Compressing Pictures using 2D DFT and Inverse DFT

Submitted By:

Shyam Rauniyar

M13944385

AEEM Fall 2020

MATH 6012

Submitted To:

Dr. Yizao Wang

Faculty of Applied Linear Algebra

MATH 6012

University of Cincinnati

Outline of Presentation

- Goals of Project
- About 2D Discrete Fourier Transform (DFT) and Inverse DFT
- Plots of Original Image and its Fourier Coefficients
- Plots of Few Basis Vectors
- Experiment 1: High or Low Pass Filter / (Scaled)
- Experiment 2: High or Low Region Filter
- Experiment 3: Directional Filters

Goals of Project

- Import a picture ($size\ m \times n$)and determine its 2D Fourier Coefficients. Plot them side by side
- Plot a few basis vectors, both low and high frequencies
- Apply some filter experiments on the Fourier Coefficients and apply inverse DFT to see the experimental effects

About 2D DFT and Inverse DFT

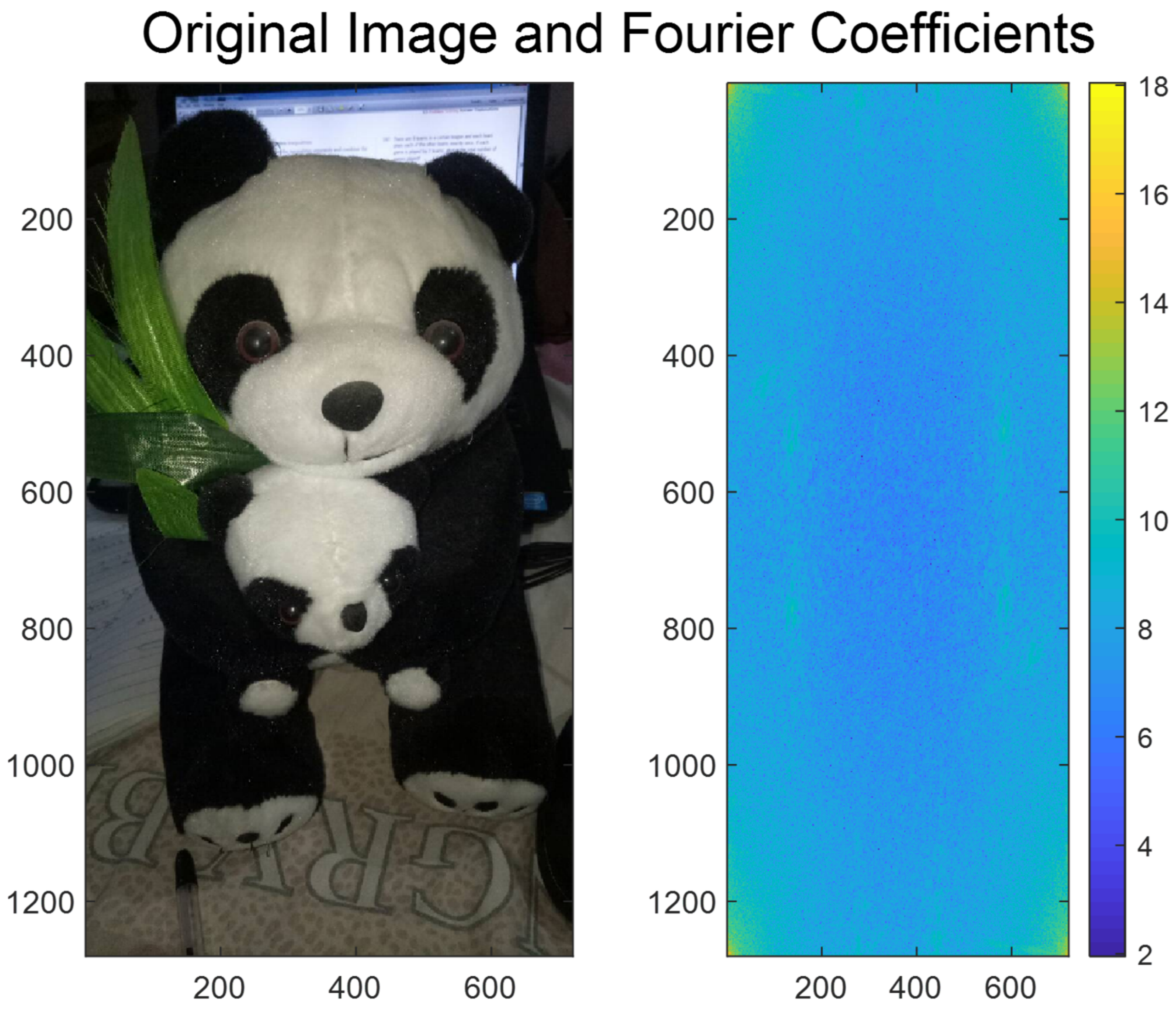
- A 2D signal ($\boldsymbol{f}$)can be decomposed into its corresponding frequencies using 2D DFT with Fourier Coefficients $\hat{\boldsymbol{f}}$:
  - $\hat{f}_{j,k} = \left\langle \boldsymbol{f}, \boldsymbol{\omega}^{(j,k)} \right\rangle = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} f_{u,v} \overline{\omega}_{u,v}^{(j,k)} = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} f_{u,v} e^{-2\pi i \left(\frac{ju}{m} + \frac{kv}{n}\right)}$

About 2D DFT and Inverse DFT

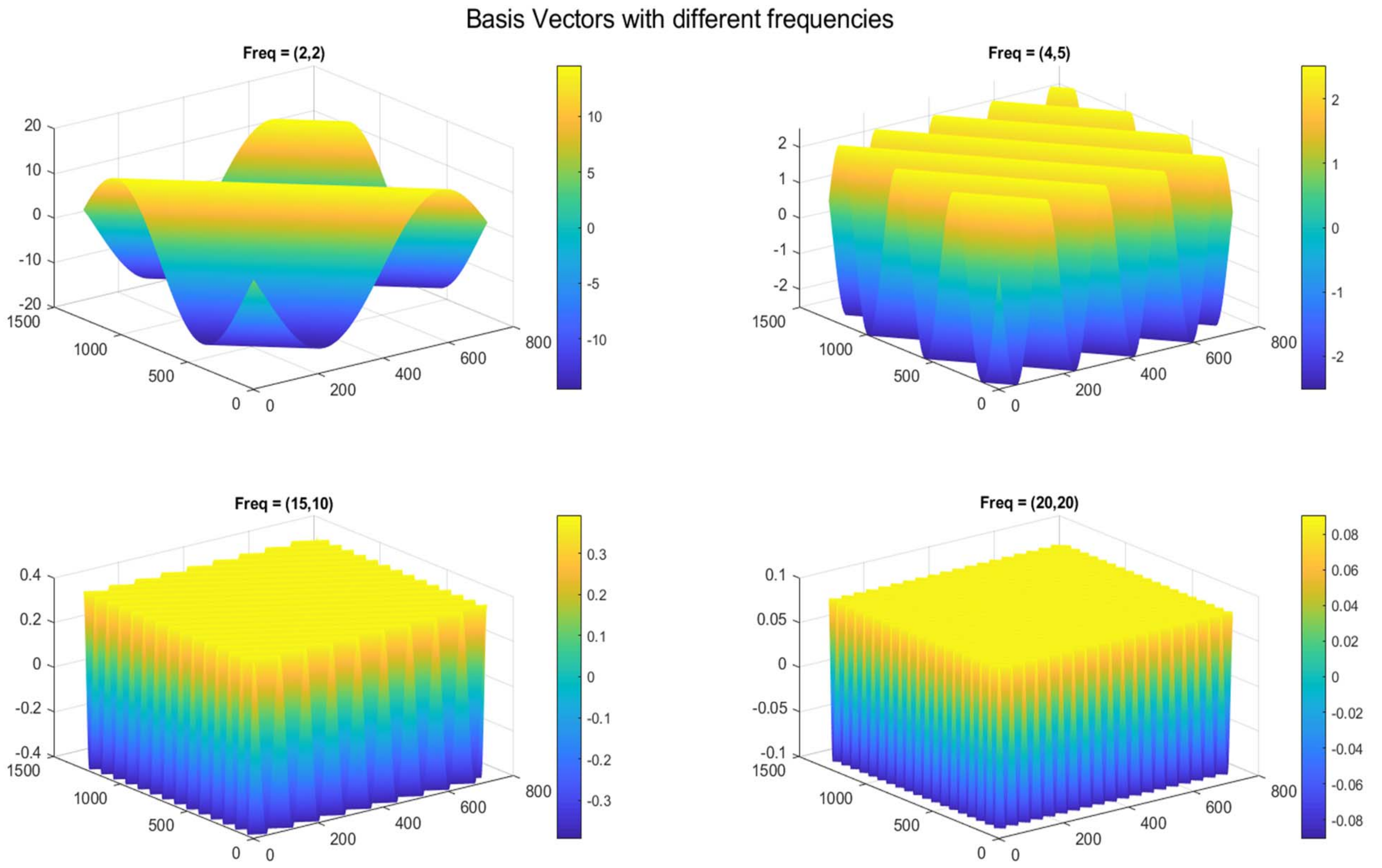
- A combination of 2D DFT coefficients can be applied with inverse DFT to get the original signal as:
  - $f_{j,k} = \left(\sum_{u,v} \hat{f}_{u,v}\omega^{(u,v)}\right)_{j,k} = \sum_{u=0}^{m-1}\sum_{v=0}^{n-1} \hat{f}_{u,v}\omega^{(u,v)} = \sum_{u=0}^{m-1}\sum_{v=0}^{n-1} \hat{f}_{u,v}e^{2\pi i\left(\frac{ju}{m}+\frac{kv}{n}\right)}$

Methods Applied

- Used Matlab for the project
- Imported image using $imread()$
- Plotted the image using $imagesc()$ for color image
- Obtained the Fourier coefficients vector using Fast Fourier Transform, $\mathrm{B_t} = fft2()$
- Plotted the gray image Fourier Coefficients (as $\log(abs(B_t) + 1)$
- Created an index matrix to eliminate or scale the chosen Fourier Coefficients and multiplied it to B_t
- Applied inverse DFT using $ifft2()$ and converted the resulting matrix to 8-bit integer for plots using $uint8()$
- Plotted the final image alongside its gray image Fourier Coefficients to show applied modifications using using imagesc()



Imported Image and Fourier Coeff.



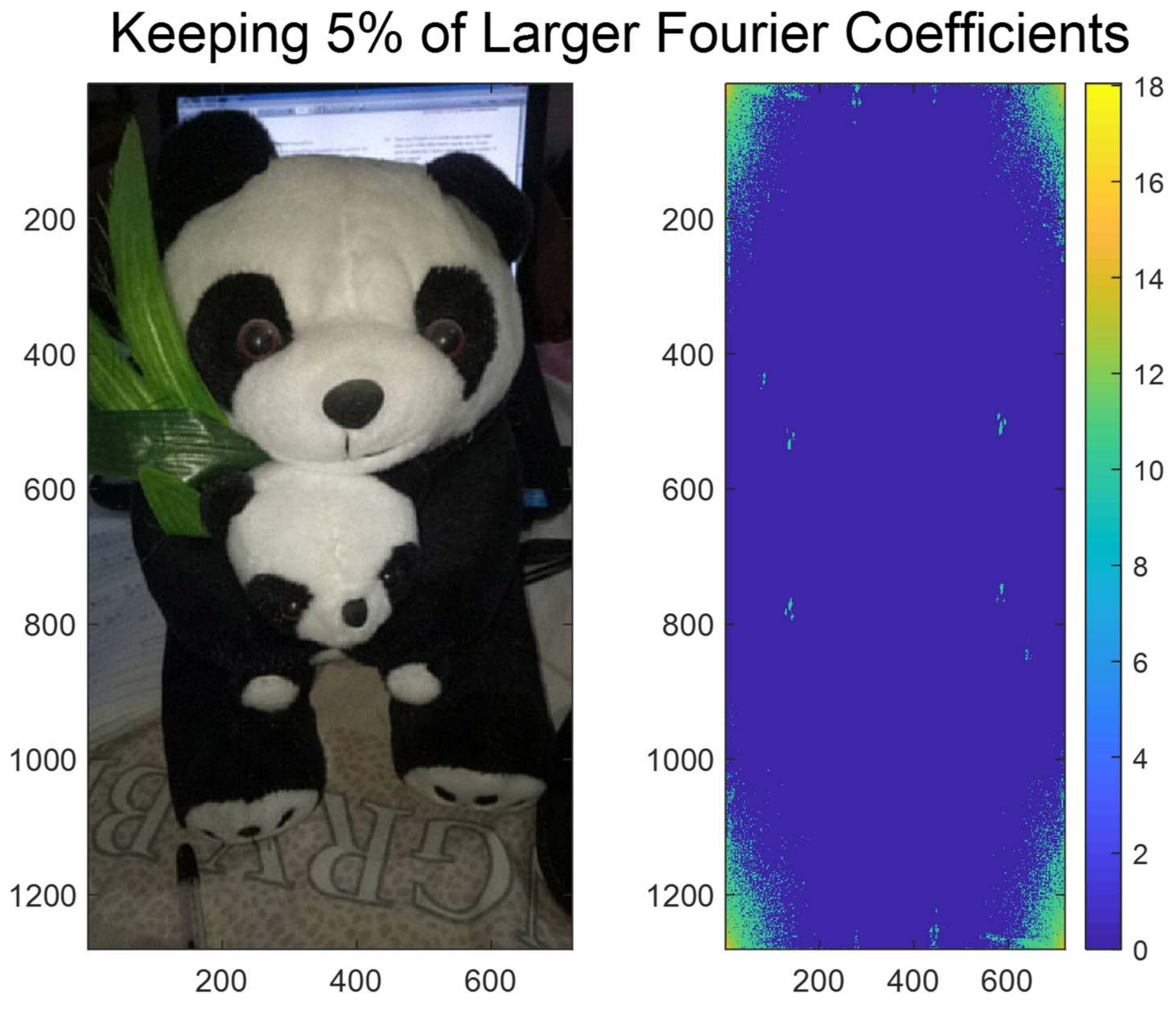
2D BASIS VECTORS (HIGH AND LOW FREQ)

High or Low Pass Filter

Eliminating low frequency (central) coefficients (High Pass) or

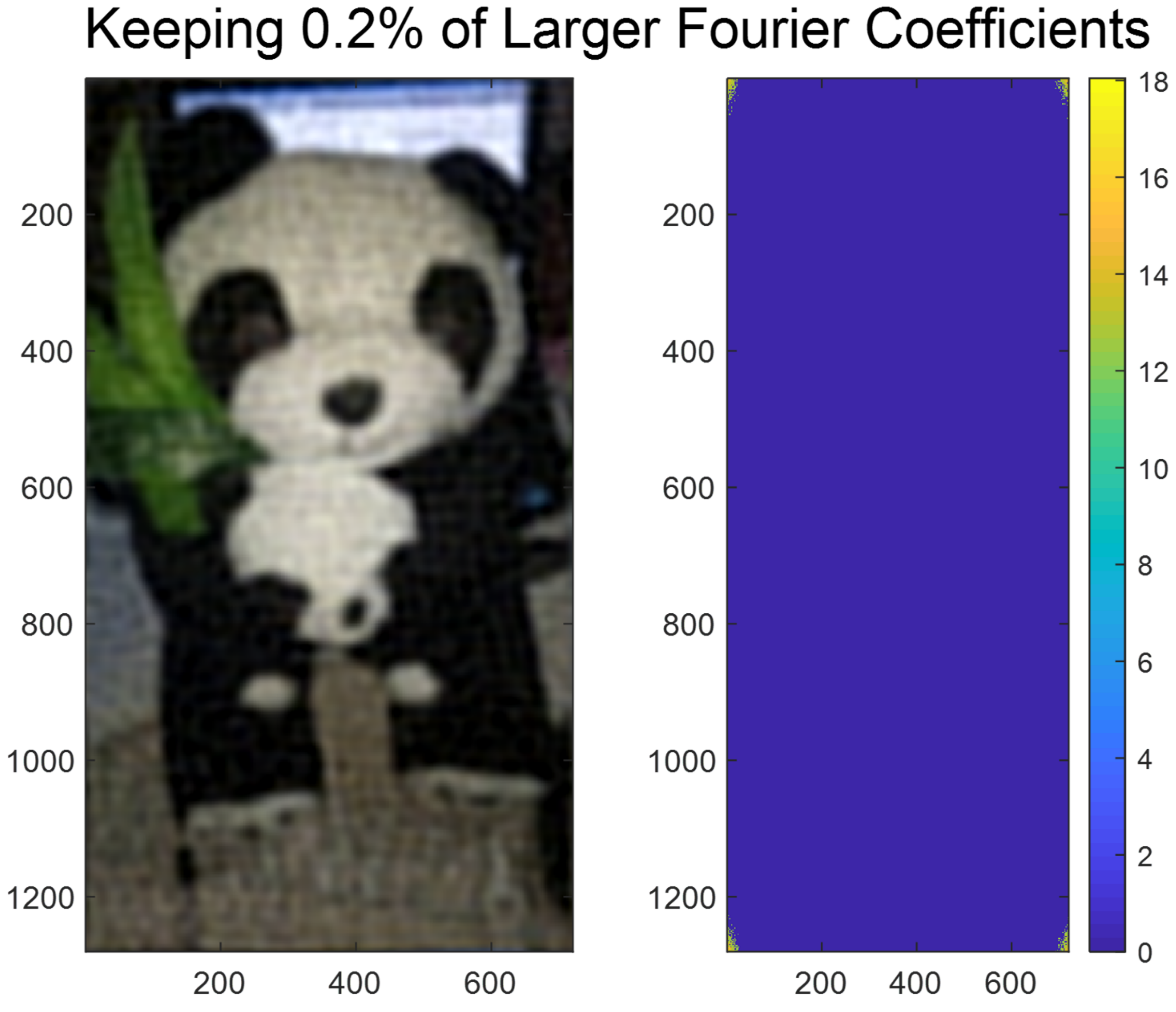
Eliminating high frequency (corner) coefficients (Low Pass)

Scaling the coefficients



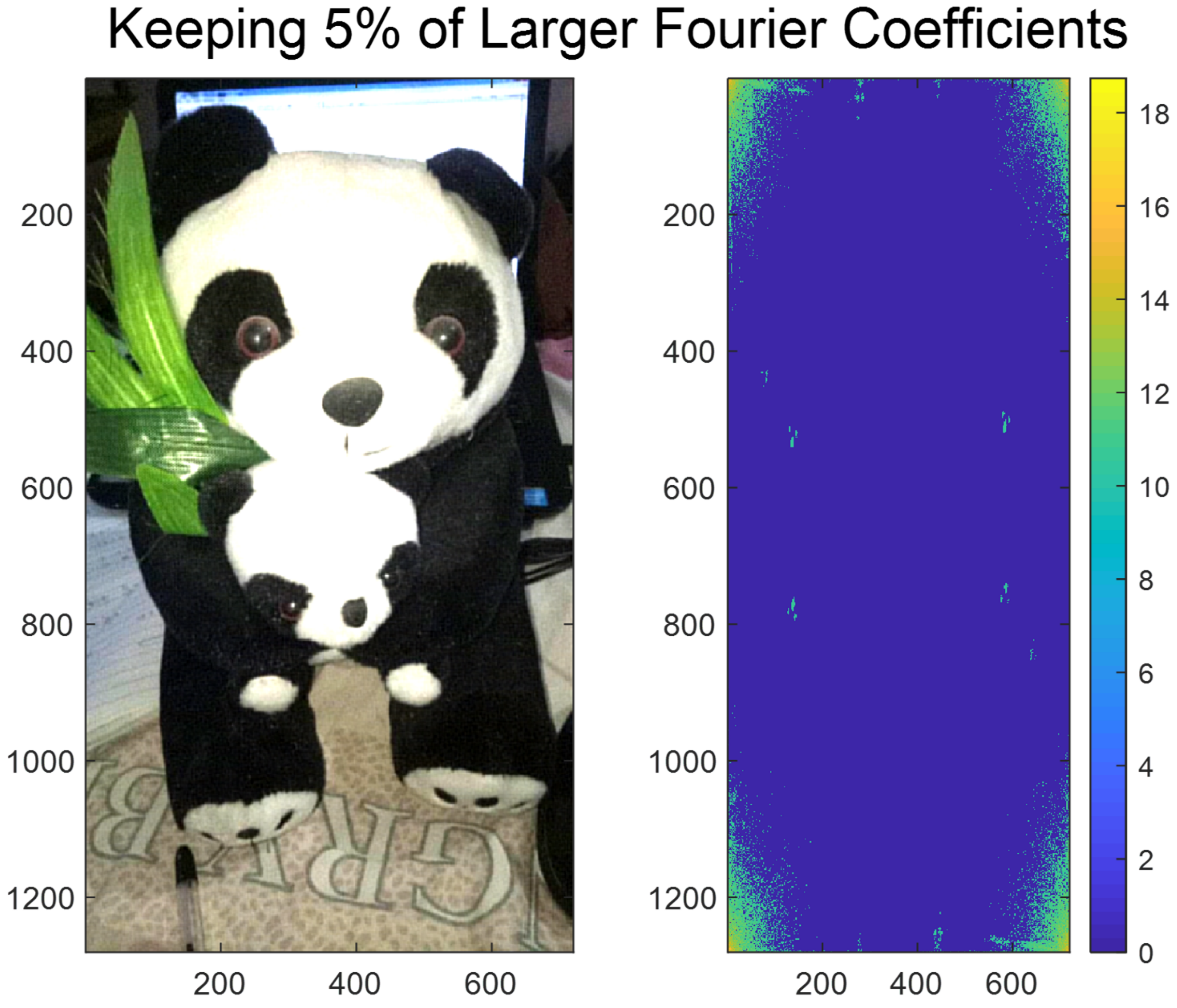
HIGH PASS FILTER

STILL SHARP



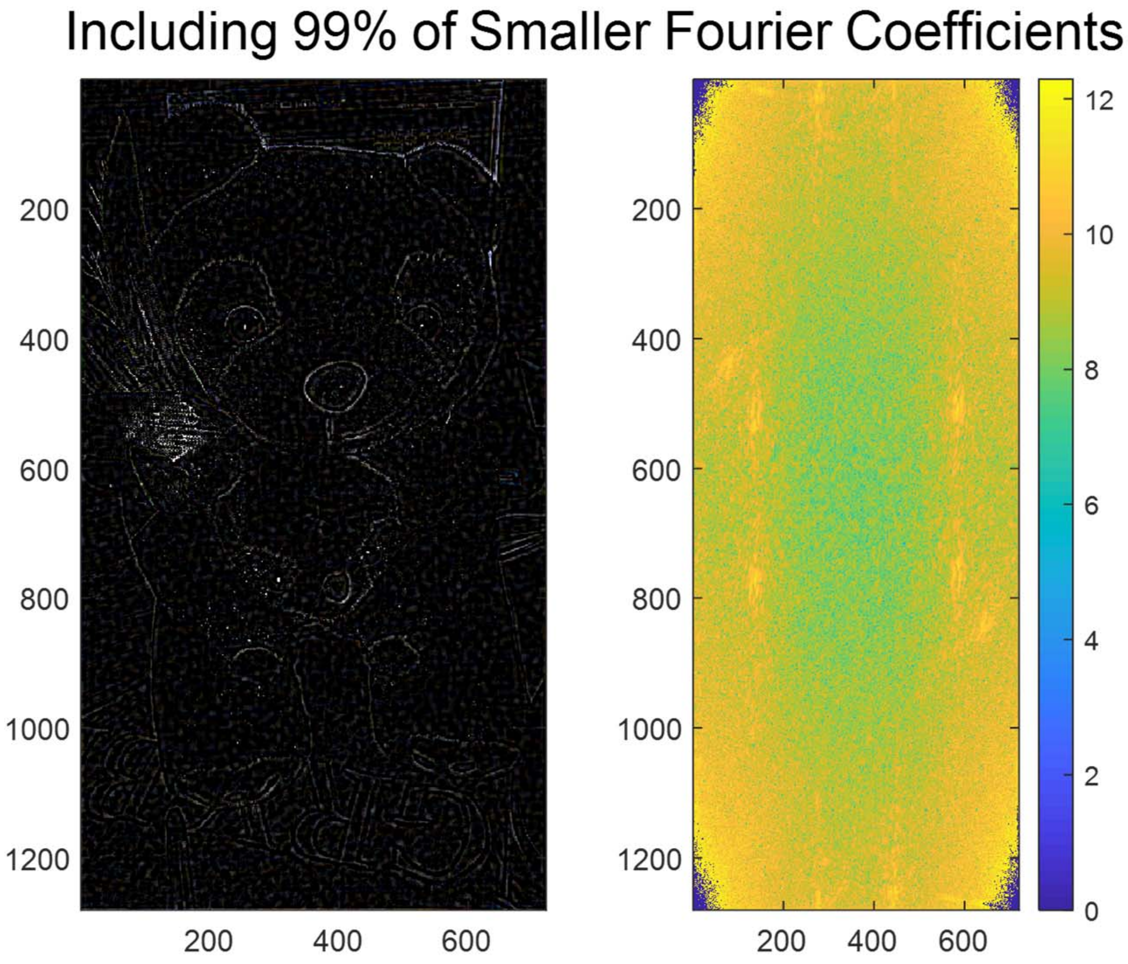
HIGH PASS FILTER

BLURRED IMAGE



HIGH PASS FILTER (SCALED)

BRIGHTER IMAGE AT SCALE = 2



LOW PASS FILTER (SCALED)

OUTLINE VISIBLE
WITH SCALE = 4

High or Low Region Filter

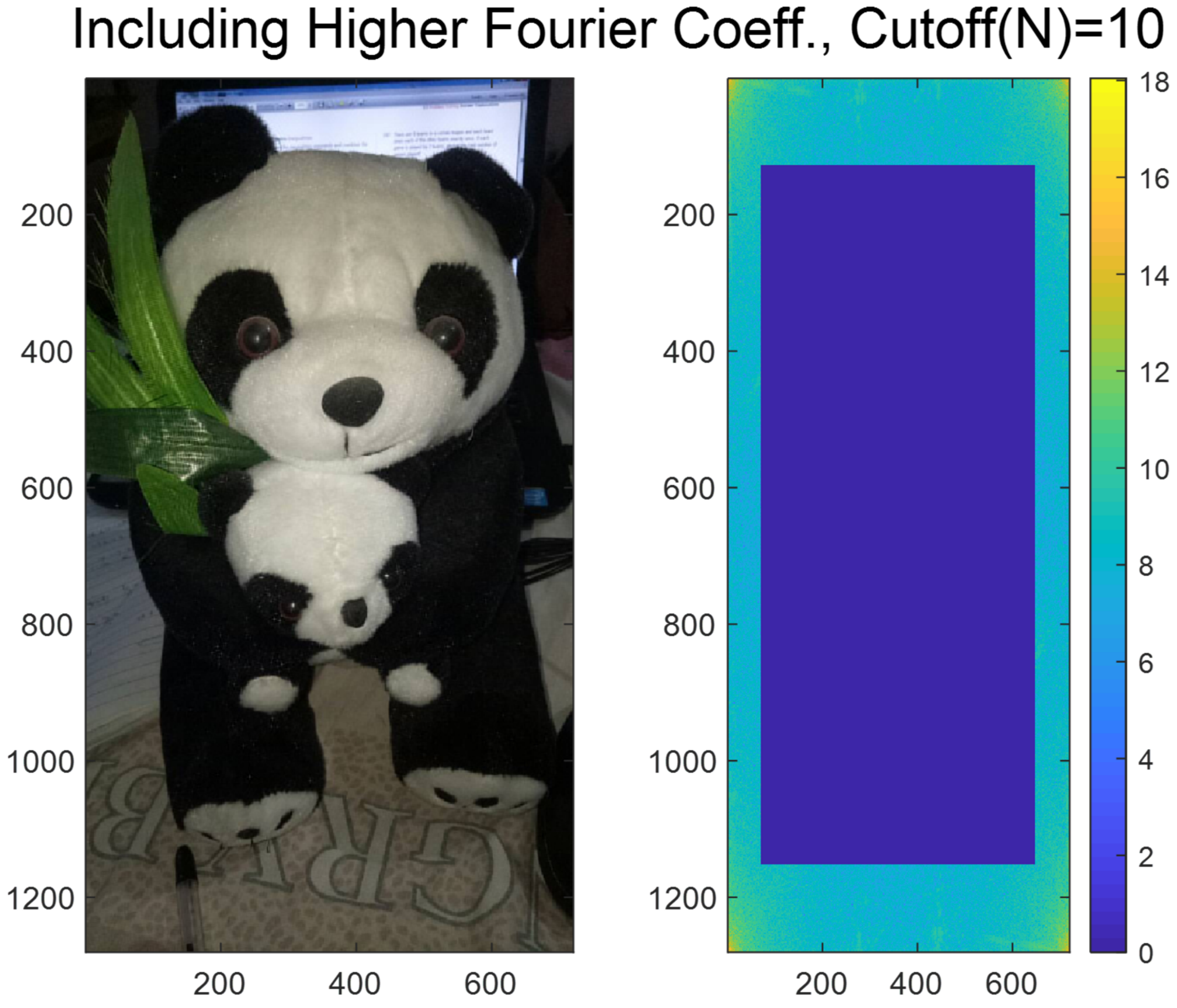
Eliminating central region (rectangular) coefficients (High Region) or

Eliminating outlying region (rectangular) coefficients (Low Region)

Fourier Coefficients $(\hat{f}_{u,v})$ of following region is modified:

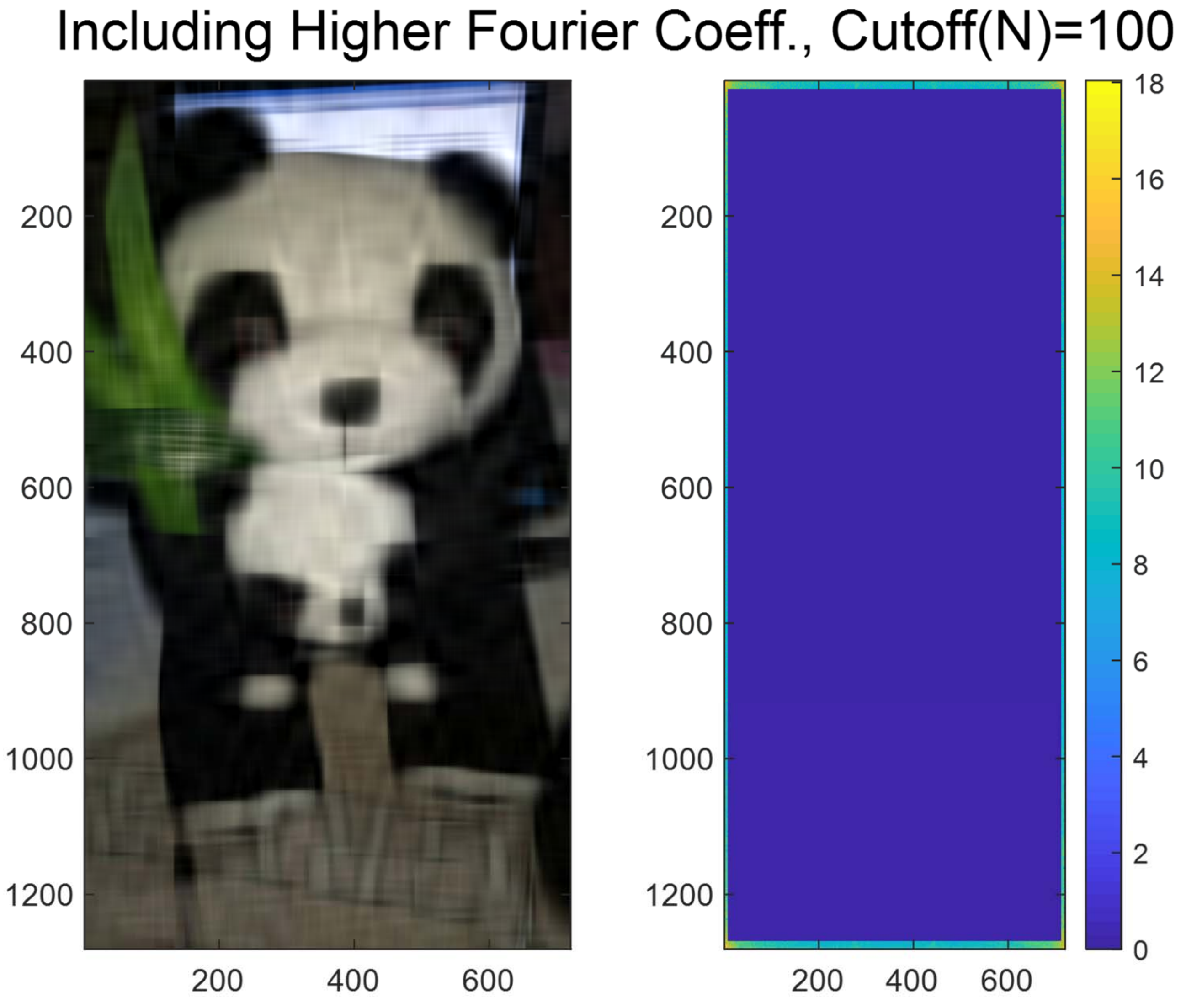
$$j, k \in \left(\frac{m}{N}, \frac{(N-1)m}{N}\right) \times \left(\frac{n}{N}, \frac{(N-1)n}{N}\right)$$

Scaling the coefficients



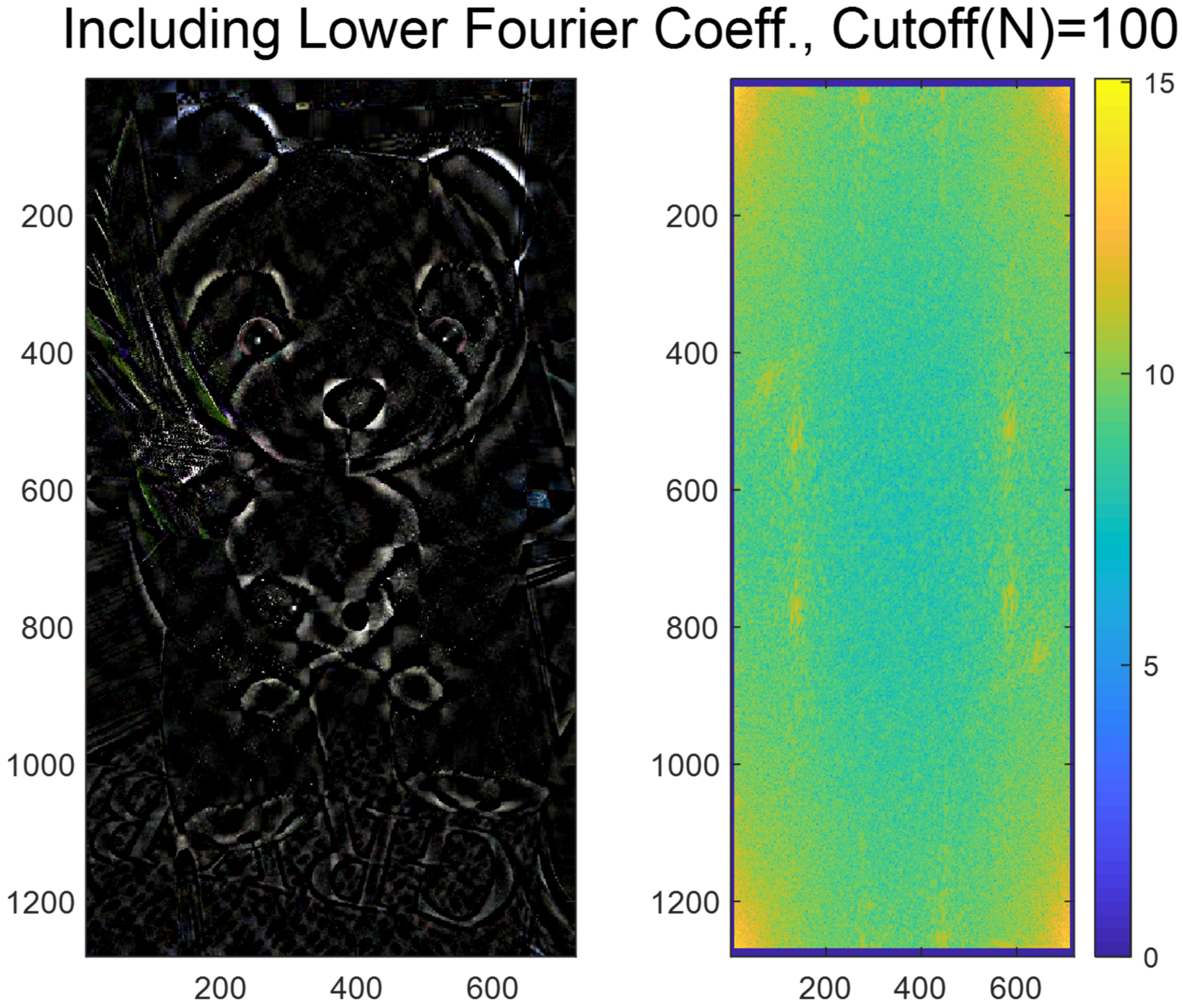
HIGH REGION FILTER

LOW CUTOFF, SHARP IMAGE



HIGH REGION FILTER

HIGH CUTOFF,
BLURRED IMAGE



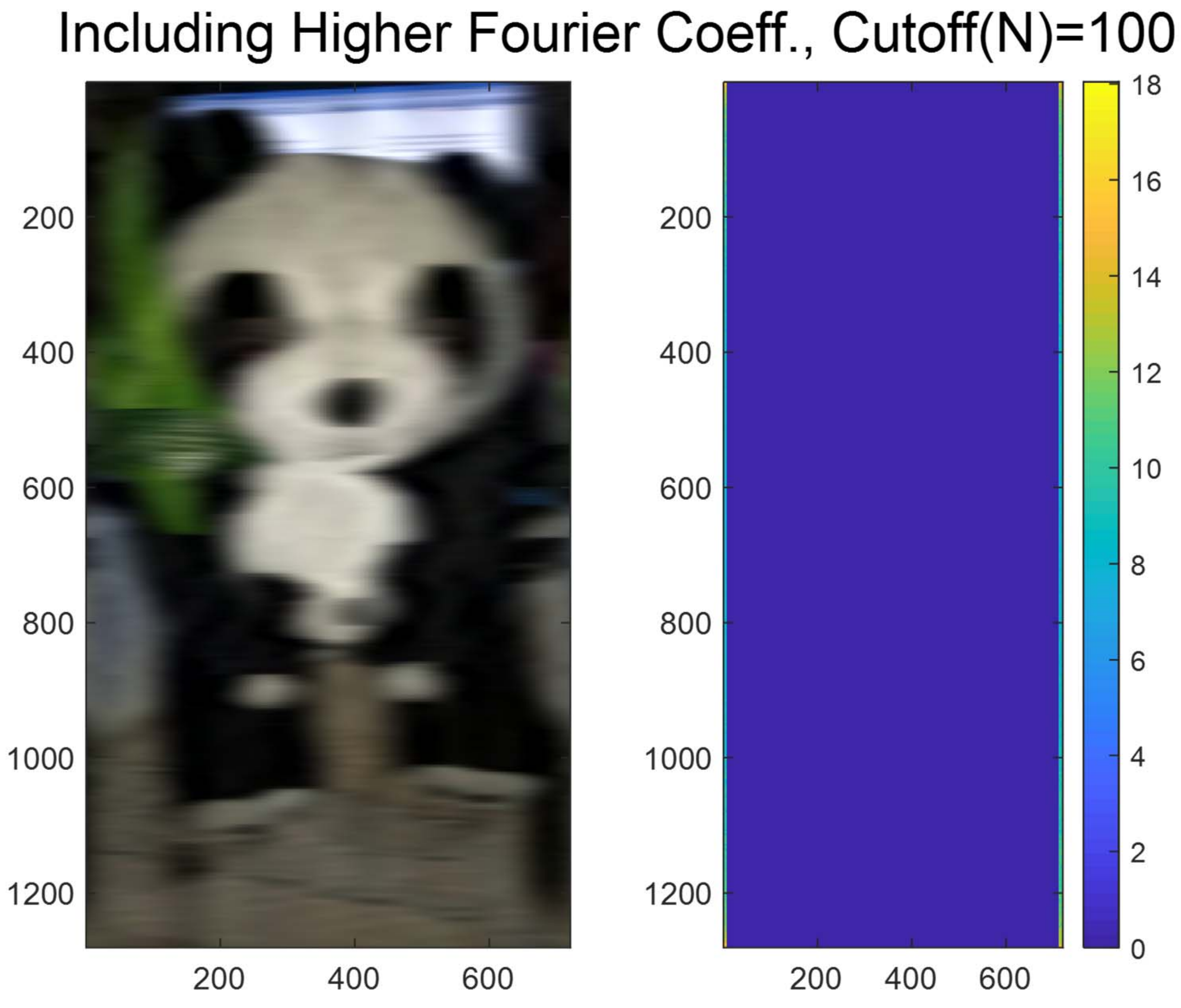
LOW REGION FILTER (SCALED)

OUTLINE VISIBLE AT SCALE = 4

Directional Filter

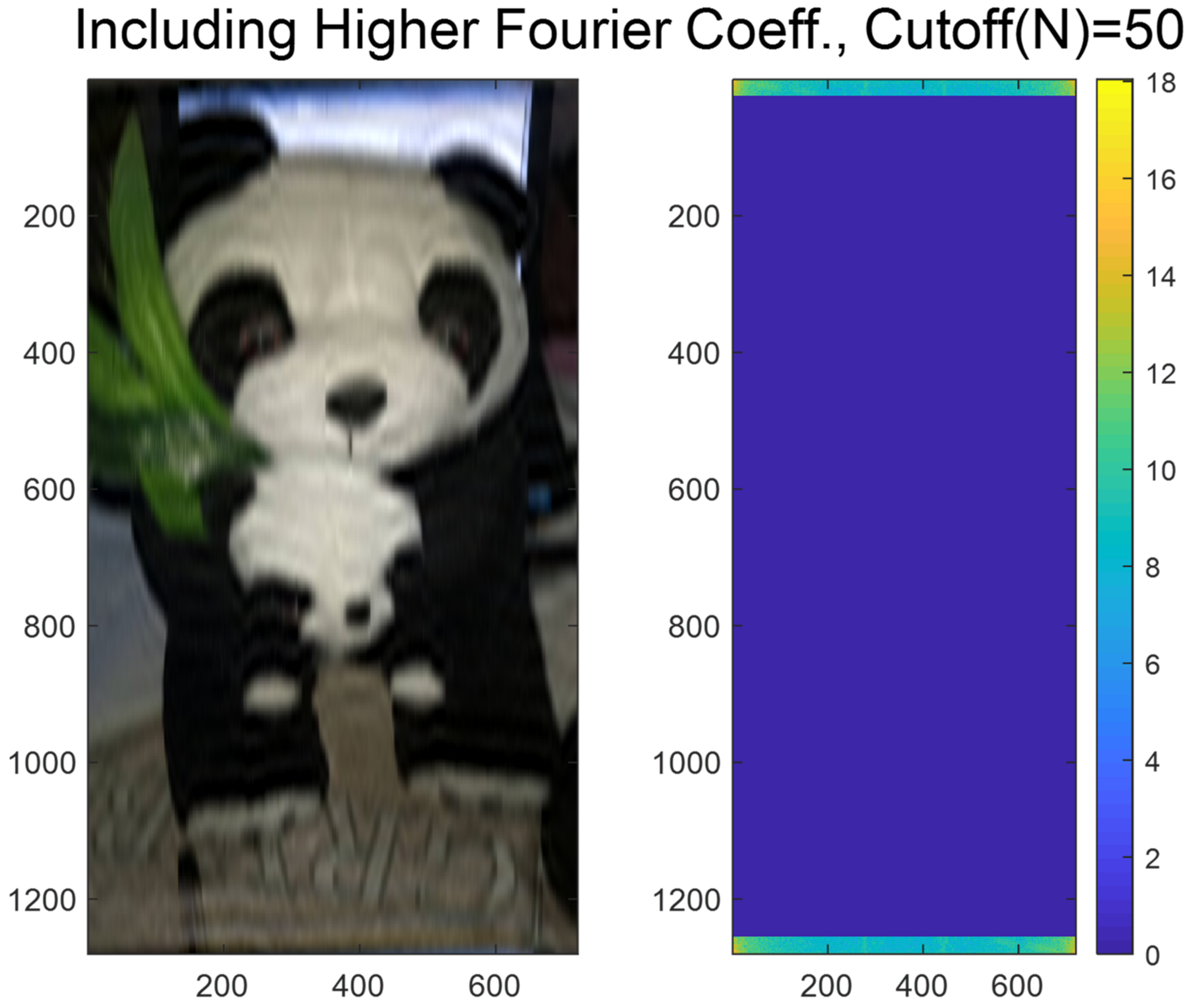
Eliminating Left/Right band of frequency coefficients or

Eliminating Top/Down band of frequency coefficients



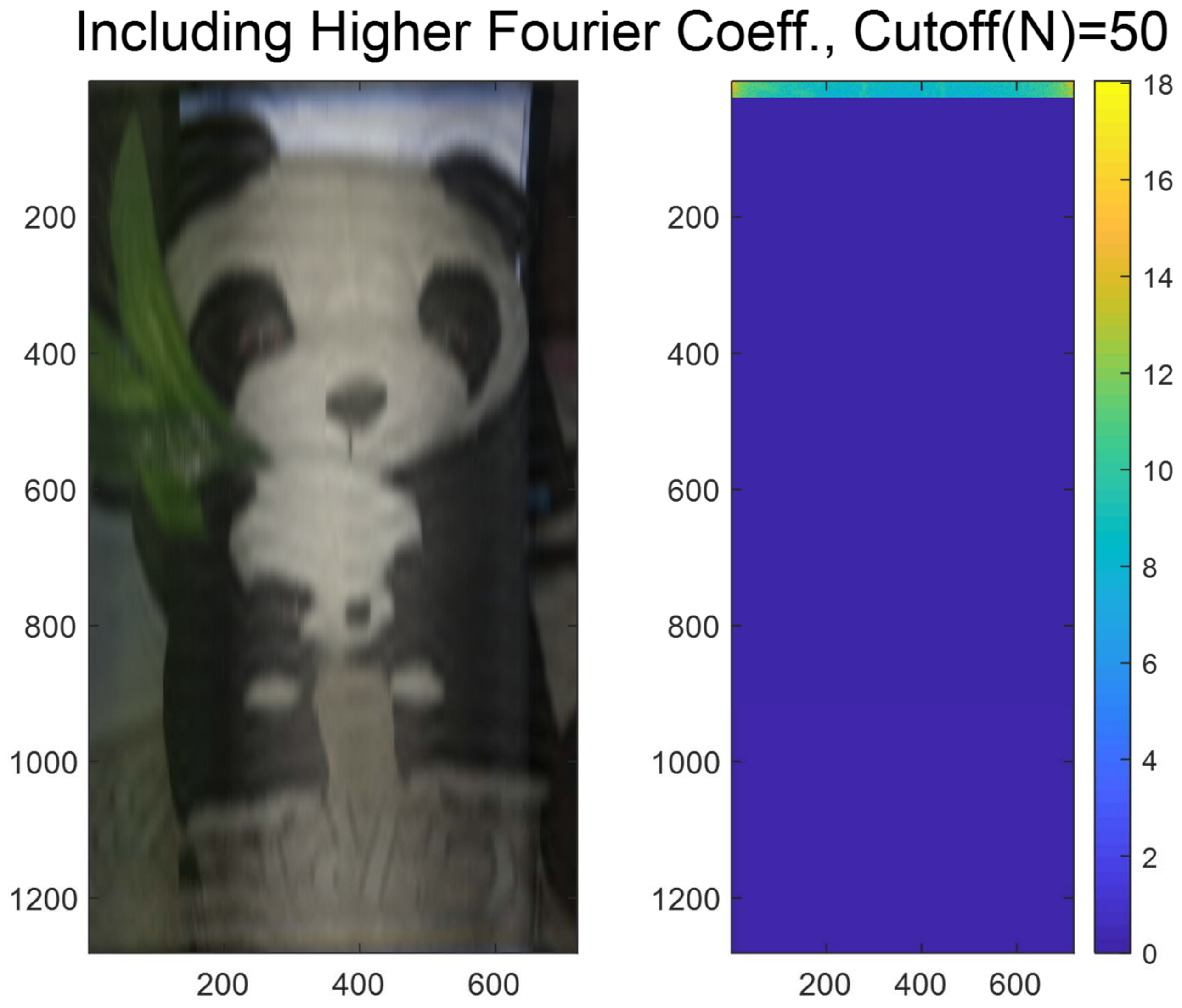
LEFT/RIGHT REGION FILTER

BLURRY HORIZONTALLY



TOP/DOWN REGION FILTER

BLURRY VERTICALLY



TOP REGION FILTER

VERTICAL CENTRALLY DISFIGURED

Concluding Remarks

- Filtering out some Fourier coefficients reduces the quality of image
- Quality of image remains intact if high frequency coefficients are kept for compression
- Compression with low frequency coefficients reduces quality significantly
- Scaling of Fourier coefficients causes sharpening/brightening of image
- Unidirectional filter disfigures the image in corresponding direction

Thank You!!