画像工学特論課題 3

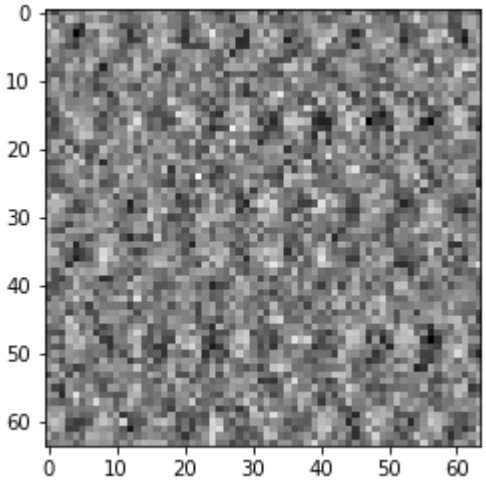
二次元フーリエ変換

エネルギー環境システム専攻修士課程1年 26213167
和田拓弥

与えられた画像について可視化したものを以下に示す.

In [1]:
```
# import library
import cv2
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
```

In [2]:
```
# read img
img = cv2.imread("sample.png", 0)

# show img
fig = plt.imshow(img, cmap='gray')
plt.show()
```



この画像について以下に示す二次元離散フーリエ変換(以下DDFT)を試みる. 画像から $N_x = 64, N_y = 64$ である.

$$F_{m_x,m_y} = \frac{1}{N_x N_y} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} f_{n_x,n_y} \exp\{-i(\frac{2\pi m_x n_x}{N_x} + \frac{2\pi m_y n_y}{N_y})\}$$

ここで上式はループ構造として4重ループ $(= {N_x}^2 {N_y}^2)$ の構成となっており, これでは実行時の処理負荷が大きくなる. これを解消するべく, 以下の式で示す G_{n_x,m_y} を事前に計算することで3重ループ $(= {N_x}^2 N_y + N_x {N_y}^2)$ に工夫した.

$$G_{n_x,m_y} = \frac{1}{N_y} \sum_{n_y=0}^{N_y-1} f_{n_x,n_y} \exp(-i\frac{2\pi m_y n_y}{N_y})$$

$$F_{m_x,m_y} = \frac{1}{N_x}\sum_{n_x=0}^{N_x-1} G_{n_x,m_y}\exp(-i\frac{2\pi m_x n_x}{N_x})$$

さらに回転子の計算についても計算過程を工夫した．F_{m_x,m_y}, G_{n_x,m_y} の実部と虚部は別々に計算し，$\exp(-i\frac{2\pi nm}{N})$ は nm を N で割った $N(=N_x, N_y)$ 通りの余りを格納した配列，合計4配列について，予め計算することより処理負荷を低減した．

In [3]:
```
# define DDFT function
def ddft(data, data2=None, mode="dft"):
    Ny, Nx = data.shape

    ReFmxmy = np.zeros_like(data).astype(float)
    ImFmxmy = np.zeros_like(data).astype(float)
    ReGnxmy = np.zeros_like(data).astype(float)
    ImGnxmy = np.zeros_like(data).astype(float)

    ReWpNx = [np.cos(2 * np.pi * i / Nx) for i in range(Nx)]
    ImWpNx = [np.sin(2 * np.pi * i / Nx) for i in range(Nx)]
    ReWpNy = [np.cos(2 * np.pi * i / Ny) for i in range(Ny)]
    ImWpNy = [np.sin(2 * np.pi * i / Ny) for i in range(Ny)]

    if mode == "dft":
        for nx in range(Nx):
            for my in range(Ny):
                for ny in range(Ny):
                    ReGnxmy[my][nx] += data[ny][nx] * ReWpNy[(my * ny) % Ny]
                    ImGnxmy[my][nx] += -data[ny][nx] * ImWpNy[(my * ny) % Ny]

        for mx in range(Nx):
            for my in range(Ny):
                for nx in range(Nx):
                    ReFmxmy[my][mx] += ReGnxmy[my][nx] * ReWpNx[(mx * nx) % Nx] + ImGnxmy[my]
                    ImFmxmy[my][mx] += -ReGnxmy[my][nx] * ImWpNx[(mx * nx) % Nx] + ImGnxmy[my

        ReFmxmy /= Nx * Ny
        ImFmxmy /= Nx * Ny

    elif mode == "idft":
        for nx in range(Nx):
            for my in range(Ny):
                for ny in range(Ny):
                    ReGnxmy[my][nx] += data[ny][nx] * ReWpNy[(my * ny) % Ny] - data2[ny][nx] * ImW
                    ImGnxmy[my][nx] += data[ny][nx] * ImWpNy[(my * ny) % Ny] + data2[ny][nx] * ReW

        for mx in range(Nx):
            for my in range(Ny):
                for nx in range(Nx):
                    ReFmxmy[my][mx] += ReGnxmy[my][nx] * ReWpNx[(mx * nx) % Nx] - ImGnxmy[my]
                    ImFmxmy[my][mx] += ReGnxmy[my][nx] * ImWpNx[(mx * nx) % Nx] + ImGnxmy[my]

        ReFmxmy /= Nx * Ny
        ImFmxmy /= Nx * Ny

    return ReFmxmy, ImFmxmy
```

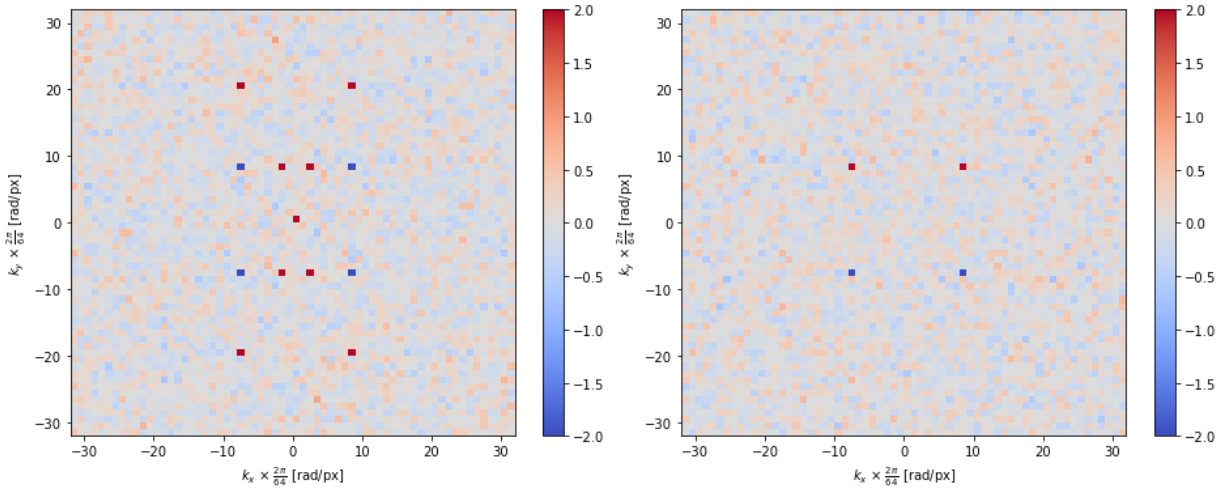
In [4]:
```
ReFm, ImFm = ddft(img, "dft")
```

In [5]:
```
ReFm2 = np.zeros_like(ReFm)
```

```
ImFm2 = np.zeros_like(ImFm)

ReFm2[0:32, 0:32], ReFm2[32:, 0:32], ReFm2[0:32, 32:], ReFm2[32:, 32:] = ReFm[32:, 32:], R
ImFm2[0:32, 0:32], ImFm2[32:, 0:32], ImFm2[0:32, 32:], ImFm2[32:, 32:] = ImFm[32:, 32:], Im
```

In [6]:

```
# show DDFT results
fig, axes = plt.subplots(nrows=1, ncols=2, sharex=False, figsize=(16.0, 6.0))
plt.subplots_adjust(wspace=0.1)

axes[0].set_xlabel(r"$k_x$ × $\frac{2\pi}{64}$ [rad/px]")
axes[0].set_ylabel(r"$k_y$ × $\frac{2\pi}{64}$ [rad/px]")
axes[1].set_xlabel(r"$k_x$ × $\frac{2\pi}{64}$ [rad/px]")
axes[1].set_ylabel(r"$k_y$ × $\frac{2\pi}{64}$ [rad/px]")

X, Y = np.mgrid[-32:33, -32:33]

ReFm_result = axes[0].pcolormesh(X, Y, ReFm2, cmap='coolwarm')
ImFm_result = axes[1].pcolormesh(X, Y, ImFm2, cmap='coolwarm')

ReFm_result.set_clim(vmin=-2, vmax=2)
ImFm_result.set_clim(vmin=-2, vmax=2)

cbar1 = fig.colorbar(ReFm_result, ax=axes[0])
cbar2 = fig.colorbar(ImFm_result, ax=axes[1])

plt.show()
```



左はDDFTの結果のうち実部を示したものであり，右は虚部を示したものである．また上記の波数区間は，DDFTを行った結果の配列に関して，第1象限と第3象限，第2象限と第4象限を入れ替えて表示している．

実部において，ピーク点は各象限に3つと原点に存在している．まず3つのピークについて，第1象限に注目すると主要な3つの波数ベクトルは以下のように求められる．

$$k_1 = \frac{2\pi}{64}(2, 8) = (\frac{\pi}{16}, \frac{\pi}{4})[\mathrm{rad/px}]$$

$$k_2 = \frac{2\pi}{64}(8, 8) = (\frac{\pi}{4}, \frac{\pi}{4})[\mathrm{rad/px}]$$

$$k_3 = \frac{2\pi}{64}(8, 20) = (\frac{\pi}{4}, \frac{5\pi}{8})[\mathrm{rad/px}]$$

原点のピークについて,

In [7]:
```
print(f"実部: {ReFm[0][0]}, 虚部: {ImFm[0][0]}")
```

実部: 128.421875, 虚部: 0.0

原点すなわち，$F_{0,0}$の値は定義式から，サンプル画像の各ピクセルの輝度値を表している.

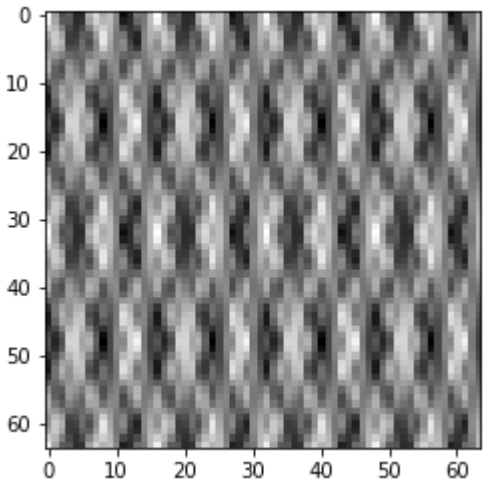
$$F_{0,0} = \frac{1}{N_x, N_y} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} f_{n_x,n_y}$$

次に，ヒントよりサンプル画像は「3つの成分の正弦波の和とランダムノイズから生成した」と記載されていることから，ランダムノイズを閾値処理により除去し，元画像の取得を試みる.
閾値処理について，DDFTの結果から1以下の値を0とするような処理に設定した.

In [8]:
```
th = 1

ReFm_th = np.where(ReFm < th, 0, ReFm)
ImFm_th = np.where(ImFm < th, 0, ImFm)
```

In [9]:
```
img_th, _ = ddft(ReFm_th, ImFm_th, "idft")
```

In [10]:
```
# show img
fig = plt.imshow(img_th, cmap='gray')
plt.show()
```



結果を見るとsample.pngより周期的な構造が得られたことが分かる.

さらに主要な波数ベクトルのみ抽出し，2次元の逆DFT(以下IDDFT)を行った．単純な比較のため，主要な波数と平均輝度値を表す原点のピーク以外の値は全て0に設定している.

In [11]:
```
ReFm_k1 = np.zeros_like(ReFm)
ReFm_k2 = np.zeros_like(ReFm)
ReFm_k3 = np.zeros_like(ReFm)

ImFm_k1 = np.zeros_like(ImFm)
ImFm_k2 = np.zeros_like(ImFm)
ImFm_k3 = np.zeros_like(ImFm)

ReFm_k1[0, 0], ReFm_k1[2, 8] = ReFm[0, 0], ReFm[2, 8]
```

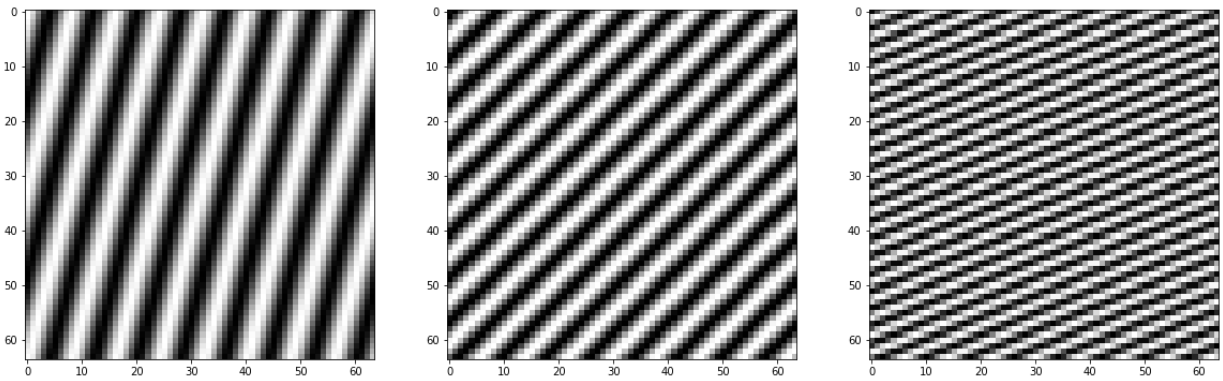
```
ReFm_k2[0, 0], ReFm_k2[8, 8] = ReFm[0, 0], ReFm[8, 8]
ReFm_k3[0, 0], ReFm_k3[20, 8] = ReFm[0, 0], ReFm[20, 8]

ImFm_k1[0, 0], ImFm_k1[2, 8] = ImFm[0, 0], ImFm[2, 8]
ImFm_k2[0, 0], ImFm_k2[8, 8] = ImFm[0, 0], ImFm[8, 8]
ImFm_k3[0, 0], ImFm_k3[20, 8] = ImFm[0, 0], ImFm[20, 8]
```

In [12]:

```
img_k1, _ = ddft(ReFm_k1, ImFm_k1, "idft")
img_k2, _ = ddft(ReFm_k2, ImFm_k2, "idft")
img_k3, _ = ddft(ReFm_k3, ImFm_k3, "idft")
```

In [13]:

```
# show img
fig, axes = plt.subplots(nrows=1, ncols=3, sharex=False, figsize=(20.0, 6.0))
plt.subplots_adjust(wspace=0.2)

axes[0].imshow(img_k1, cmap='gray')
axes[1].imshow(img_k2, cmap='gray')
axes[2].imshow(img_k3, cmap='gray')

plt.show()
```



上図はそれぞれの主要な波数ベクトルのみ抽出したIDDFTの結果であり，波数空間でのスペクトル値の大小によって，振幅が異なっている様子がわかる． 左は64 pxの間にx方向に8周期，y方向に2周期，中央はx, y方向に8周期，右はx方向に20周期，y方向に8周期の波を有することが図から読み取れ，これはDDFTの結果に一致する．

sample.pngは，上図に示すこれらの画像を重ね合わせたものに，ランダムノイズを追加したものであると考えられる．