

-۲

الف) با توجه به این که بردار های پایه بر هم عمود هستند، برای به دست آوردن ضریب یک بردار پایه کافیست کل عبارت را در آن بردار پایه ضرب داخلی کنیم؛ به این ترتیب ضرایب سایر بردار های پایه صفر می شوند و با تقسیم عبارت بر توان دو اندازه بردار پایه مد نظر، ضریب آن به دست می آید.

$$\alpha_i = \frac{\langle w_i, z \rangle}{\|w_i\|^2}$$

$$A = \frac{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ -20 & -5 \end{bmatrix}}{\left\| \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \right\|^2} = \frac{10 + 20 - 20 - 5}{4} = \frac{5}{4}$$

$$B = \frac{\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ -20 & -5 \end{bmatrix}}{\left\| \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} \right\|^2} = \frac{10 - 20 - 20 + 5}{4} = \frac{-25}{4}$$

$$C = \frac{\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ -20 & -5 \end{bmatrix}}{\left\| \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \right\|^2} = \frac{10 + 20 + 20 + 5}{4} = \frac{55}{4}$$

$$D = \frac{\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 10 & 20 \\ -20 & -5 \end{bmatrix}}{\left\| \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right\|^2} = \frac{10 - 20 + 20 - 5}{4} = \frac{5}{4}$$

(ب)

T: تابع تبدیل فوریه

U, V: مختصات بردار های پایه تبدیل فوریه، در واقع هر جفت U و V نشان دهنده یکی از بردار های پایه است.

S: بردار های پایه تبدیل فوریه معکوس

f: تابع تصویر

x, y: مختصات تصویر

(c)

$f(x, y)$:

1	1	1	1
1	1	1	1

$T(u, v)$:

$T(0,0)$	$T(0,1)$	$T(0,2)$	$T(0,3)$
$T(1,0)$	$T(1,1)$	$T(1,2)$	$T(1,3)$

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v)$$

$$r(x, y, u, v) = e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

$$T(0,0) = \sum_{x=0}^1 \sum_{y=0}^3 f(x, y) = 12$$

$$T(0,1) = \sum_{x=0}^1 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{y}{4})} = 0$$

$$T(0,2) = \sum_{x=0}^1 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{2y}{4})} = 0$$

$$T(0,3) = \sum_{x=0}^1 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{3y}{4})} = 0$$

$$T(1,0) = \sum_{x=0}^1 \sum_{y=0}^3 f(x, y) e^{-j2\pi(\frac{x}{2})} = 0$$

.....

12	0	0	0
0	$2-2j$	0	$2+2j$

توابع پایه:

$$r(x, y, u, v) = e^{-j2\pi \left(\frac{ux}{M} + \frac{vy}{N} \right)}$$

$$M = 2, N = 4$$

$$r(x, y, 0, 0) = e^0 = 1$$

1	1	1	1
1	1	1	1

$$r(x, y, 0, 1) = e^{-j2\pi \left(\frac{y}{4} \right)}$$

1	-j	-1	j
1	-j	-1	j

$$r(x, y, 1, 0) = e^{-j2\pi \left(\frac{x}{2} \right)}$$

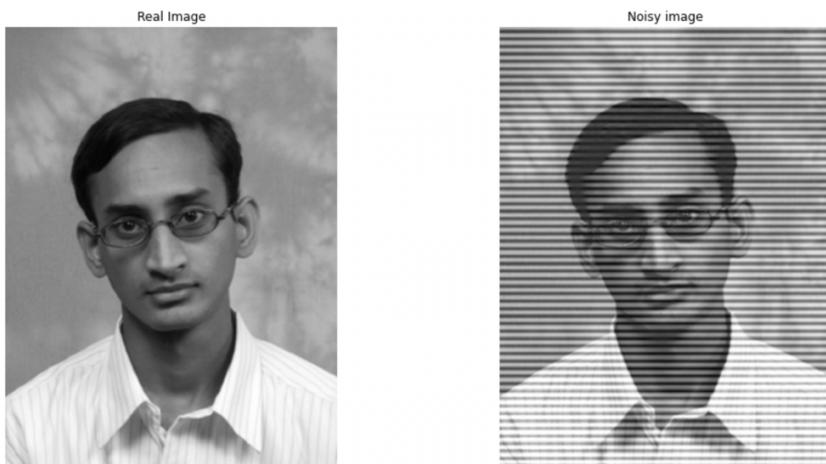
1	1	1	1
-1	-1	-1	-1

Remove Noise by DFT

تصاویر را خوانده و نمایش می‌دهیم

```
## read and take a look to images here ##
fig = plt.figure(figsize= (16, 8))
cols = 2
rows = 1
fig.add_subplot(rows , cols , 1)
real = cv2.imread('NoNoise.png', cv2.IMREAD_GRAYSCALE)
plt.imshow(real, cmap='gray')
plt.title('Real Image')
plt.axis('off')

fig.add_subplot(rows , cols , 2)
image = cv2.imread('NoisyImg.png', cv2.IMREAD_GRAYSCALE)
plt.imshow(image, cmap='gray')
plt.title('Noisy image')
plt.axis('off')
```



```
[ ] # define functions #
def dft(image):
    return np.fft.fft2(image)

def shift_fft(dft):
    return np.fft.fftshift(dft)

def reverse_shift(shifted_image):
    return np.fft.ifftshift(shifted_image)

def idft(image):
    return np.fft.ifft2(image)
```

با استفاده از کتابخانه numpy و متد های مربوط به تبدیل فوریه دو بعدی، توابع بالا را پیاده سازی می کنیم.

```

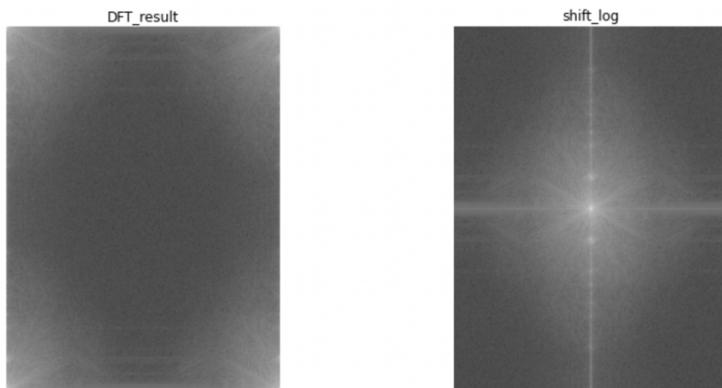
dft_img = dft(image)
dft_log = 20*np.log(np.abs(dft_img))

shift_img = shift_fft(dft_img)
shift_log = 20*np.log(np.abs(shift_img))

```

dft تصویر را محاسبه می‌کنیم. برای نمایش بهتر dft تصویر، ابتدا قدر مطلق و سپس لگاریتم آن را می‌گیریم و در یک عدد ضرب می‌کنیم تا بهتر نمایش داده شود.

سپس آن را شیفت می‌دهیم تا فرکانس های پایین در وسط تصویر قرار گیرند.



با نگاه به تصویر متوجه می‌شویم که مولفه های نویز عمودی هستند و با نگاه به تبدیل فوریه شیفت یافته نیز می‌توانیم خط عمودی که ناشی از این نویز هست را مشاهده کنیم. بنابراین ماسکی در نظر می‌گیریم که این خط عمودی را حذف کند.

filter mask



```

# enhance image #
rows, cols = image.shape
crow, ccol = int(rows / 2), int(cols / 2)

mask = np.ones((rows, cols), np.uint8)
t = 20
r = 40
mask[:crow-r,ccol-t:ccol+t]=0
mask[crow+r:,ccol-t:ccol+t]=0

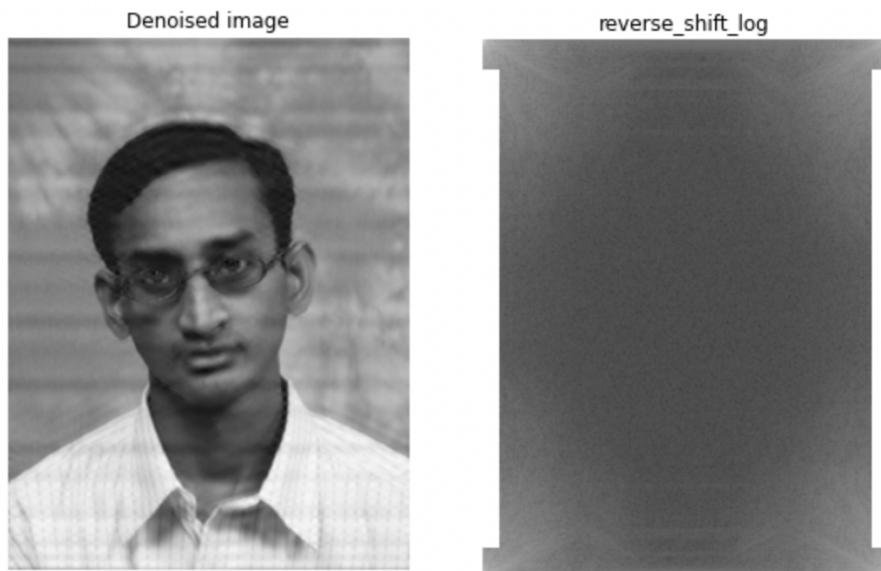
```

برای ساخت این ماسک از کد بالا استفاده می‌کنیم.

ماسک را در تبدیل فوریه ضرب می‌کنیم، شیفت معکوس اعمال می‌کنیم، و در آخر با گرفتن تبدیل فوریه معکوس به تصویر بهود یافته می‌رسیم.

```
masked_fft = shift_img * mask
reverse_shifte_img = reverse_shift(masked_fft)
reverse_shifte_log = 20*np.log(np.abs(reverse_shifte_img))

img_back = idft(reverse_shifte_img)
img_back = img_back.real
```



چاپ مقادیر ارزیابی:

```
# evaluate by criteries #
mse = skimage.metrics.mean_squared_error(img_back,real)
psnr = skimage.metrics.peak_signal_noise_ratio(real,img_back)
ssim = skimage.metrics.structural_similarity(img_back,real)

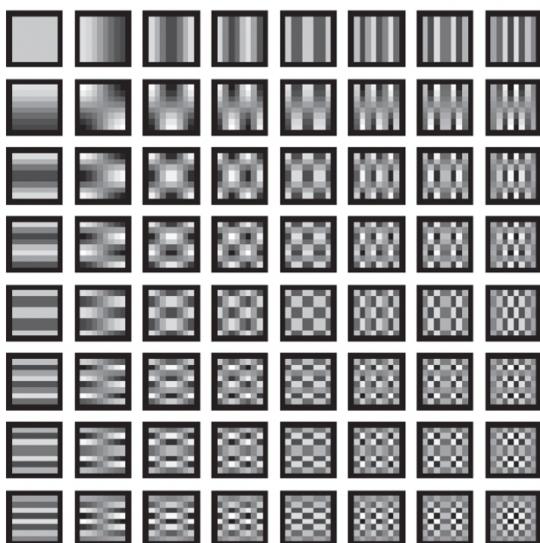
mse=146.9158876627812
psnr=26.46011597404545
ssim=0.6650605941002975
```

Remove Noise by DCT

توابع تبدیل dct و معکوس آن:

```
# define functions #
def dct_transform(image):
    return dct(image)

def idct_transform(image):
    return idct(image)
```



در تبدیل dct چپ ترین مولفه ها مربوط به نویز های عمودی هستند. نویز تصویر مانیز از همین جنس است؛ پس برای حذف نویز چپ ترین مولفه ها را حذف می کنیم.

```
# enhance image #
rows, cols = image.shape

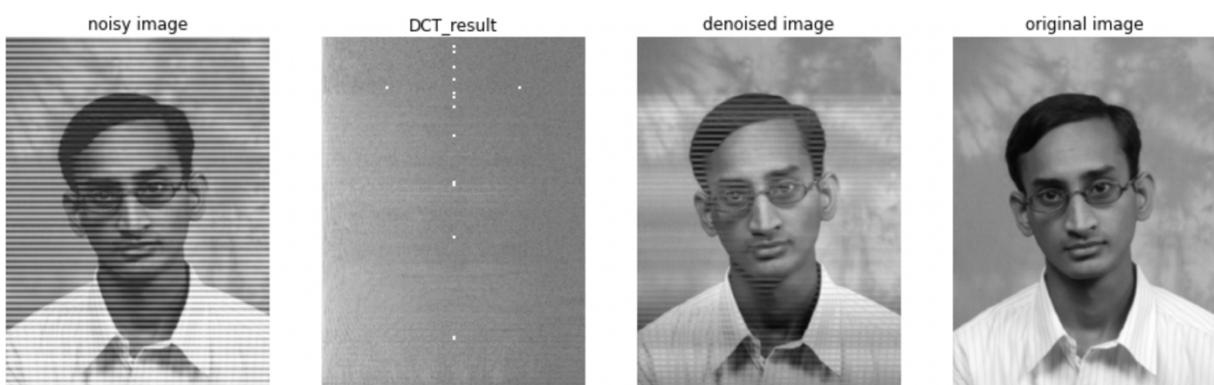
mask = np.ones((rows, cols), np.uint8)
mask[:, :1]=0

img_DCT = dct_transform(image)
DCT_log = np.log(np.abs(img_DCT))

masked_DCT = img_DCT * mask
masked_DCT_log = np.log(np.abs(masked_DCT))

enhanced_image = idct_transform(masked_DCT)
```

برای این منظور ماسکی می سازیم که خط سمت چپ آن به اندازه یک پیکسل صفر است و آن را در تصویر ضرب می کنیم سپس تبدیل معکوس می گیریم.



```
# evaluate by criteries #
mse = skimage.metrics.mean_squared_error(enhanced_image,real)
psnr = skimage.metrics.peak_signal_noise_ratio(real,enhanced_image)
ssim = skimage.metrics.structural_similarity(enhanced_image,real)
```

```
mse=1312918244.1128201
psnr=-43.05157322391309
ssim=6.5485960302937795e-06
```

(c)

MSE

In [statistics](#), the **mean squared error (MSE)**^[1] or **mean squared deviation (MSD)** of an [estimator](#) (of a procedure for estimating an unobserved quantity) measures the [average](#) of the squares of the [errors](#)—that is, the average squared difference between the estimated values and the actual value. MSE is a [risk function](#), corresponding to the [expected value](#) of the [squared error loss](#).^[2] The fact that MSE is almost always strictly positive (and not zero) is because of [randomness](#) or because the estimator [does not account for information](#) that could produce a more accurate estimate.^[3] In [machine learning](#), specifically [empirical risk minimization](#), MSE may refer to the *empirical* risk (the average loss on an observed data set), as an estimate of the true MSE (the true risk: the average loss on the actual population distribution).

The MSE is a measure of the quality of an estimator. As it is derived from the square of [Euclidean distance](#), it is always a positive value that decreases as the error approaches zero.

PSNR

Peak signal-to-noise ratio (PSNR) is an engineering term for the ratio between the maximum possible power of a [signal](#) and the power of corrupting [noise](#) that affects the fidelity of its representation. Because many signals have a very wide [dynamic range](#), PSNR is usually expressed as a [logarithmic](#) quantity using the [decibel](#) scale.

PSNR is commonly used to quantify reconstruction quality for images and video subject to [lossy compression](#).

SSIM

The **structural similarity index measure (SSIM)** is a method for predicting the perceived quality of digital television and cinematic pictures, as well as other kinds of digital images and videos. SSIM is used for measuring the similarity between two images. The SSIM index is a [full reference metric](#); in other words, the measurement or prediction of [image quality](#) is based on an initial uncompressed or distortion-free image as reference.

SSIM is a perception-based model that considers image degradation as *perceived change in structural information*, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. The difference with other techniques such as [MSE](#) or [PSNR](#) is that these approaches estimate *absolute errors*. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or "texture" in the image.

References:

https://en.wikipedia.org/wiki/Mean_squared_error

https://en.wikipedia.org/wiki/Structural_similarity

https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio