Digital Image Processing

HW#2

TA: Chih-En Yeh 葉志恩

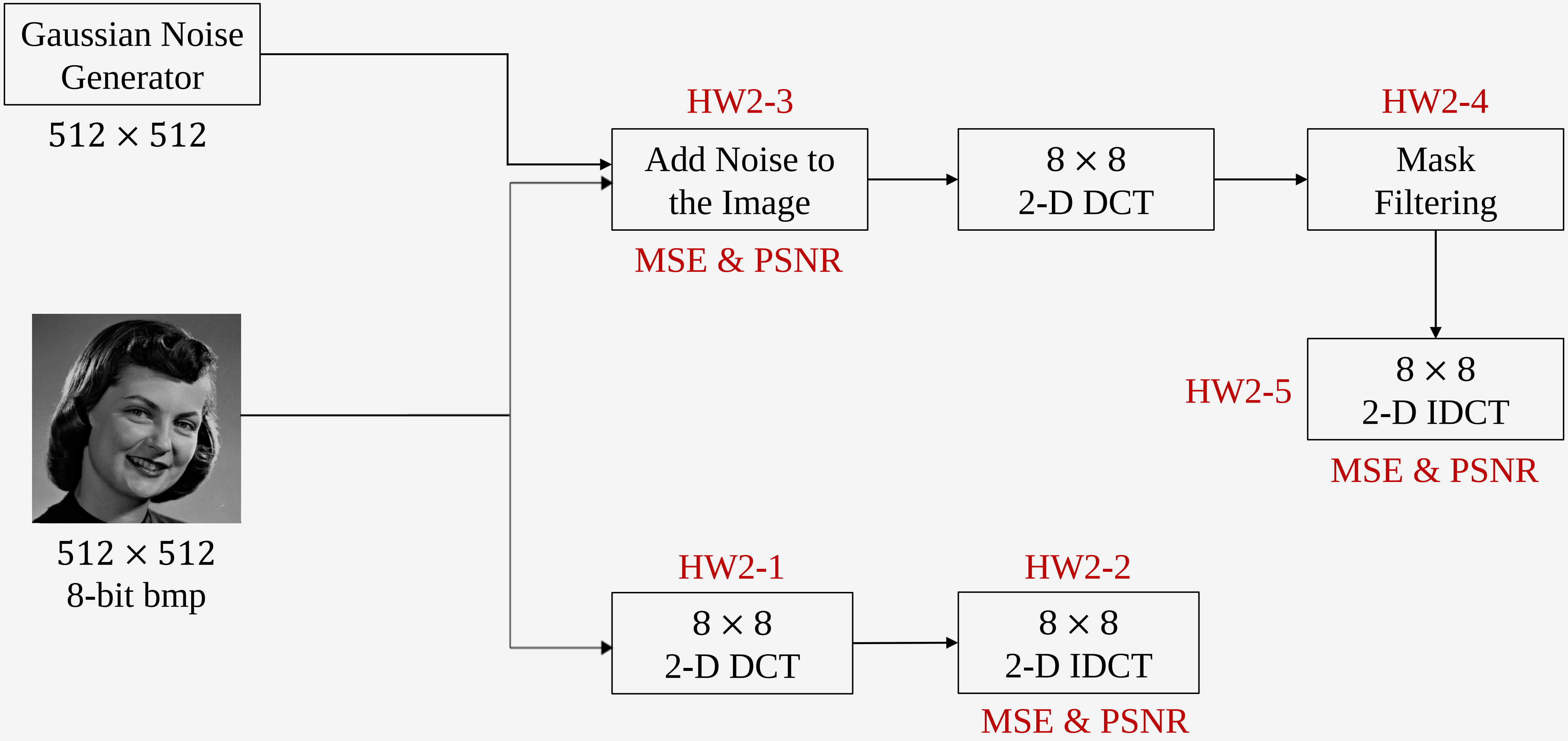
Instructor: Prof. Chih-Wei Tang

Visual Communications Lab

Dep. of Communications Engineering,
National Central University

2 November, 2020

Block Diagram



2-D Discrete Cosine Transform

- DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers:
  - The 8×8 DCT:

$$\mathrm{F}(u,v) = \frac{1}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}\left[C(u)C(v)f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)\right]$$

  - The 8×8 Inverse DCT:

$$f(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7}\left[C(u)C(v)F(u,v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)\right]$$

$where$

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \ u,v = 0 \\ 1 & otherwise \end{cases}$$

$f(x,y)$: spatial domain image.
$F(u,v)$: frequency domain image.
(x,y): index of spatial domain.
(u,v): index of frequency domain.

Bretl.com. *Discrete Cosine Transform*. [online] Available at: <http://www.bretl.com/mpeghtml/DCxfrm.HTM> [Accessed 14 October 2020].

2-D Discrete Cosine Transform

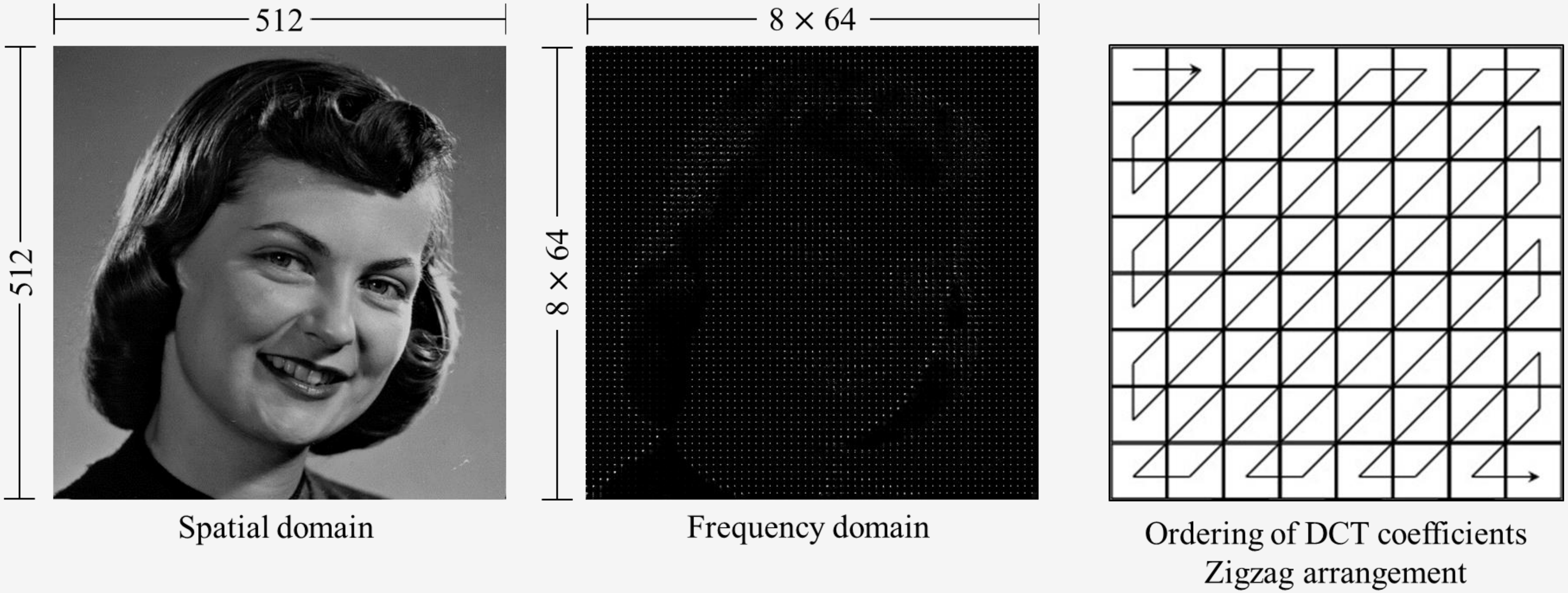


Figure: co-efficients, Z., Mendharkar, R., E2, M. and discontentment, J., 2020. *Zigzag Ordered Encoding Of DCT Co-Efficients*. [online] Mathematica Stack Exchange. Available at: <https://mathematica.stackexchange.com/questions/109869/zigzag-ordered-encoding-of-dct-co-efficients> [Accessed 13 October 2020].

Generating Gaussian Noise

- The general formula for the probability density function of the Gaussian distribution is:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- Generate a 512×512 matrix that each element of the matrix follows:

$$X \sim N(\mu, \sigma^2) = N(0, 16^2)$$

$$where$$

μ: mean

σ: standard deviation

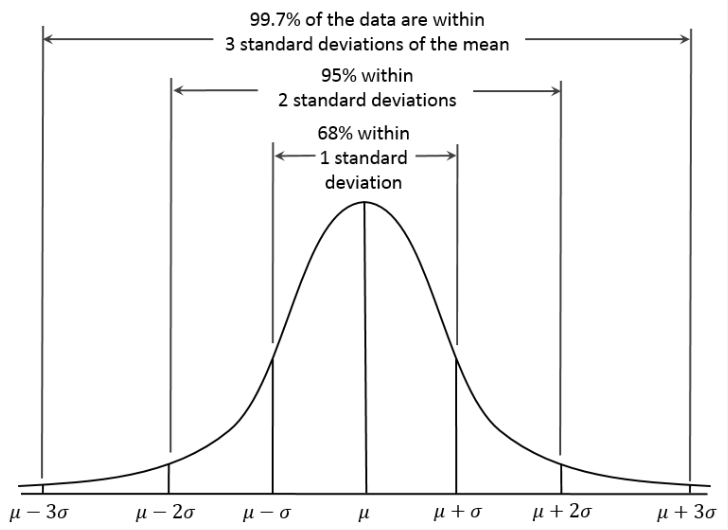
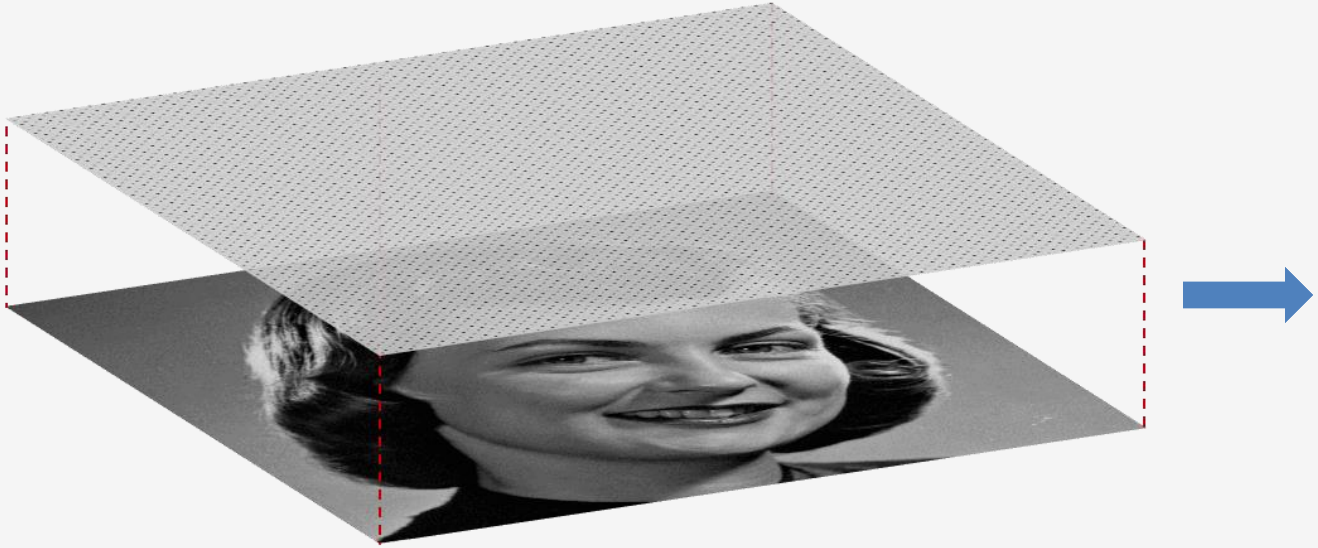


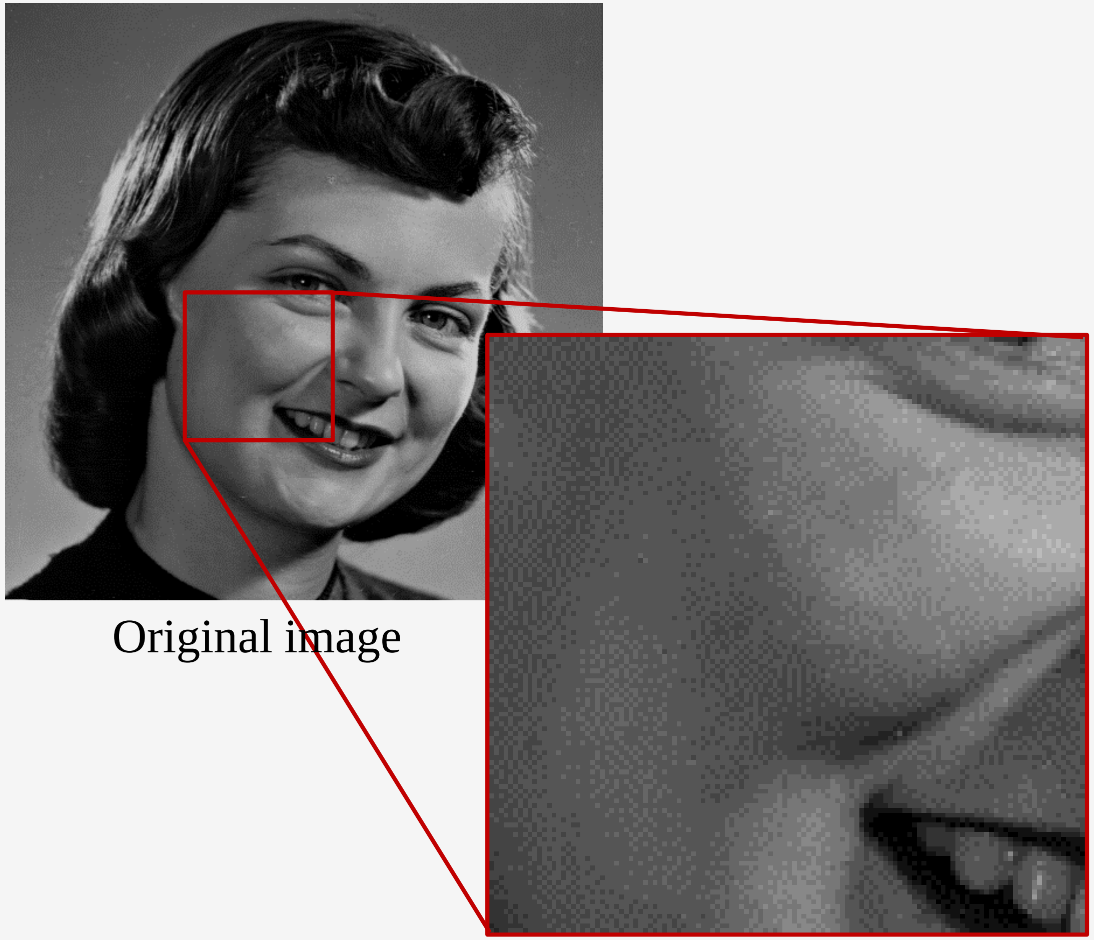
Figure: En.wikipedia.org. 2020. *68–95–99.7 Rule*. [online] Available at: https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule [Accessed 13 October 2020].

Adding Noise to Image

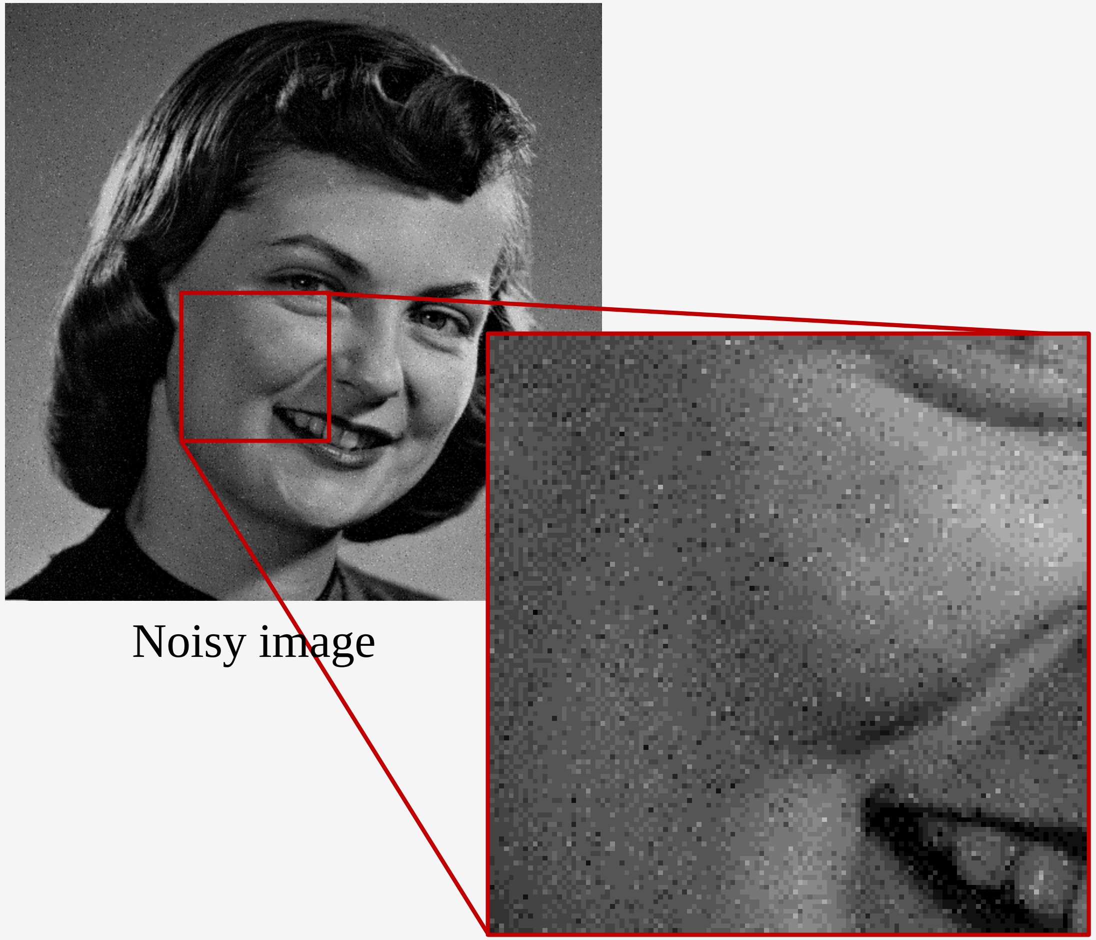
- Generate a noisy image by adding the Gaussian noise matrix to the original image pixel by pixel:
  - Each pixel on the original image has a probability of 25% to be affected by noise.
  - After adding noise, **set grayscale values over 255 to 255, and to 0 for those are less than 0**.



Adding Noise to Image



Original image



Noisy image

Mask Filtering

- In frequency domain, filtering is implemented by multiplying the image's spectra, $F(u, v)$, with the filter spectra $H(u, v)$:

$$F(u,v) \times H(u,v)$$

- A 8×8 mask is considered to remove noise in the image (Choose the filter coefficients by yourself!):
- **Objective:** Achieve higher PSNR of the denoised image than the noisy image.

```
mask=[ 1  1  1  0  0  0  0  0
       1  1  0  0  0  0  0  0
       1  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0]
```

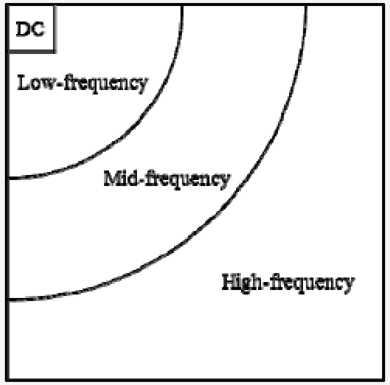


Figure: Jeon, Jaehwan, Jinhee Lee, and Joonki Paik. "Robust focus measure for unsupervised auto-focusing based on optimum discrete cosine transform coefficients." *IEEE Transactions on Consumer Electronics* 57.1 (2011): 1-5.

Mask Filtering

Original image

Noisy image

Denoised image
(after IDCT)

Image Quality Metrics

- MSE (Mean Squared Error):

$$MSE = \frac{1}{Image\ Size} \sum_{i=1}^{Image\ Size} \left(Y_i - \hat{Y}_i\right)^2$$

- PSNR (Peak Signal-to-Noise Ratio):

$$PSNR = 10 \times \log\left(\frac{255^2}{MSE}\right)$$

$where$

Y_i: The i-th pixel value of the original image

$\hat{Y}_i$: The i-th pixel value of the image processed by IDCT

$Image\ Size$: Image length × Image width

Grading

- **Code & Demo (70%): Use the C/C++ only. Matlab or OpenCV is not allowed**
  - 2-D DCT (15%) (HW2-1)
  - 2-D IDCT & MSE, PSNR measuring (10% + 5%) (HW2-2)
  - Generating Gaussian-noisy image & MSE, PSNR measuring (10% + 5%) (HW2-3)
  - Mask filtering (15%) (HW2-4)
  - 2-D IDCT & MSE, PSNR measuring (5% + 5%) (HW2-5)
- **Report (30%):**
  - Flowchart (10%)
  - Experiment results (10%)
  - Discussions (10%)

Due Date & Demo Schedule

- **Demo Date**: Nov. 23 (Monday) or Nov. 24 (Tuesday)
- **Demo Time & Location**: 13:30 ~ 17:30 @E1-214-1
- The demo schedule will be announced at the TA webpage.
- You should compress your entire project (including .c/.cpp, .exe file, etc.) and report (.pdf) as a .zip file and submit to LMS before Nov. 23, 13:00.
- No delay. (If you have any special case, please inform us by sending an email early.)

Note

- Do it yourself!
- You will get a zero when you delay or fail to operation in demo (code and demo part), but you can still get points in report part.
- Everyone will be asked a few questions and operations when you are in demo. (Do not call for help.)
- The TA will use another image to test your code.
- If you have a notebook, please bring your own notebook. Otherwise, some people may not be able to execute the code during the demo.
- Remote connection/control is not allowed.

The details will be announced on our course website:
http://140.115.154.29/html/course/DIP2020.html

References

- Gonzalez, Rafael C., and Richard E. Woods, “Digital image processing,“ Prentice Hall, 2007.
- Yu, Guoshen, and Guillermo Sapiro. "DCT image denoising: a simple and effective image denoising algorithm." *Image Processing On Line 1* (2011): 292-296.
- Test image “girlface.bmp” download: http://www.hlevkin.com/hlevkin/TestImages/girlface.bmp