CS270 Homework1

Question 1

Algorithm Description

In order to remove the haze in given image, I've tried several methods, and some combinations of these methods:

- Histogram equalization (HE)
- Contrast limited histogram equalization (CLHE)
- Contrast limited adaptive histogram equalization (CLAHE)
- Dark channel prior
- Gamma transformation

I implement each algoritm by opencv-python + numpy manually, the procedure are shown below.

- **Histogram equalization (HE)**
    - First, split the image into 3 channels (RGB).
    - For each channel, calculate the histogram:

$$Hist[k] = N(img[i,j] = k) = \sum_i \sum_j 1_{\{img[i][j]==k\}}$$

    - Do the HE operation for each channel:

$$s = T(r) = \frac{L-1}{M \cdot N} \sum_{j=0}^{k} n_j$$

- Merge 3 channels
- **Contrast limited histogram equalization (CLHE)**
    - First, split the image into 3 channels (RGB).
    - For each channel, calculate the histogram.
    - Clip the histogram by `threshold`.
        - If `Hist[k] > threshold`, assign `Hist[k]` to `threshold`.

        - Otherwise, compensate the excess part to `Hist[k]`.
    - Do HE operation to the clipped histogram.
    - Merge 3 channels
- **Contrast limited adaptive histogram equalization (CLAHE)**
    - First, split the image into 3 channels (RGB).
    - For each channel:
        - Use a `W*W` sliding window to traverse the image, do CLHE to this window and update `M*M` centeral pixels.
        - Use bi-linear inrerploiation to reduce "grid effect"
    - Merge 3 channels
- **Gamma Transformation**
    - First, split the image into 3 channels (RGB).
    - Create a lookup table:

$$LUT(x) = c \cdot x^{\gamma}$$

    - Apply the lookup table to the image for each channel:

$$Output[i, j, ch] = LUT[Input[i, j, ch]]$$

    - Merge 3 channels
- **Dark Channel Prior**
    - This method is introduced in *Single Image Haze Removal Using Dark Channel Prior - CPVR 2009*
    - The dark channel is defined as:

$$Dark(x) = \min_{y \in \Omega(x)} (\min_{c \in [r,g,b]} Img(y))$$

    - The basic idea is, the dark channel in non-foggy images is an almost all black image.
    - The basic equation is:

$$I(x) = J(x)t(x) + A(1 - t(x))$$

  where `I(x)` is the foggy image, `J(x)` is the output image, `t(x)` is the transmission
  rate, `A` is the global atmospheric light composition.

- `A` can be calculated as:
  - Find pixels with rank 0.1% highest intensity in Dark channel of Input image.
  - Find the highest intensity in input image at these positions, take the average to be the global atmospheric light composition (A).
- `t(x)` can be calculated by:

$$t(x) = 1 - \omega Dark(\frac{Img}{A})$$

where `w` is a parameter ranges from 0 to 1.

- Hence, `A` and `t(x)` are known, we can obtain the output image:

$$Output(x) = \frac{Input(x) - A}{\max\left(t(x), t_0\right)} + A$$

where `t0` is a parameter (I set it to 0.1)

In order to add haze to the input image, I used inverse Contrast-Stretching Transformation or Gamma Transformation first

- **Gamma Transformation**
  - Already introduced
- **Inverse Contrast-Stretching Transformation**
  - First, split the image into 3 channels (RGB).
  - For each channel:
    - Calculate the mean value m
    - Create a lookup table:

$$LUT(x) = \frac{m}{(1-x)^E}$$

    - Apply the lookup table to the image for each channel:

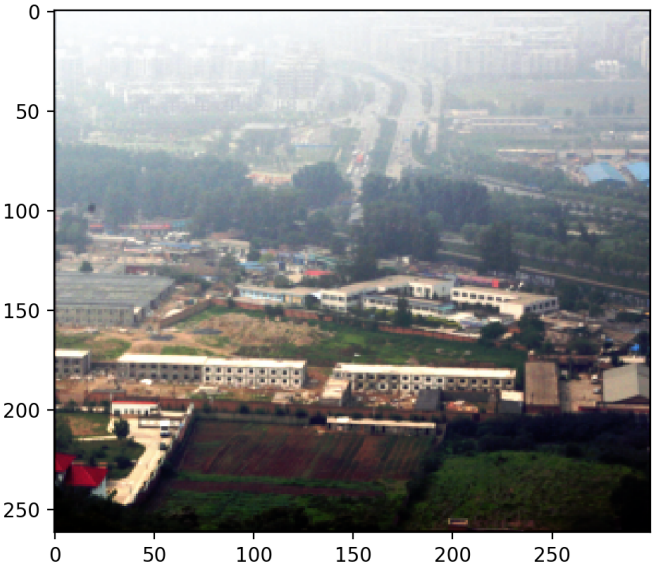
$$OutputCh[i,j] = LUT[InputCh[i,j]]$$

  - Merge 3 channels
    Then, Add a bias to each pixel to make the image looks better:

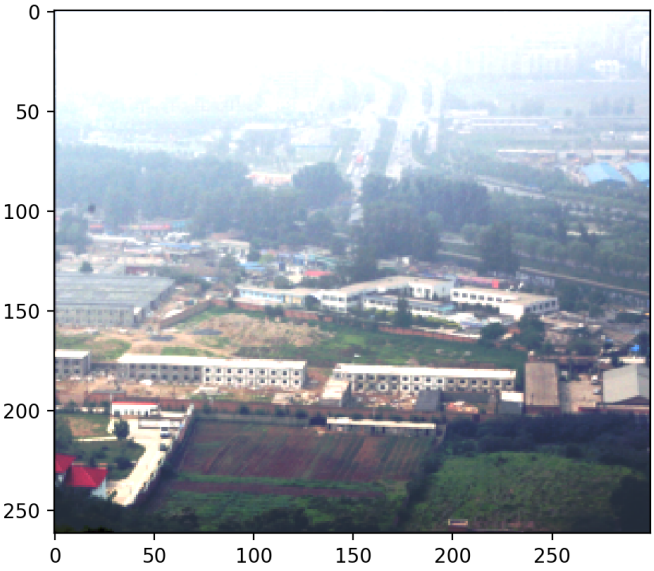
$$img[i, j, ch] \leftarrow img[i, j, ch] + Bias$$

Results

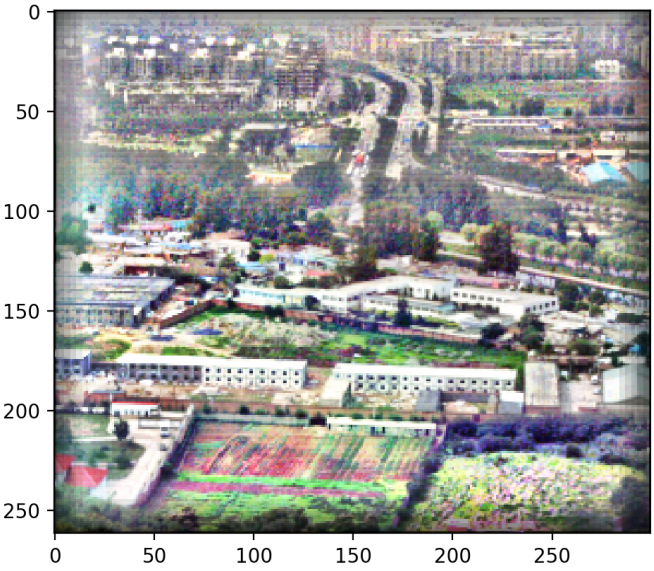
- Dehaze: HE



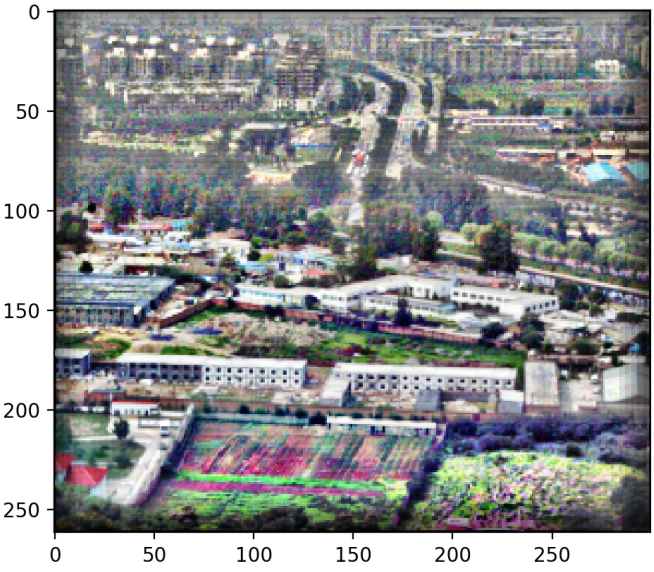
- Dehaze: CLHE (threshold = 0.013)



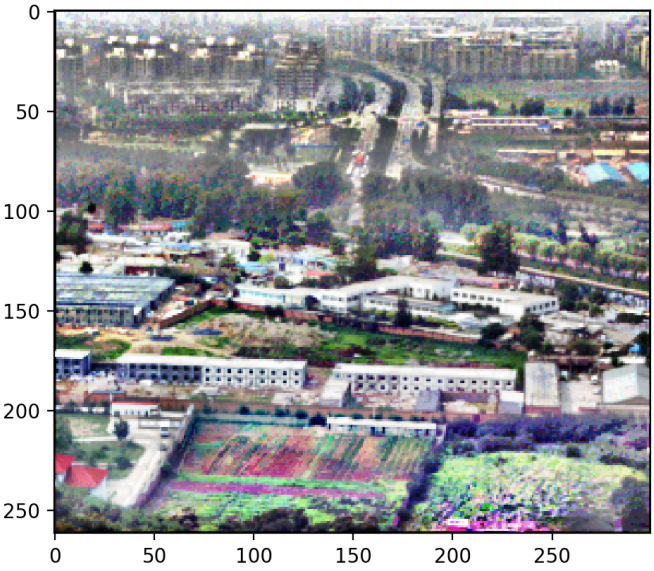
- Dehaze: CLAHE (threshold = 0.015)



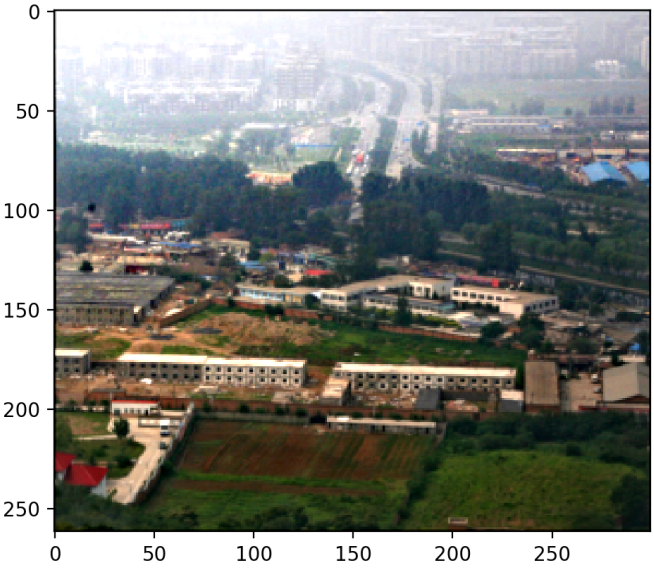
- Dehaze: CLAHE (threshold = 0.020)



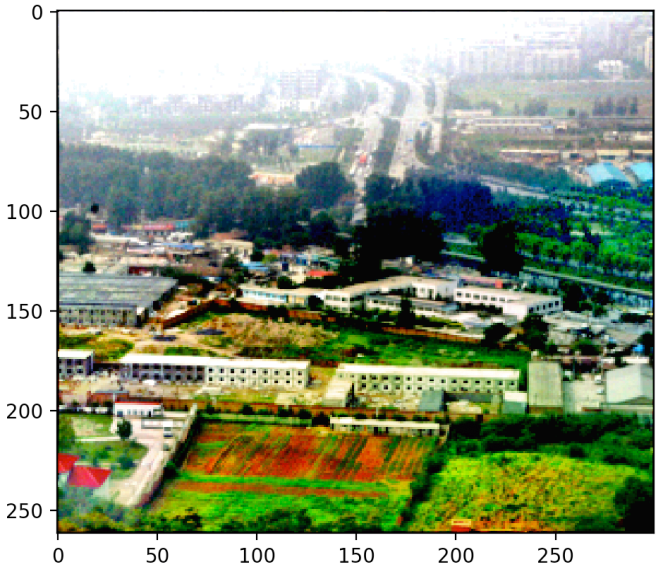
- Dehaze: Gamma Transformation + CLAHE (threshold = 0.02)



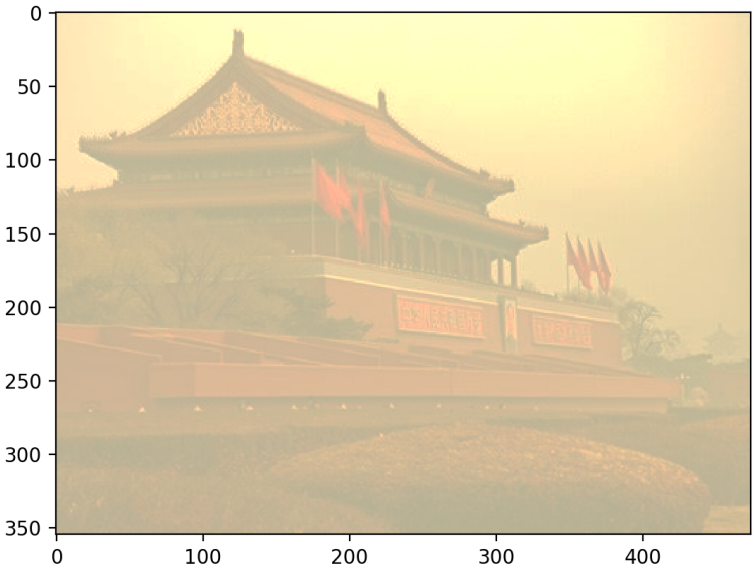
- Dehaze: Dark channel prior(w=0.97)



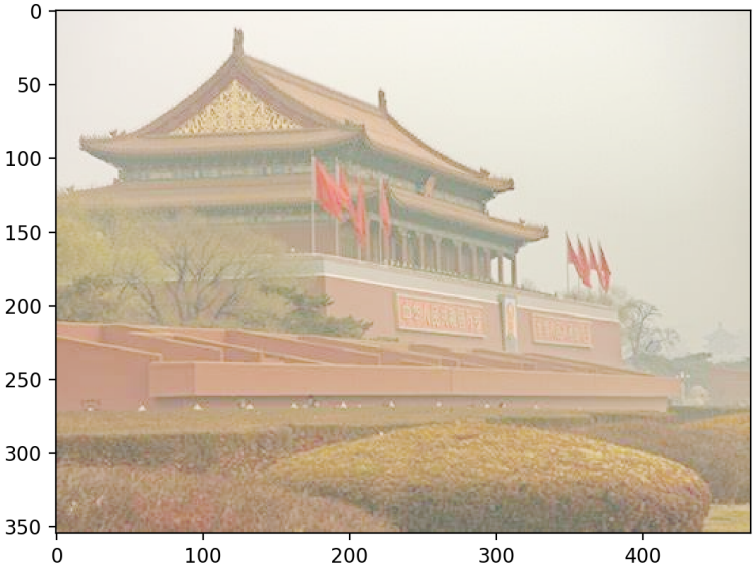
- Dehaze: Dark channel prior + CLAHE



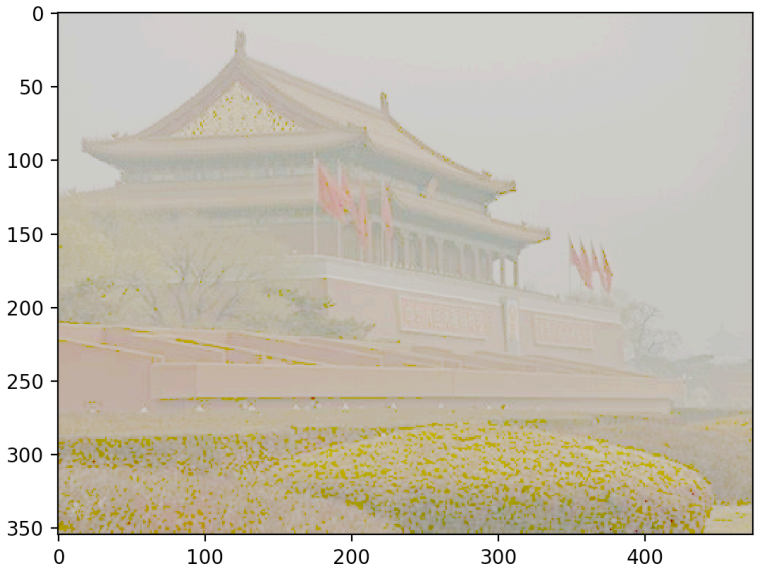
- Add Fog - Inverse Contrast-Stretching Transformation



- Add Fog - Gamma Transformation



- Add Fog - Gamma Transformation + Bias

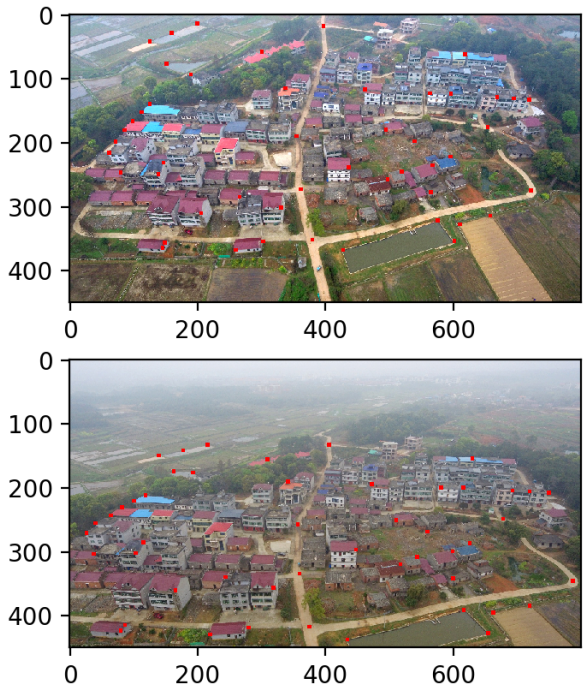


Question2

Algorithm Description

I use Thin plate spine (TPS) algorithm to fit two images.

- First, use `cpselect` in matlab, select 49 pairs of control points(Selected points are shown in following figure).



- Save `MovingPoints` and `FixedPoints` into a txt file and load it into python.
- Calculate radio basis function for each pair of control points:

$$r_{ij} = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}$$

$$\varphi(r_{ij}) = r_{ij}^2 log(r_{ij})$$

- Construct the following linear system:

$$\begin{bmatrix} \varphi(r_{11}) & \varphi(r_{12}) & \dots\dots & \varphi(r_{1n}) & x_1 & y_1 & 1 \\ & & \dots\dots & & & & \\ & & \dots\dots & & & & \\ \varphi(r_{11}) & \varphi(r_{12}) & \dots\dots & \varphi(r_{1n}) & x_n & y_n & 1 \\ x_1 & \dots\dots & & x_n & 0 & 0 & 0 \\ y_1 & \dots\dots & & y_n & 0 & 0 & 0 \\ 1 & \dots\dots & & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{11} \\ \omega_{12} & \omega_{12} \\ \vdots & \vdots \\ \omega_{1n} & \omega_{2n} \\ a_{11} & a_{21} \\ a_{12} & a_{22} \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} x'_1 & y'_1 \\ x'_1 & y'_2 \\ \vdots & \vdots \\ x'_n & y'_n \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- Use `np.linalg.solve` to solve this linear system.

- For each pixel in src image, transform it by following equation:

$$x^{'} = \sum_{i} \omega_{1i} \varphi(r_i) + a_{11}x + a_{12}y + b_1$$

$$y^{'} = \sum_{i} \omega_{2i} \varphi(r_i) + a_{21}x + a_{22}y + b_2$$

- If `x'` and `y'` are in valid range, transform it to the dst image.
- I used nearest interploitation to reduce the meshgrid effect.

Result

Question3

Algorithm Description

- GLCM calculating:
  - Construct a 8*8 matrix GLCM
  - for each pixel in the image, update (0 degree for examplt):

```
GLCM[img[i,j], img[i, j+1]] += 1
```

- Features calculating:
  - Use a 7*7 sliding window traverse the image, update the center pixel by calculating features in the window
  - For each angle, calculate by the given formula
- Energy (for each 7*7 window):

$$Energy = \sum_i \sum_j P(i,j)^2$$

- Correlation

$$Corr = \sum_i \sum_j \frac{P(i,j)(i-\mu_i)(j-\mu_j)}{\sigma_i \sigma_j}$$

where

$$\mu_i = \sum_i \sum_j iP(i,j)$$
$$\mu_j = \sum_i \sum_j jP(i,j)$$
$$\sigma_i^2 = \sum_i \sum_j (i-\mu_i)^2 P(i,j)$$
$$\sigma_j^2 = \sum_i \sum_j (j-\mu_j)^2 P(i,j)$$

- Homogeneity

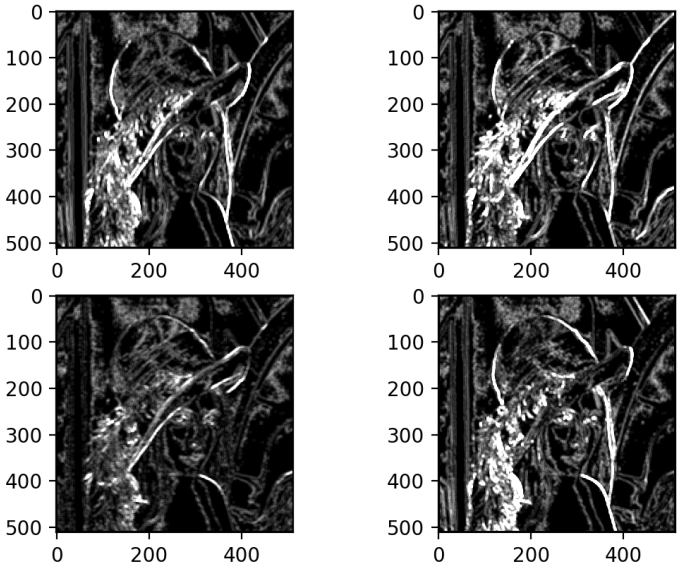
$$Homo = \sum_i \sum_j \frac{P(i,j)}{1+\|i-j\|)}$$

- Contrast

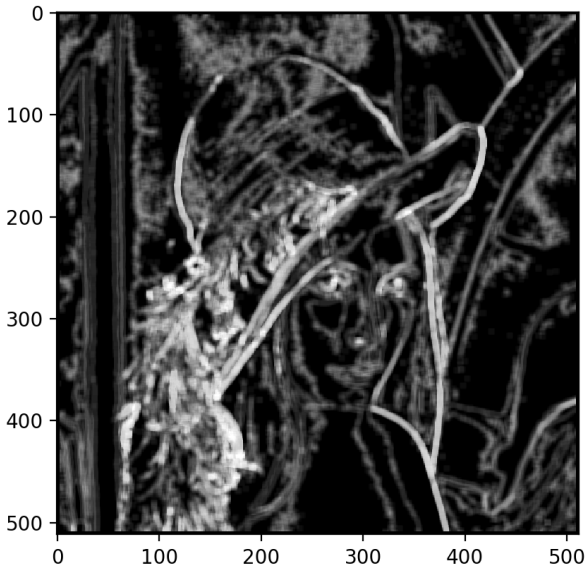
$$Contrast = \sum_i \sum_j P(i,j)(i-j)^2$$

Results

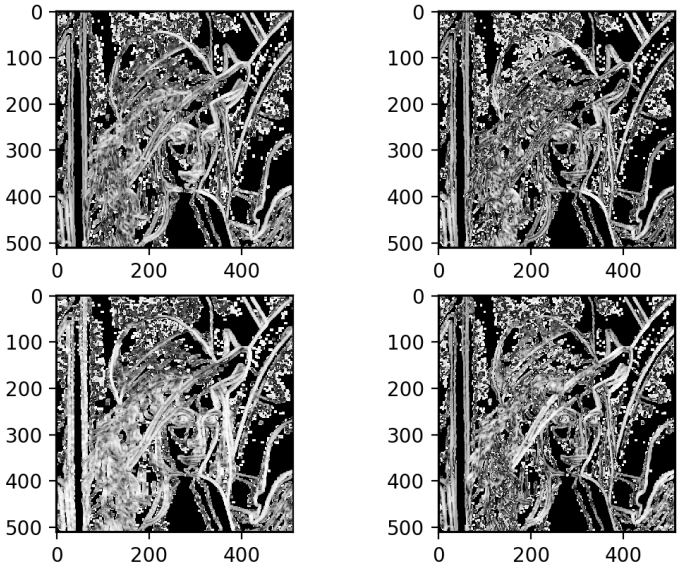
- Contrast - Different Angles



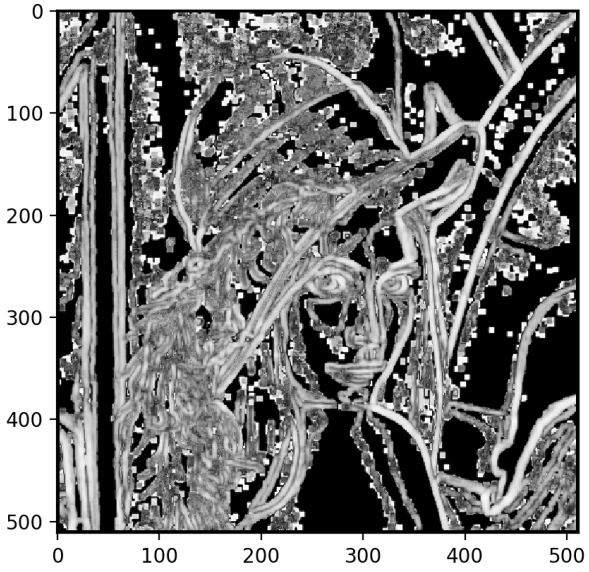
- Contrast - Average



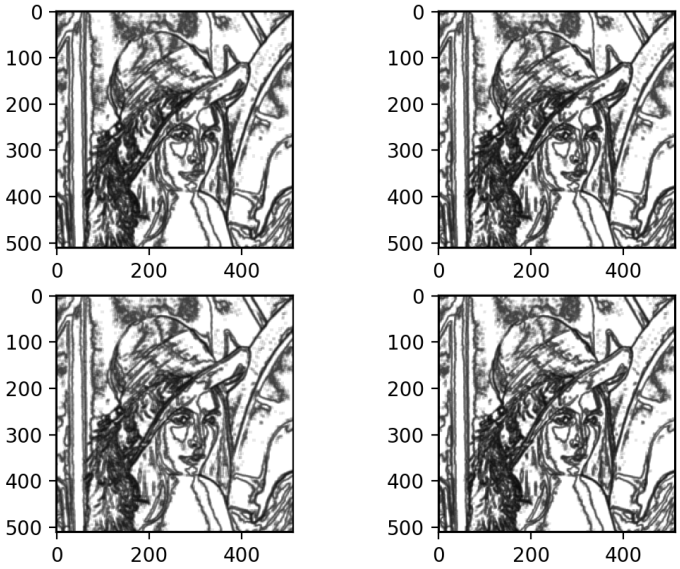
- Correlation - Different Angles



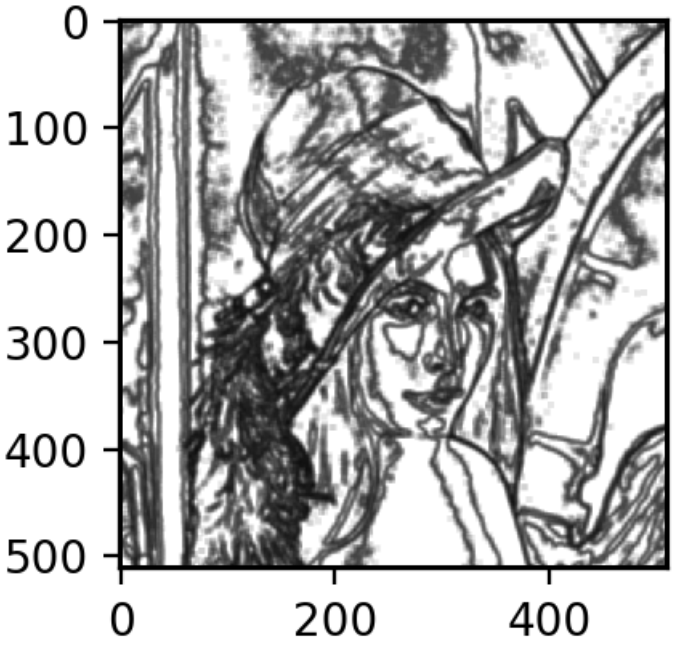
- Correlation - Average



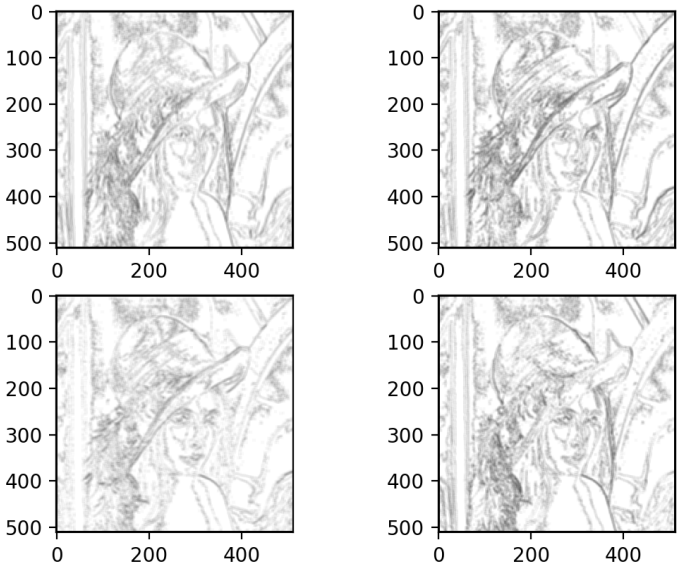
- Energy - Different Angles



- Energy - Average



- Homogeneity - Different Angles



- Homogeneity - Average