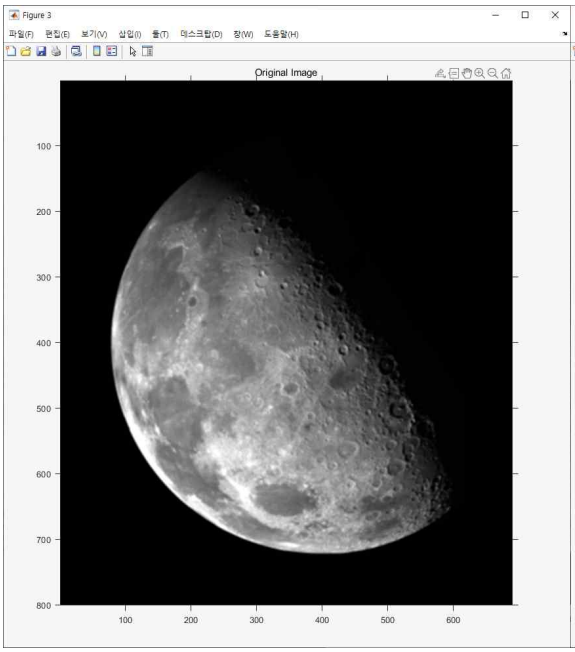

Fig 3. Original Image

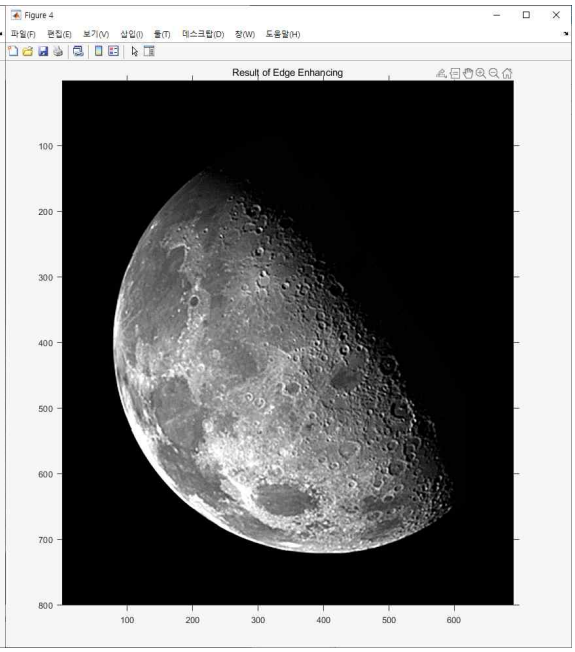

Fig 4. Result of Edge Enhancement in Spatial Domain

2. (b)

Diagonal term을 포함한 Laplacian kernel의 transfer function을 구하는 과정은 이미 앞에서 보였고, 그 결과는 아래와 같다.

$$H(u,v) = 2\cos(2\pi u/M) + 2\cos(2\pi v/N) + 4\cos(2\pi v/N)\cos(2\pi u/M) - 8$$

이를 활용하여, frequency domain에서 Laplacian kernel을 적용한 결과는 아래와 같다.

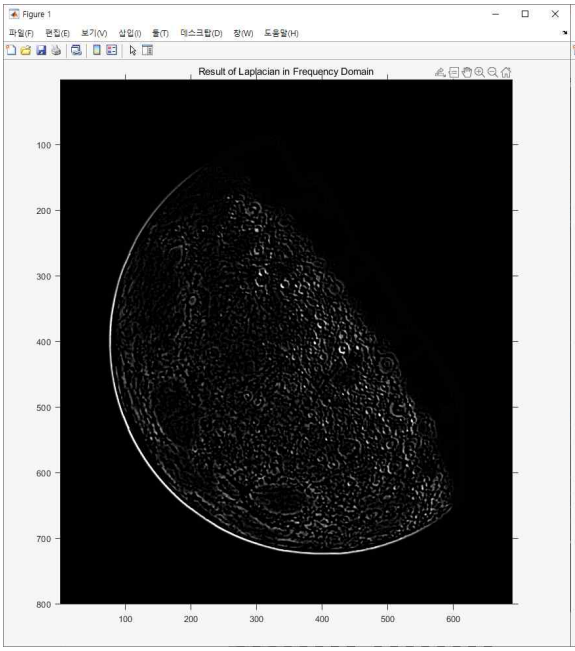

Fig 5. Result of Laplacian in Frequency Domain

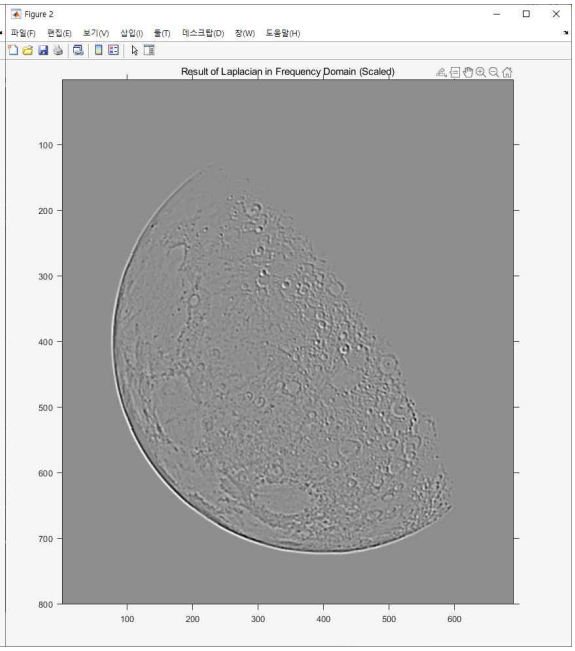

Fig 6. Result of Laplacian in Frequency Domain (scaled)

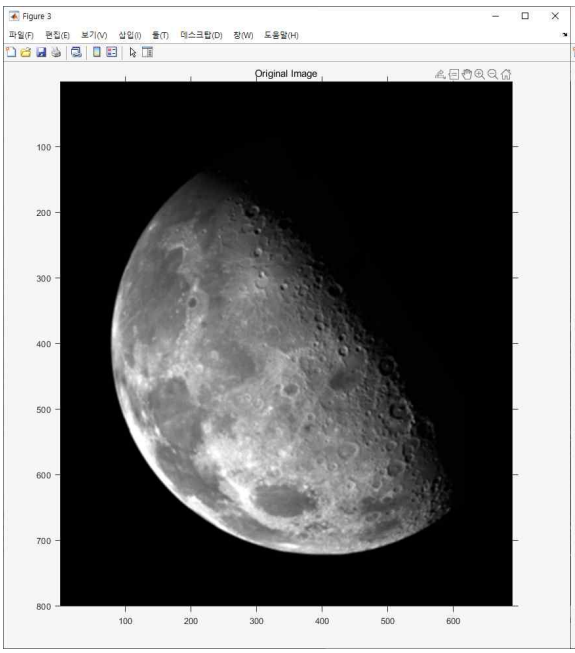

Fig 7. Original Image

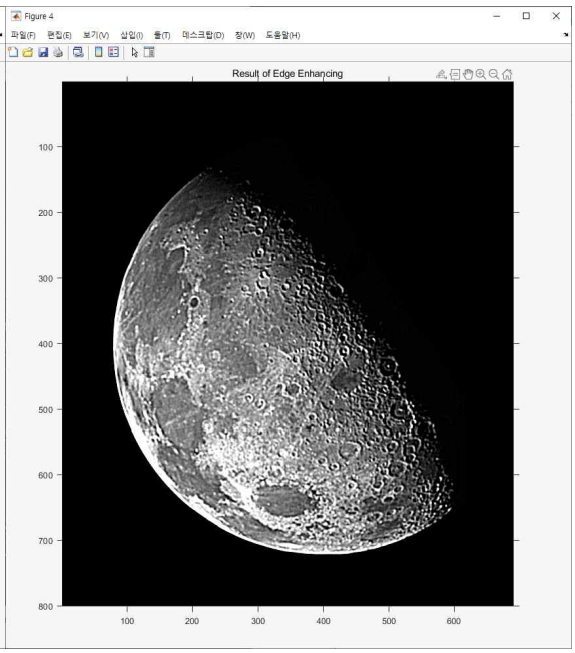

Fig 8. Result of Edge Enhancingment in Frequency Domain

2. (c)

Fig 4는 Spatial domain에서 Laplacian kernel을 적용한 결과이며, Fig 8은 frequency domain에서 Laplacian kernel을 적용한 결과이고, Fig 9는 두 결과의 차이를 명확하게 확인하기 위해 spatial domain에서 얻은 laplacian의 결과에서 frequency domain에서 얻은 laplacian의 결과를 뺀 후 scaling 한 결과이다. Frequency domain에서 Laplacian을 적용한 결과가 edge 영역들을 더욱 잘 강조해주는 것을 확인할 수 있다. 이러한 차이가 발생하는 이유는 spatial domain의 Laplacian kernel에서는 해당 pixel의 아주 작은 이웃 영역만을 고려하는 반면, frequency domain의 Laplacian kernel에서는 전체 이미지의 정보를 활용하여 연산이 수행되기 때문이다.

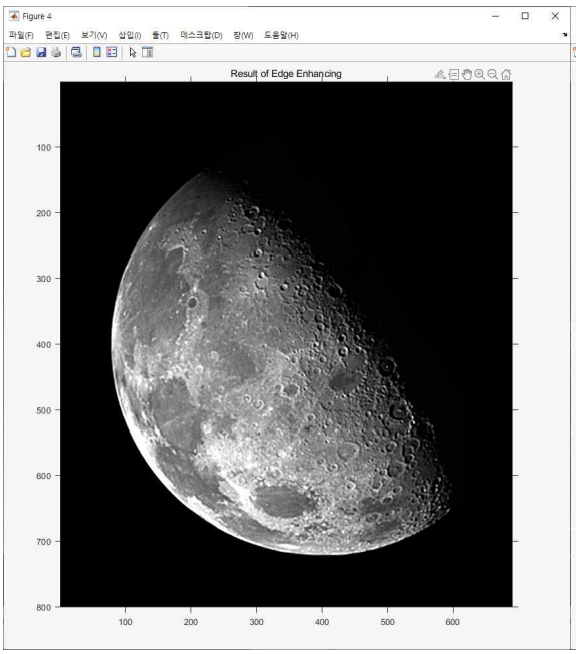

Fig 4. Result of Edge Enhancement in Spatial Domain

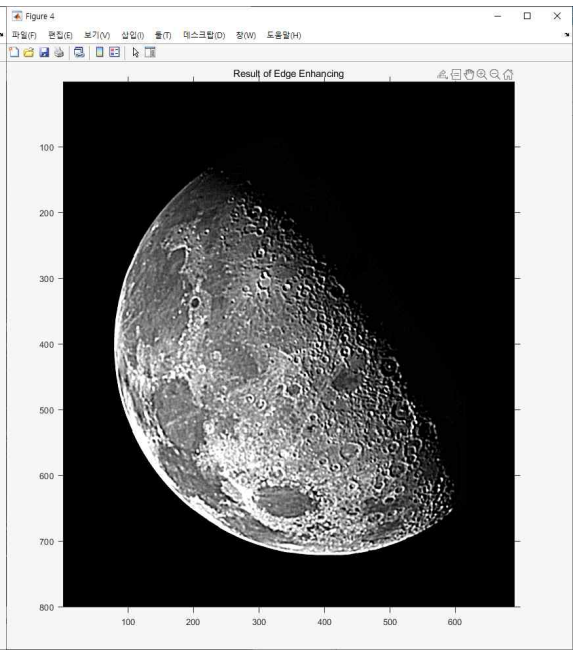

Fig 8. Result of Edge Enhancingment in Frequency Domain

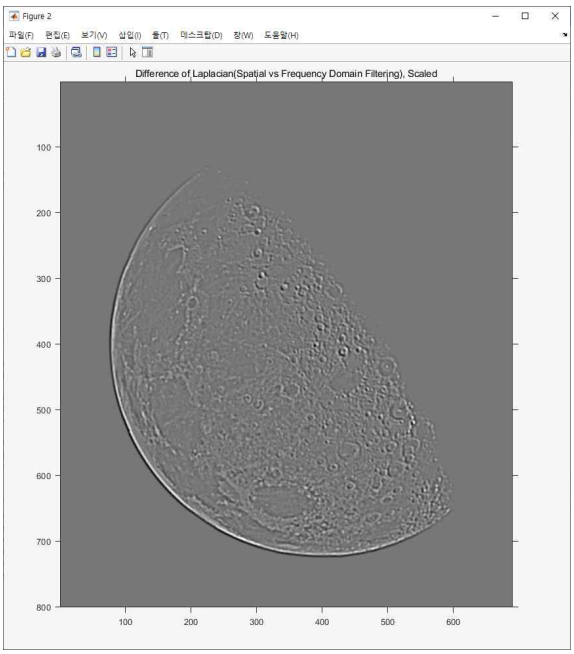

Fig 9. Difference of Laplacian Result (scaled)

3. (a)

MATLAB의 padarray function을 이용하여 symmetric padding을 수행한 결과는 아래와 같다. 좌측 상단의 1024x1024 영역이 원본 이미지의 값이고, 나머지 영역이 padding을 통해 채워진 값이다

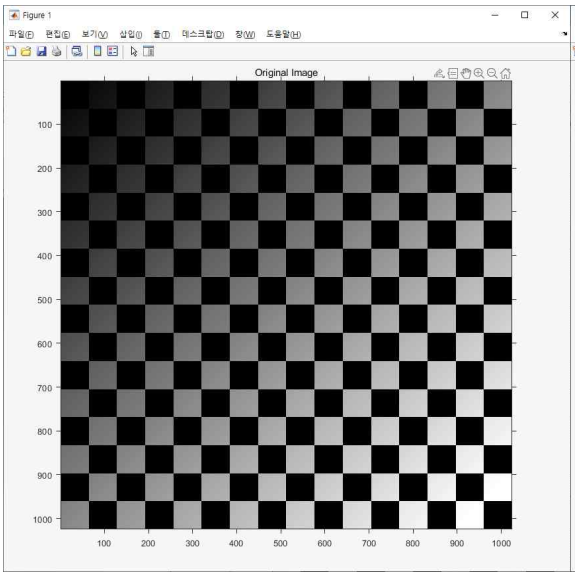

Fig 10. Original Image, 1024 by 1024

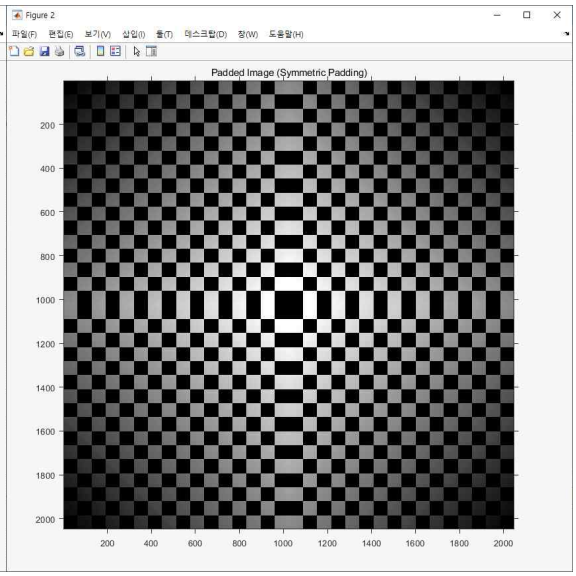

Fig 11. Padded Image (Symmetric Padding), 2048 by 2048

Gaussian Lowpass Filter의 transfer function은 아래와 같다.

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

여기서 $D(u,v)$는 중심에서부터 (u,v)까지의 euclidean distance를 의미하며, D_0는 클수록 더욱 넓은 영역의 주파수를 통과시켜 blur 정도가 약해지게 된다. 다양한 D_0 값(1, 5, 10)을 이용하여 수행한 Gaussian Lowpass Filtering의 결과 및 최종적으로 얻게 된 Shading Correction 결과는 아래와 같다. D_0 값이 커질수록 점점 넓은 영역의 주파수를 통과시키게 된다. 이에 따라, D_0를 10으로 설정하여 계산한 shading pattern에는 체크무늬가 존재함을 확인할 수 있다.

1) $D_0 = 1$

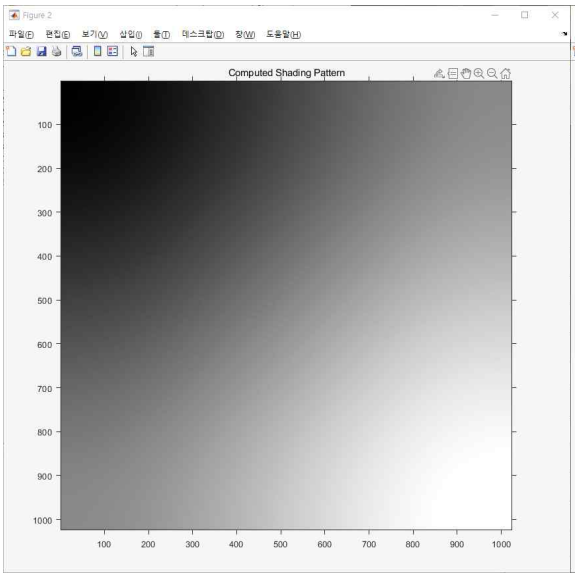

Fig 12. Computed Shading Pattern ($D_0 = 1$)

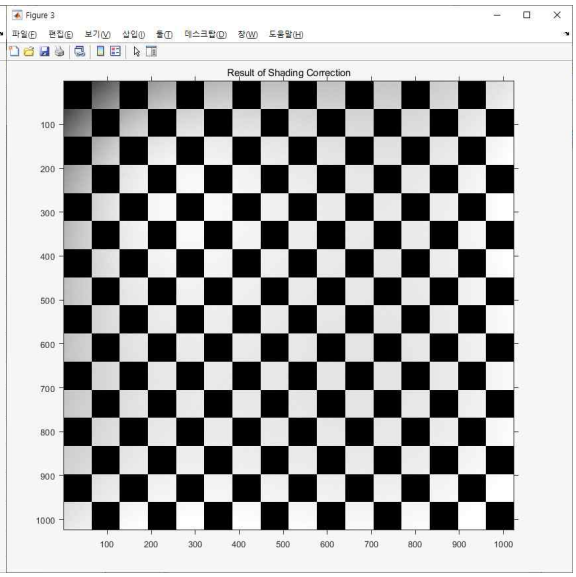

Fig 13. Result of Shading Correction ($D_0 = 1$)

2) $D_0 = 5$

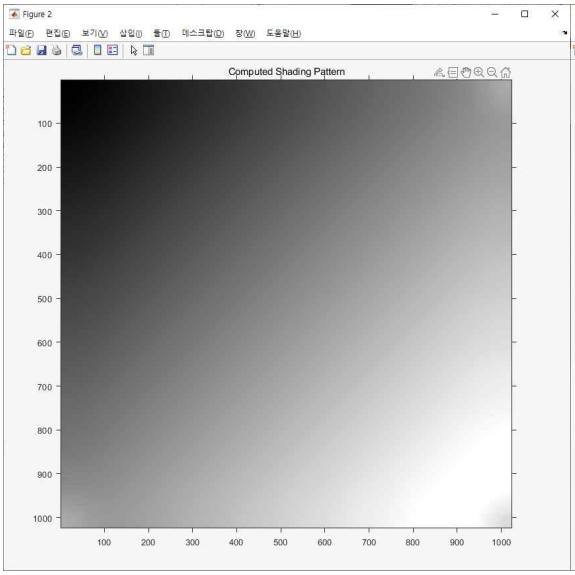

Fig 14. Computed Shading Pattern ($D_0 = 5$)

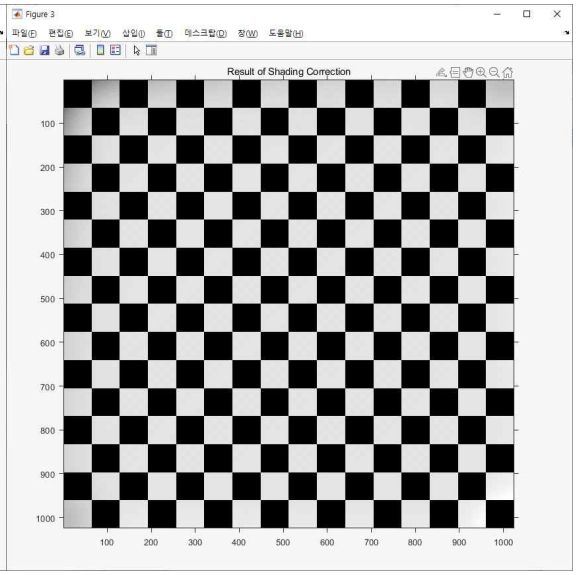

Fig 15. Result of Shading Correction ($D_0 = 5$)

3) $D_0 = 10$

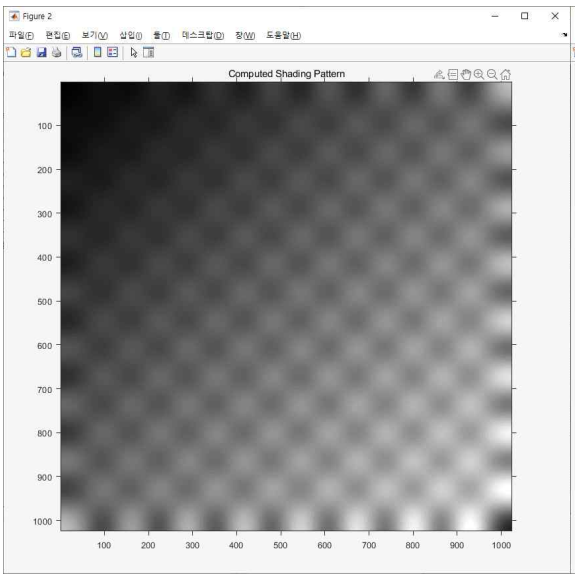

Fig 16. Computed Shading Pattern ($D_0 = 10$)

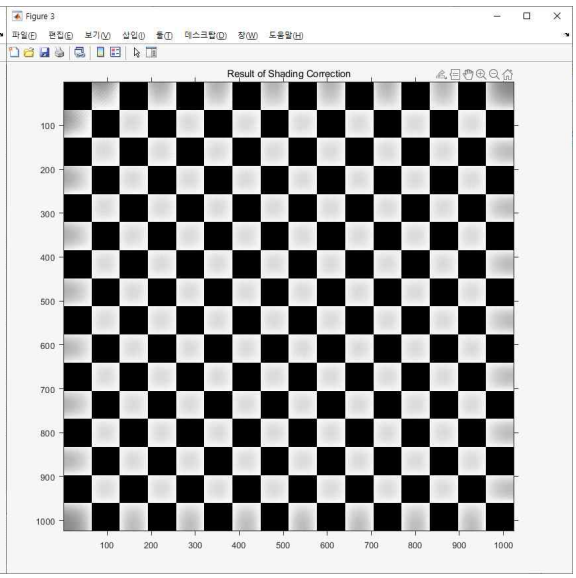

Fig 17. Result of Shading Correction ($D_0 = 10$)

3. (b)

MATLAB에서 복소수의 크기는 abs() function을 이용하여 계산할 수 있다. Frequency domain에서 image의 power를 구하는 과정은 아래와 같다.

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

$$P_{Total} = \sum_{u=0}^{P-1}\sum_{v=0}^{Q-1} P(u,v)$$

문제에서 요구한 사라진 power의 percentage를 구하기 위해서는 padded image에 FFT를 적용한 결과의 전체 power 값(코드상의 P_padded)과 Gaussian LPF를 적용한 후의 전체 power 값(코드상의 P_result)을 계산해야 하고, 이는 위의 수식을 통해 계산할 수 있다. 이후, P_padded에 대한 P_result의 percentage는 아래와 같이 계산할 수 있다.

$$percentage = 100 * \frac{P_{result}}{P_{padded}}$$

최종적으로, MATLAB을 활용해 $D_0 = 1, 5, 10$인 경우에 대해 계산한 결과는 아래와 같다.

$D_0 = 1, 5, 10$인 경우 각각 약 7.1%, 45.9%, 49.5%의 power를 통과시키므로, 각각 약 92.9%, 54.1%, 50.5%의 power를 제거하였다는 것을 알 수 있다.