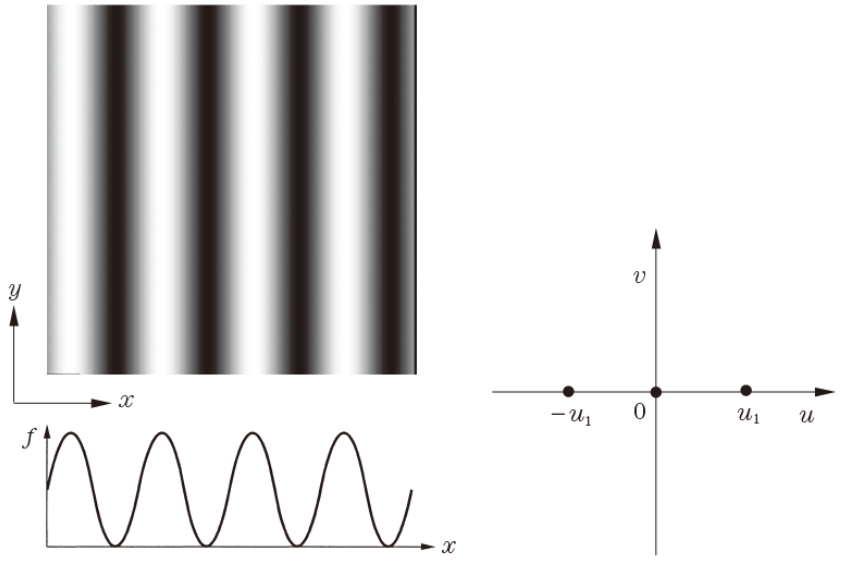
空間周波数

空間周波数

- 画像信号における周波数（**空間周波数**）は，
  単位長さ中に存在する濃淡で表される縞模様の数
- 空間周波数領域上のスペクトルで表現される



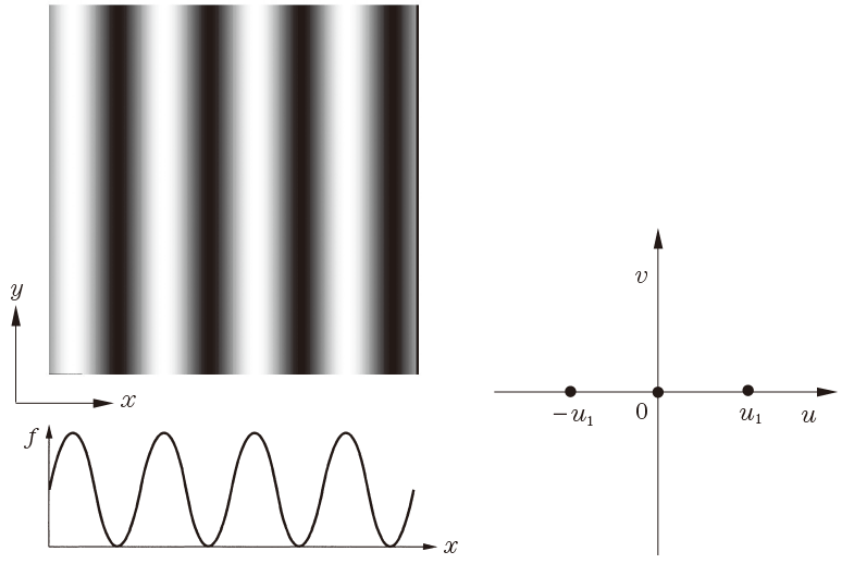
（a） $f(x,\ y) = A \sin u_1 x + A$ の画像　　（b） 空間周波数領域での表現

図 4.2　2 次元正弦波の例（x 軸方向のみに濃淡変化がある場合）

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門―」 コロナ社より

空間周波数

- x軸方向の濃度変化がある場合，u軸上にスペクトル
  - 直流成分：$(0,0)$
  - 交流成分：$(u_1,0),(-u_1,0)$

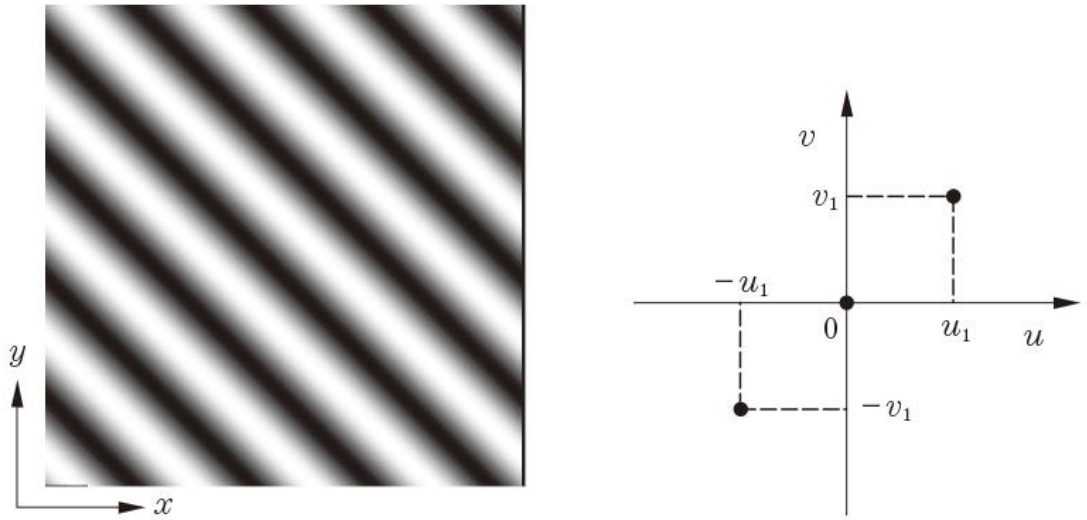


（a） $f(x,\ y)=A\sin u_1 x + A$ の画像　　（b） 空間周波数領域での表現

図 4.2 2次元正弦波の例（x 軸方向のみに濃淡変化がある場合）

空間周波数

- 斜め方向に濃淡変化がある場合
  - 直流成分：$(0,0)$
  - 交流成分：$(u_1, v_1), (-u_1, -v_1)$



（a） $f(x, y) = A \sin(u_1 x + v_1 y) + A$ の画像　　（b） 空間周波数領域での表現

図 4.3 2 次元正弦波の例（x 軸および y 軸方向に濃淡変化がある場合）

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門―」 コロナ社より

2次元DFTの基底

- $M = N = 8$における2次元DFTの基底
  （$W_M^{xk} W_n^{yl}$の実部を濃淡で表現）

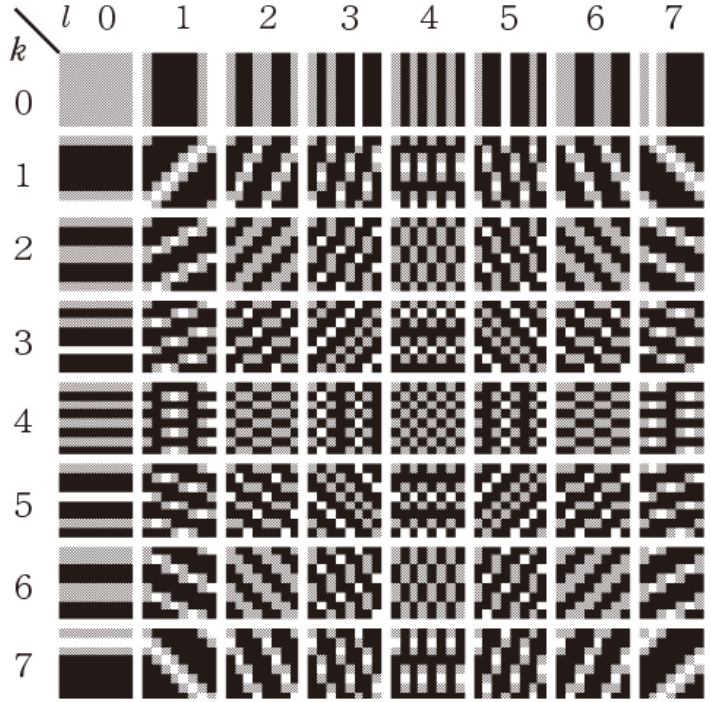
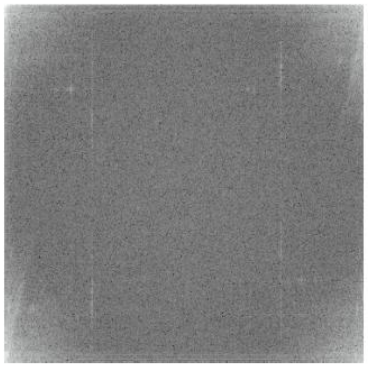


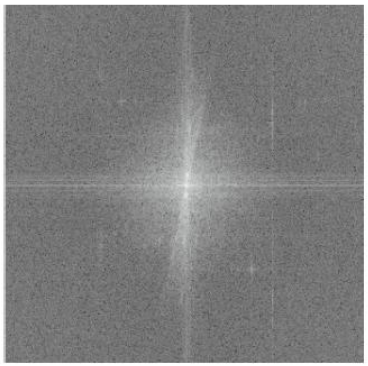
図 4.8　2 次元 DFT の基底 $M=N=8$ の場合）

画像の2次元DFT

（a） 原画像（512×512 画素）



（b） 2次元離散フーリエ変換の結果
（4隅が低周波成分）



（c） 2次元離散フーリエ変換の結果
（中央が低周波成分）

（d） 2次元離散フーリエ逆変換の結果

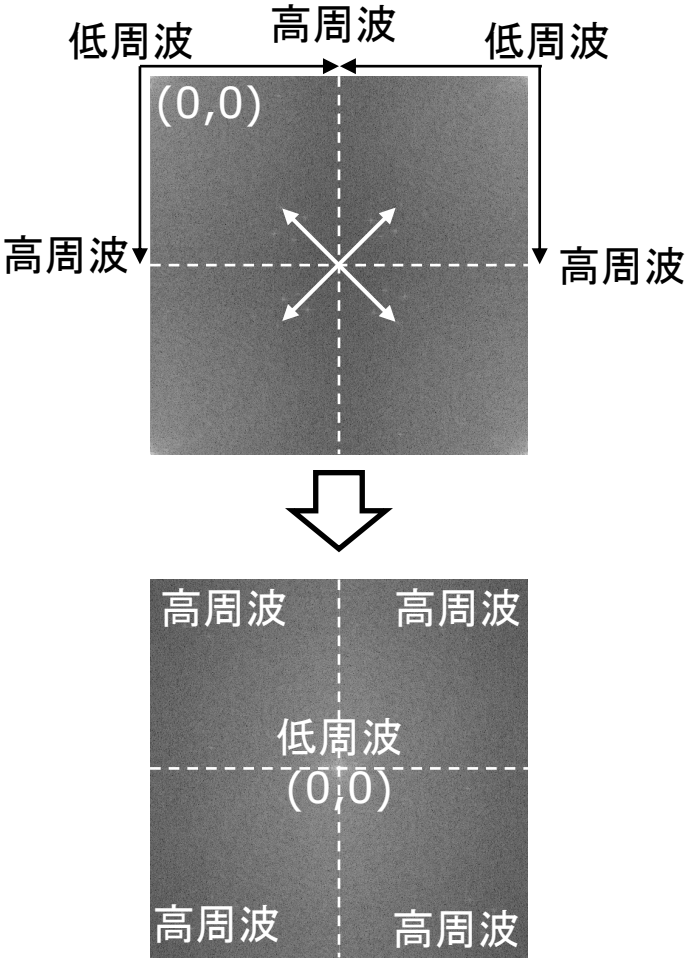


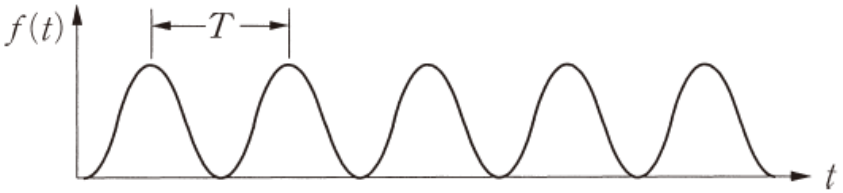
図 4.7 2 次元離散フーリエ変換

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門」 コロナ社より

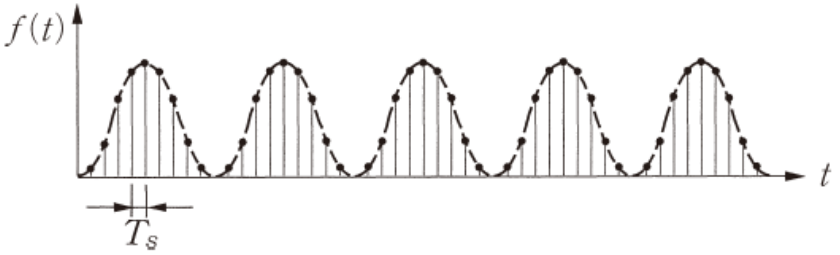
標本化定理

標本化定理

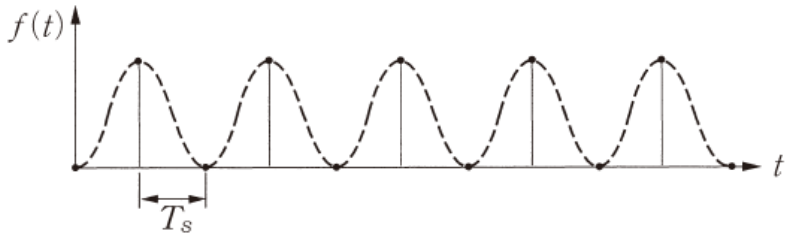
- 元の正弦波信号の周期Tの½より小さい間隔T_sで標本化すれば，元の信号を再現できる



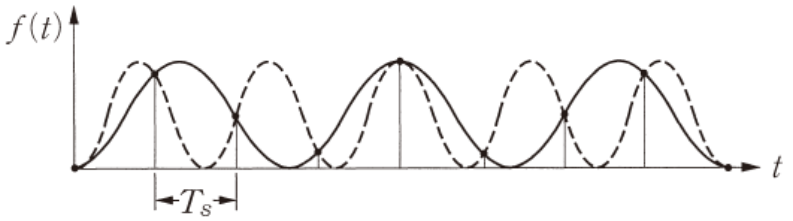
（a） 元の正弦波（アナログ信号）



（b） 標本化周期 $T_s = T/10$ とした場合



（c） 標本化周期 $T_s = T/2$ とした場合



（d） 標本化周期を $T_s = 5T/8$ とした場合

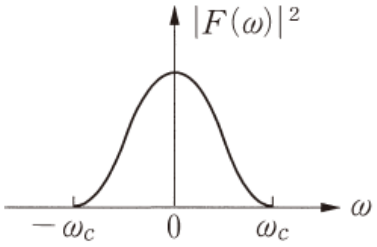
図 4.10 周期 T の 1 次元正弦波信号の標本化

標本化定理(周波数での表現)

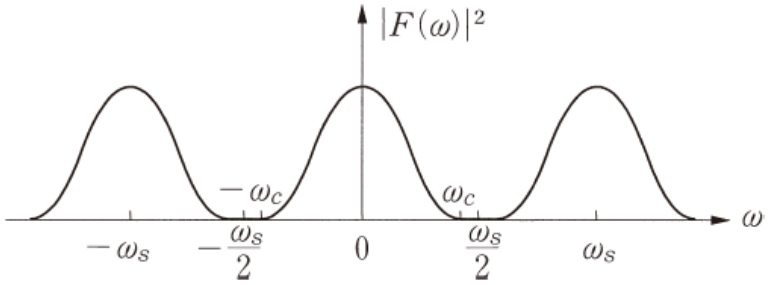
- 周波数で表現すると
  「周波数ω(＝1/T)の正弦波を離散化するためには,
  標本化周波数ω_sをω_s＞2ωとする必要がある」
- 別の見方をすれば,
  「標本化周波数ω_sで標本化するとき,
  ω_s /2を超える周波数成分は表現できない」
- ω_s /2:**ナイキスト周波数(遮断周波数)**と呼ばれる
- ナイキスト周波数より高い周波数を持つ正弦波を
  標本化した場合,**エイリアシング**が発生する

パワースペクトルと標本化定理

- 一般的な信号波形には，様々な周波数成分が含まれているため，「**最大周波数の2倍以上の周波数で標本化**」すればよい．
- ある信号の最大周波数がω_cであるとき，
  これを十分高い標本化周波数ω_sで標本化すると，
  ω_sを周期としてパワースペクトルが繰り返し出てくる
  分布となる．



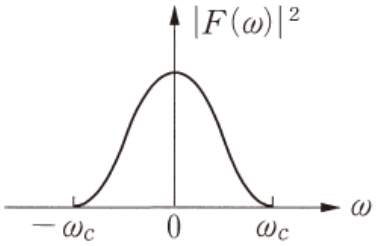
（a） ある１次元アナログ信号のパワースペクトル



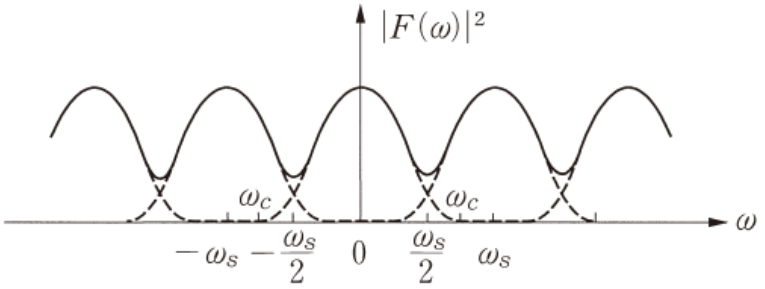
（b） $\omega_s(>2\omega_c)$で標本化された信号のパワースペクトル

パワースペクトルと標本化定理

- 標本化周波数を$\omega_s < 2\omega_c$とすると，パワースペクトルに重なりが生じてしまう
- このような**折り返し誤差（エイリアシング誤差）**が生じると，元の信号を復元することはできない



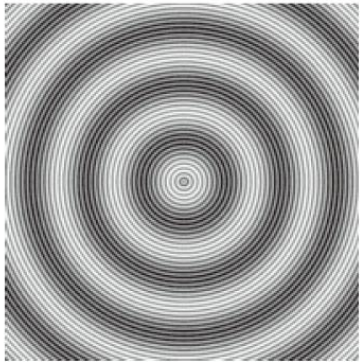
（a） ある１次元アナログ信号のパワースペクトル



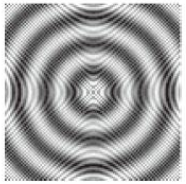
（c） $\omega_s(<2\omega_c)$で標本化された信号のパワースペクトル

図 4.11 周波数領域で見た標本化定理

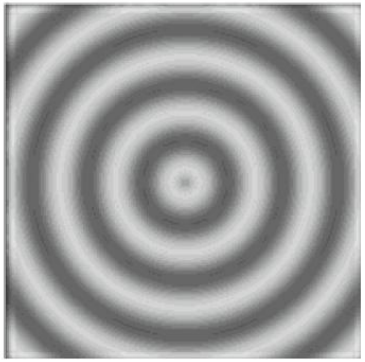
画像信号のエイリアシング



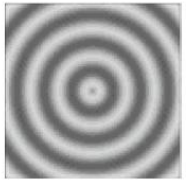
（a） 二つの周波数成分 ω_{c1}，ω_{c2}（$\omega_{c1}>\omega_{c2}$）が含まれた画像（解像度 200×200）



（b） 画素を一つおきに間引きして縮小した画像（解像度 100×100）



（c） ローパスフィルタにより，ω_{c1} の周波数成分を取り除いた画像（解像度 200×200）



（d） 図（c）に対して画素を間引きして縮小した画像（解像度 100×100）

図 4.12 画像のリサイズとエリアシング

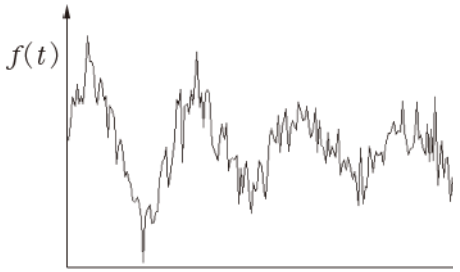
山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門―」 コロナ社より

周波数フィルタリング

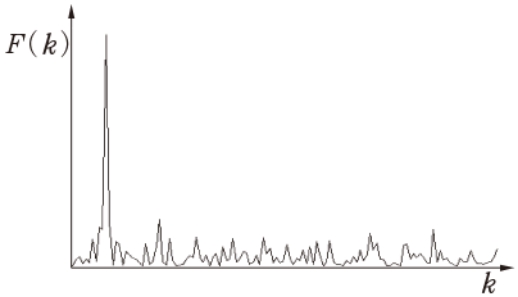
フィルタ処理

- フィルタリングの手順

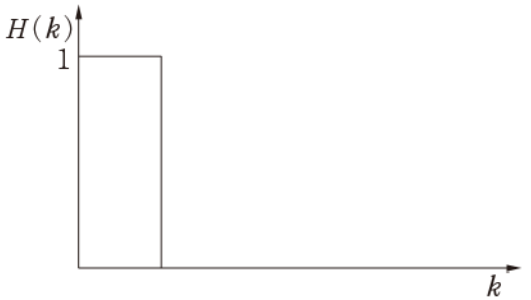
1. 信号をFFTしてスペクトル$F(k)$を得る
2. フィルタ関数$H(k)$を用意して信号のスペクトル$F(k)$と掛け合わせる
3. 演算後のスペクトル$F(k)H(k)$をIFFTする



（a） 信号$f(t)$



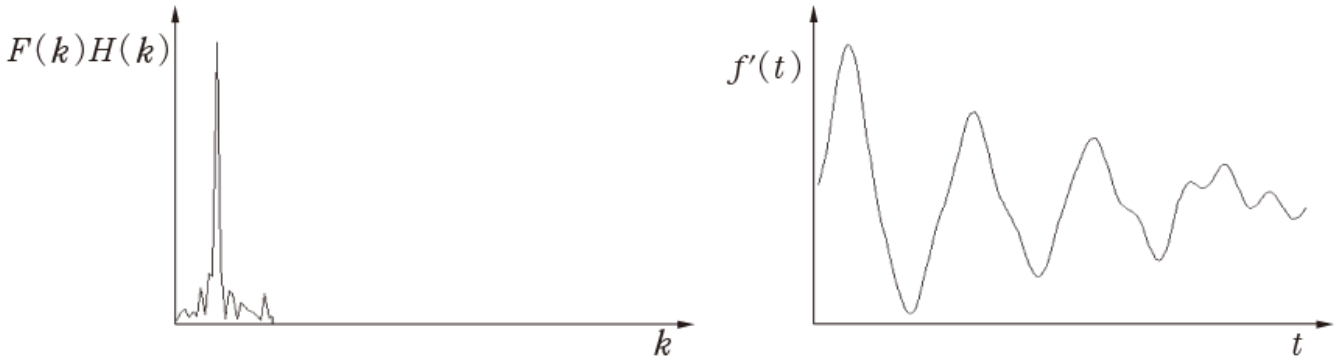
（b） 信号$f(t)$の振幅スペクトル$F(k)$



（c） フィルタ関数$H(k)$
（ローパスフィルタ）

フィルタ処理

- フィルタ関数$H(k)$の形状を変化させることで，様々な周波数フィルタリングを実現できる



（d） 振幅スペクトル$F(k)$とフィルタ関数$H(k)$を掛け合わせた信号

（e） （d）をフーリエ逆変換した信号

図 4.14 周波数領域でのフィルタ処理

フィルタの種類（1次元）

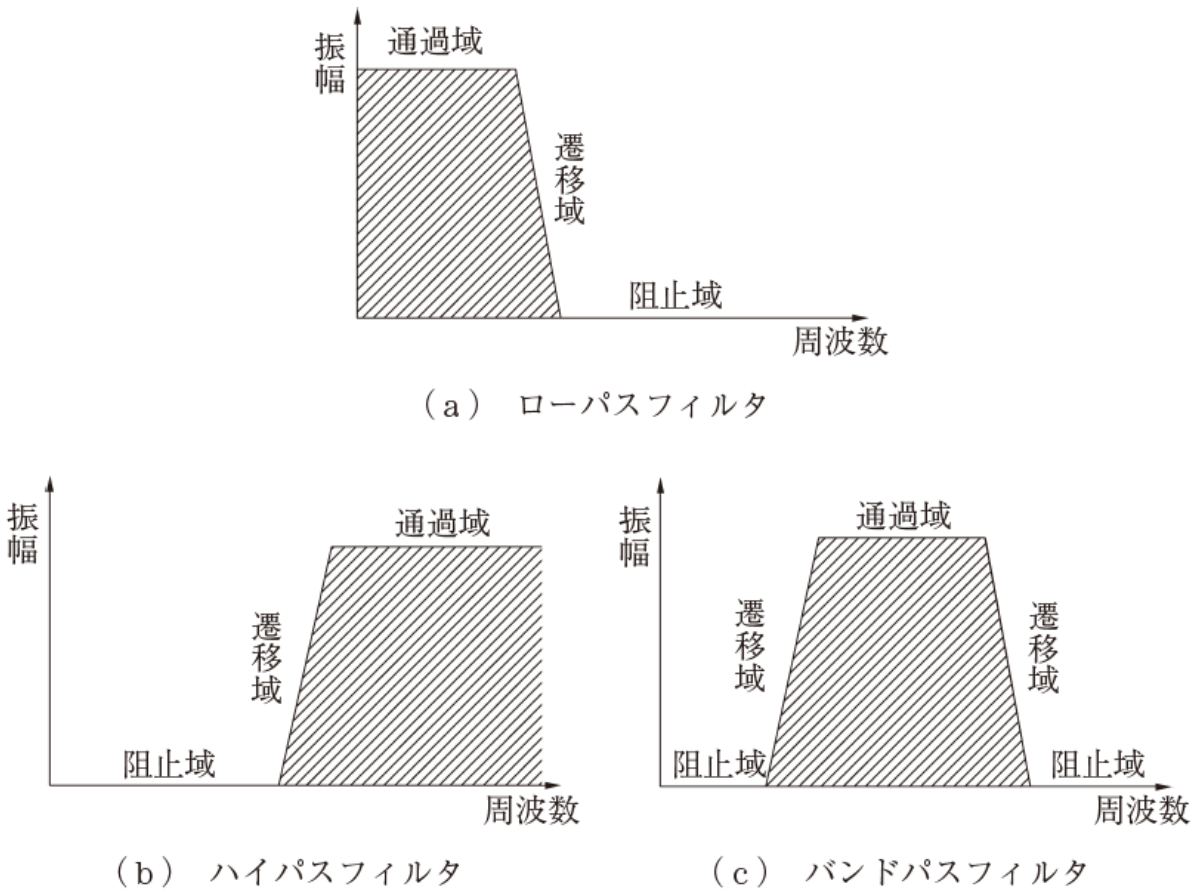
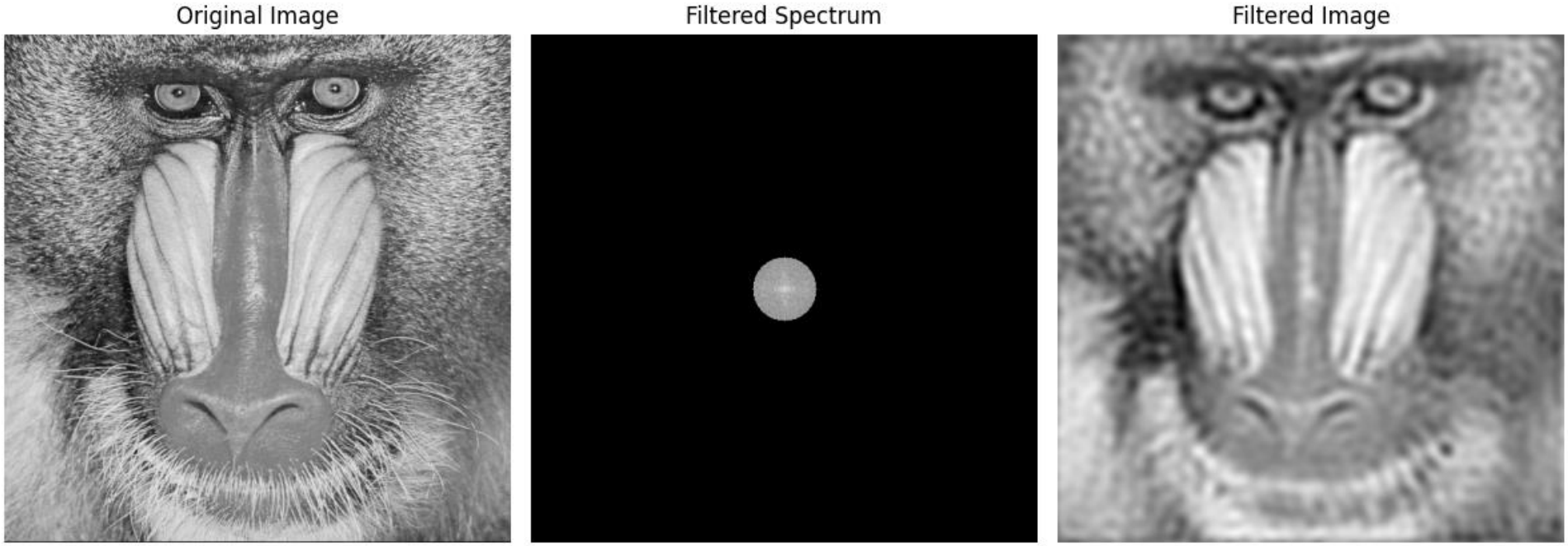


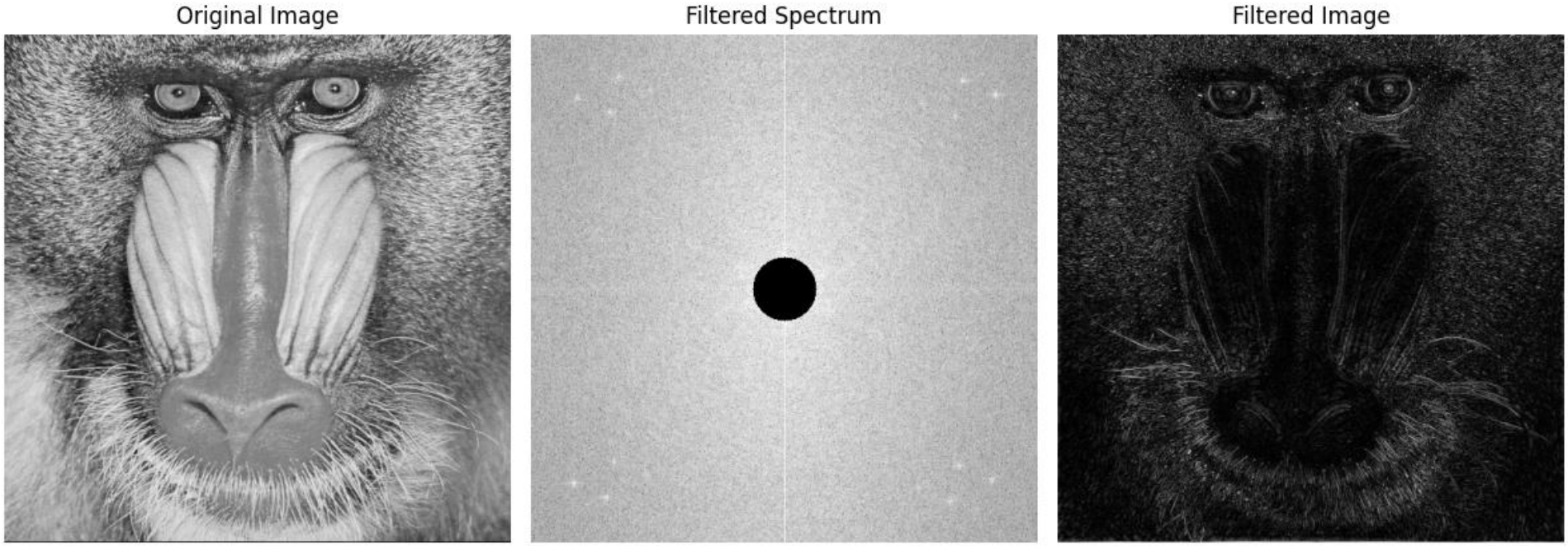
図 4.15 フィルタ関数 $H(\mathrm{k})$ の種類

山田 宏，末松 良一 「画像処理と画像認識 ―AI時代の画像処理入門―」 コロナ社より

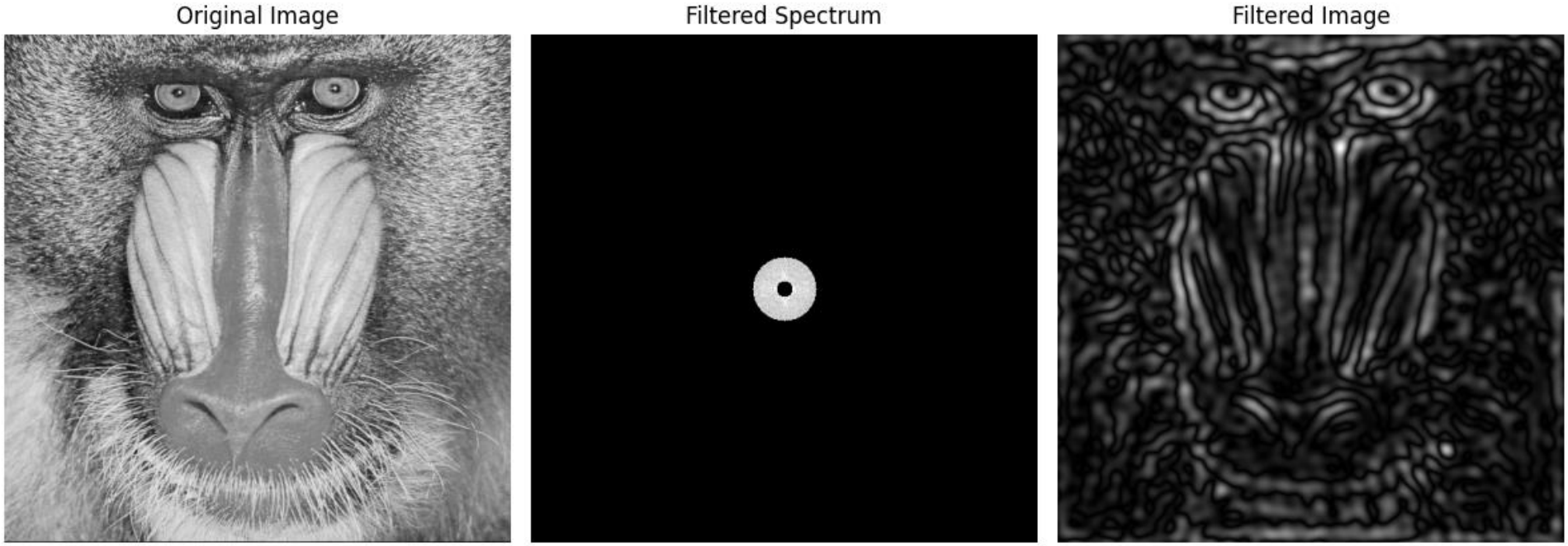
周波数フィルタリング（LPF）



ハイパスフィルタ（HPF）

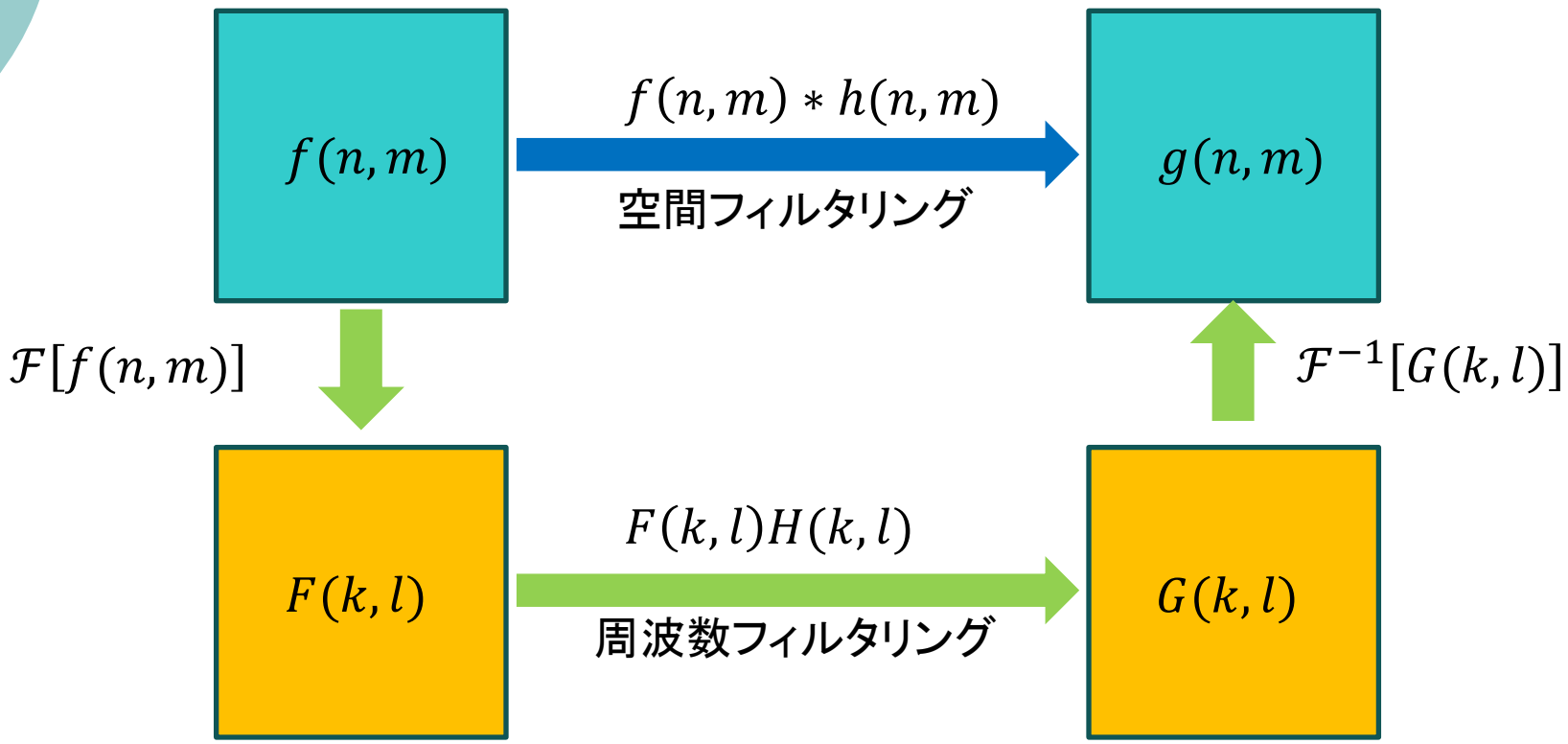


バンドパスフィルタ（BPF）



空間フィルタリングとの関係

畳み込み計算 $g(n,m) = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} f(k,l)h(n-k,m-l)$



畳み込み定理

空間領域の畳み込み計算は以下のように表せる.

$$g(n,m) = f(n,m) * h(n,m)$$
$$= \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} f(k,l)h(n-k,m-l)$$

これは，フーリエ変換を用いて以下のようにも計算でき，
畳み込み定理と呼ばれる.

$$g(n,m) = \mathcal{F}^{-1}[F(k,l) \cdot H(k,l)]$$
$$= \mathcal{F}^{-1}\left[\mathcal{F}[f(n,m)] \cdot \mathcal{F}[h(n,m)]\right]$$

フーリエ変換にFFTを用いれば，空間領域の畳み込み計算よりも
高速に計算できる場合がある.

画像処理プログラミング６
周波数フィルタリング

注意

- 周波数フィルタリングにおいては，これまで画像表示に用いてきた**show_images関数は使用しない**
- フーリエ変換や周波数フィルタリングを行うと，変換前後で画像のダイナミックレンジ（vmin, vmax）が変化することがある
（これまではvmin=0,vmax=255で固定でした）
- 変換前と後の画像それぞれにvmin,vmaxを指定する必要があるが，show_images関数は共通のvmin,vmaxしか受け取れない形で定義してしまっている、、、
- show_images関数を修正するか，サンプルプログラムを参考にその都度表示する処理を書いてください

2次元高速フーリエ変換の実行1

- 2次元フーリエ変換 np.fft.fft2
  - 第1引数：入力画像データ
  - 戻り値：空間周波数スペクトル

```
spectrum = np.fft.fft2(image)
```

- 2次元逆フーリエ変換 np.fft.ifft2
  - 第1引数：入力空間周波数スペクトル
  - 戻り値：画像データ（複素数）

```
image = np.fft.ifft2(spectrum).real
```

※戻り値自体は複素数だが，変換後の画像データは
実部（real part）に保存されるので.realを付けて実部のみ取り出す

2次元高速フーリエ変換の実行2

- 振幅スペクトルの計算 np.abs
  - 第1引数：入力データ（複素数）
  - 戻り値：入力データの絶対値（振幅）

```
amp_spectrum = np.abs(spectrum)
```

  - 以下のように一行で実行することも可能

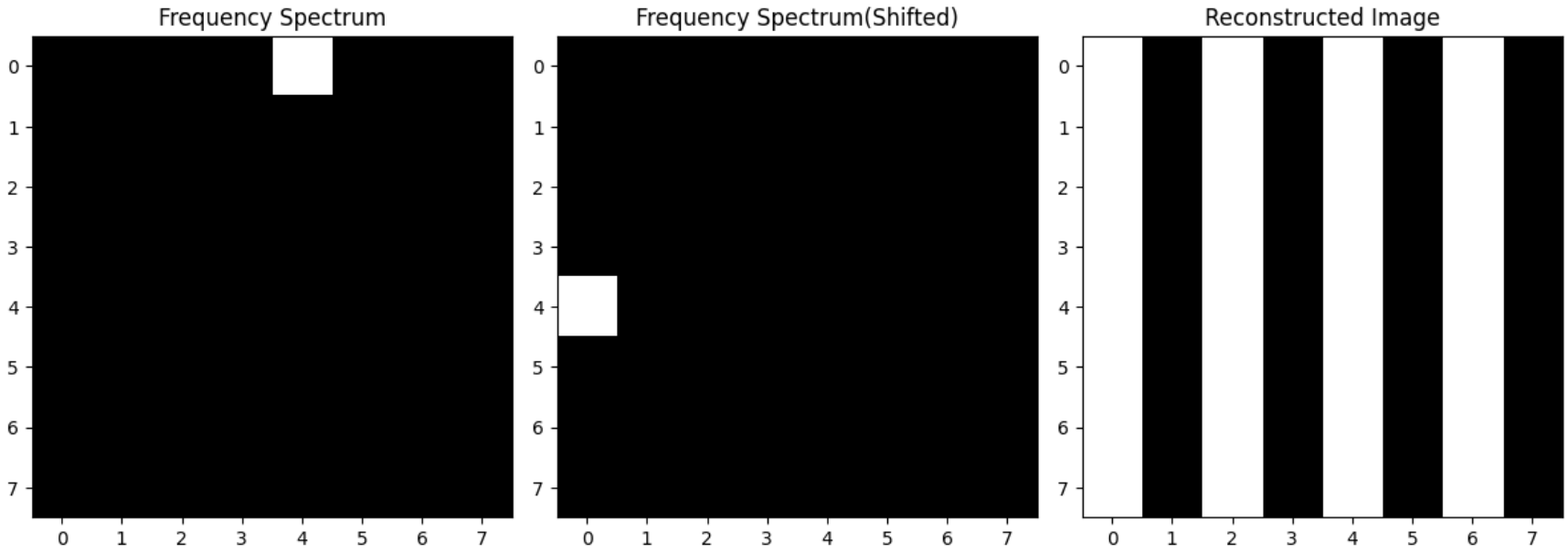
```
amp_spectrum = np.abs(np.fft.fft2(image))
```

- 周波数シフト np.fft.fftshift
  - 空間周波数スペクトルについて，
    4隅低周波⇔中心低周波への変換を行う

```
shifted_spectrum = np.fft.fftshift(spectrum)
```

空間周波数スペクトルと画像の関係

- 8×8画素の空間周波数スペクトルを逆フーリエ変換して画像を確認
- $f_x = 4$のスペクトル⇔周波数4の横縞画像



自然画像の振幅スペクトル

- 一般に自然画像には低周波成分が多く含まれている
- 2次元フーリエ変換を行い振幅スペクトルを求めてそのまま表示すると，低周波成分に大きな値が集中し，高周波成分がほとんど確認できない（値のダイナミックレンジが広すぎる）
- 以下のように対数をとり，ダイナミックレンジの変化を緩やかにして可視化する

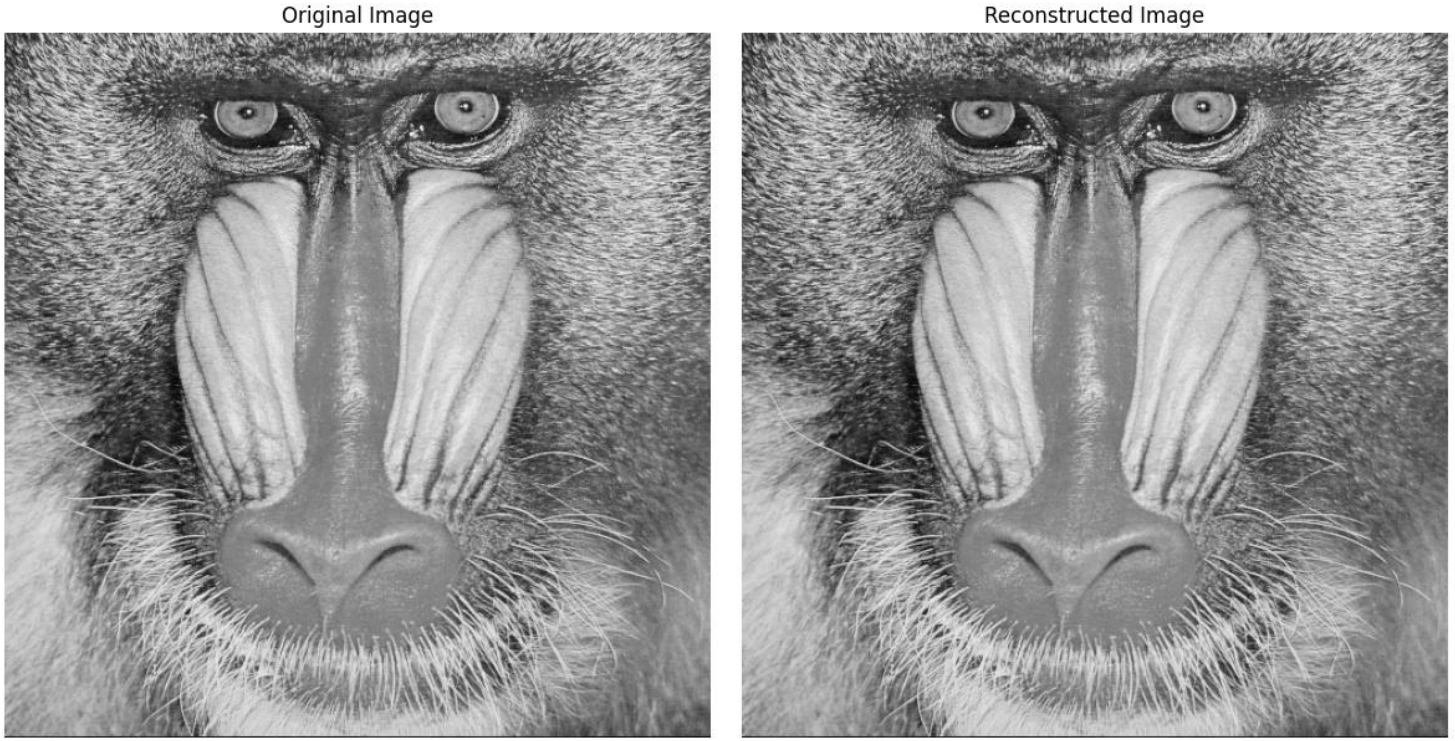
$$\log_{10}(|F(k,l)|+1)$$

```
# 2D-FFT
spectrum = np.fft.fft2(gray_img)
#振幅スペクトルの計算
amp_spectrum = np.log10(np.abs(spectrum)+1)
```

逆フーリエ変換による画像の再構成

- 画像を2D-FFTして求めた空間周波数スペクトルを，そのまま2D-IFFTをすれば元の画像に戻る

  空間領域（画像）⇔空間周波数領域

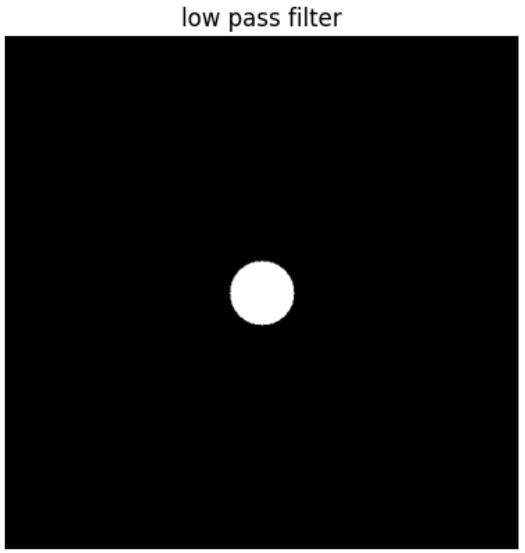


周波数フィルタリング（LPF）

- 円形ローパスフィルタを作成する関数 generate_low_pass_filter(shape, radius)
  - 第1引数(shape)：画像サイズ
  - 第2引数(radius)：フィルタ透過領域の半径
  - 戻り値(mask)：ローパスフィルタ

generate_low_pass_filter関数の使用例

```
# フィルタサイズ
shape = (512,512)
# 透過領域の半径
radius = 32
#ローパスフィルタの生成
low_pass_filter = generate_low_pass_filter(shape, radius)
```



周波数フィルタリングの手順（LPF）

1. 画像を読み込んで，FFT＋周波数シフトを行い
   周波数スペクトルを得る

```
# 2D-FFTと周波数シフト
fourier_spectrum = np.fft.fftshift(np.fft.fft2(gray_img))
```

2. 透過領域の半径を指定してLPFを生成

```
# フィルタリングを行う画像の画素数取得
shape = gray_img.shape

# 透過領域の半径
radius = 32

#ローパスフィルタの生成
low_pass_filter = generate_low_pass_filter(shape, radius)
```

周波数フィルタリングの手順（LPF）

3. 周波数スペクトルとローパスフィルタを乗算し
   周波数フィルタリング

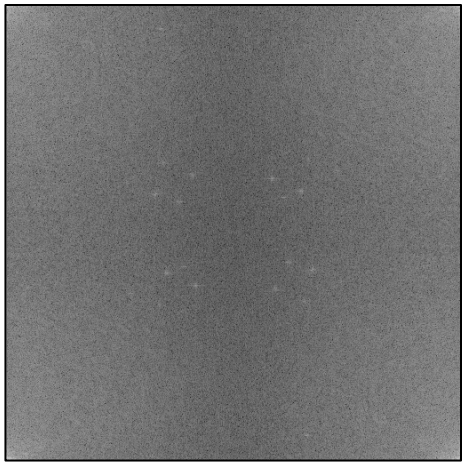
```
# ローパスフィルタの適用
filtered_spectrum = fourier_spectrum * low_pass_filter
```

4. フィルタリング後のスペクトルに対して
   周波数シフト＋IFFTを行い実部を取り出して
   画像を得る

```
# 周波数シフトをしてから逆FFTをかけて実部を取り出す
filtered_img = np.fft.ifft2(np.fft.ifftshift(filtered_spectrum)).real
```

画像処理演習

1. 自然画像の振幅スペクトルについて，
   周波数シフトを行い，中心が低周波となるように
   周波数スペクトルを表示してください．



周波数スペクトル
（4隅が低周波）

周波数スペクトル
（中心が低周波）

画像処理演習

2. generate_low_pass_filterを参考にハイパスフィルタを生成する関数を作成し，周波数フィルタリングを行ってください．

注意：HPFでは画像の直流成分がカットされます
フィルタリング後の結果が負の値になることを防ぐため，絶対値を取ったものを表示してください

```
# 周波数シフトをしてから逆FFTをかけて実部を取り出す
# 直流成分をカットした場合は値が負になる可能性があるため絶対値をとる
filtered_img = np.abs(np.fft.ifft2(np.fft.ifftshift(filtered_spectrum)).real)
```