デジタルメディア処理1

担当: 井尻 敬

Contents

達成目標
- 線形フィルタ（Convolution）の計算方法や性質について正しく説明できる
- フーリエ変換の計算方法や性質について正しく説明できる
- 逆畳み込み(Deconvolution)について正しく説明できる

Contents
- 線形フィルタと畳み込み
- フーリエ変換と畳み込み
- 逆畳み込み Deconvolution

Deconvolution

手振れ・焦点ずれによりボケてしまった画像

ピンボケを模倣した例（点拡がり関数がガウシアン）

ボケのない画像を復元

ボケを引き起こした**点広がり関数**が既知ならば綺麗な復元が可能

Deconvolution

手振れ・焦点ずれによりボケてしまった画像

手ブレを模倣した例（線分状の点広がり関数）

ボケのない画像を復元

ボケを引き起こした**点広がり関数**が既知ならば綺麗な復元が可能

線形フィルタと畳み込み

復習

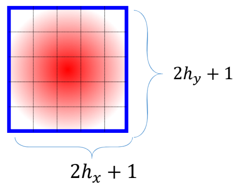
画像の線形フィルタ

周囲画素の重み付和で出力画素値を計算するフィルタ

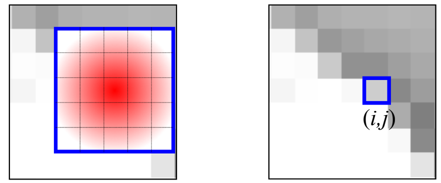
$$I'(i,j) = \sum_{m=-h_y}^{h_y} \sum_{n=-h_x}^{h_x} h(m,n)\ I(i+m, j+n)$$

重みの定義されたフィルタを用意

- サイズ 3x3 5x5 などがよく利用される
- 重みによりいろいろな出力が可能



計算時，各画素を中心にフィルタを重ね合わせ重み付き和を計算



$I(i,j)$ 入力画像　$I'(i,j)$ 出力画像

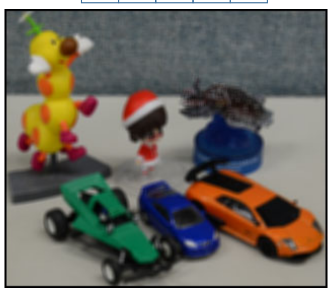
復習

線形フィルタ：平滑化フィルタ

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

サイズ 3x3 : 9近傍の平均



サイズ 5x5 : 25近傍の平均

復習

線形フィルタ：ガウシアンフィルタ

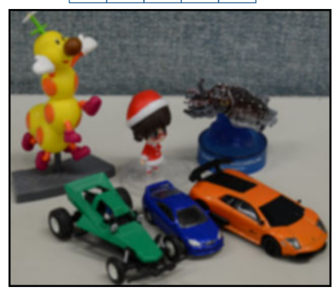
係数をガウス分布に近づけ
中央ほど強い重みに

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

$\frac{1}{256}$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

サイズ 3x3



サイズ 5x5

復習

線形フィルタ :エッジ検出のための**微分フィルタ**

- 単純なフィルタはノイズにも鋭敏に反応する
- ノイズを押さえつつエッジを検出するフィルタが必要

横方向の変化を検出 ： 横方向微分 し 縦方向平滑化 する

縦方向の変化を検出 ： 縦方向微分 し 横方向平滑化 する

Prewitt filter

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Sobel filter

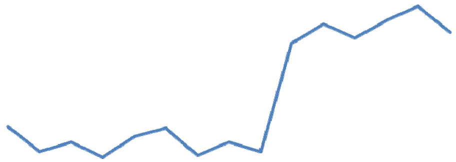
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

復習

1D線形フィルタのイメージ

サイズ3画素の平滑化フィルタ
を掛ける場合

1/3	1/3	1/3



33	33	15	22	11	26	32	12	22	15	94	108	98	111	121	102	102
1/3	1/3	1/3														
	27	23	16	20	23	23	22	16	44	72	100	106	110	111	108	

イメージ：
1. フィルタを左から右に移動し
2. フィルタ範囲の重み付き和を出力する

予習・復習

Convolution(畳み込み)とは

二つの関数 $f(x)$ $g(x)$ を重ね合わせる演算で以下の通り定義される

連続関数

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

離散関数

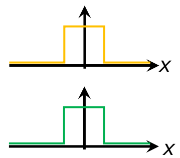
$$(f * g)(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k)$$

※ 関数$f(x)$と$g(x)$を入力すると、新たな関数 $f{*}g\ (x)$ が出力される

練習問題1：
2つの関数f gの畳み込みを求めよ

$$f(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$

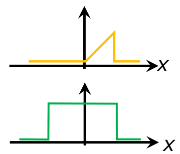
$$g(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



練習問題2：
2つの関数f gの畳み込みを求めよ

$$f(x) = \begin{cases} x & 0 \le x \le 1 \\ 0 & otherwise \end{cases}$$

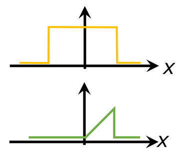
$$g(x) = \begin{cases} 1 & -1 \le x \le 1 \\ 0 & otherwise \end{cases}$$



練習問題3：
2つの関数f gの畳み込みを求めよ

$$f(x) = \begin{cases} 1 & -1 \le x \le 1 \\ 0 & otherwise \end{cases}$$

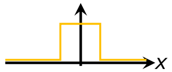
$$g(x) = \begin{cases} x & 0 \le x \le 1 \\ 0 & otherwise \end{cases}$$



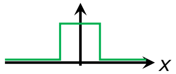
練習問題1：
2つの関数f gの畳み込みを求めよ

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

$$f(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



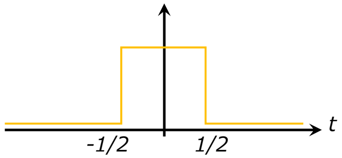
$$\mathrm{g}(x) = \begin{cases} 1 & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$



2つの関数 f(x)とg(x)を畳の込みの式に入れるため変形すると

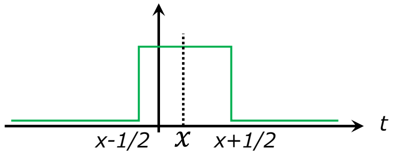
$$f(t) = \begin{cases} 1 & -\frac{1}{2} \le t \le \frac{1}{2} \\ 0 & otherwise \end{cases}$$

※tの関数



$$\mathrm{g}(x-t) = \begin{cases} 1 & x-\frac{1}{2} \le t \le x+\frac{1}{2} \\ 0 & otherwise \end{cases}$$

※tの関数、xは定数として扱う



Convolution(畳み込み)とは

二つの関数$f(x)$ $g(x)$ から一つの関数を出力する演算
以下の通り定義される

連続関数
$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

離散関数
$$(f * g)(n) = \sum_{k=-\infty}^{\infty} f(k)g(n-k)$$

$f(x)$を固定し，$g(x)$を平行移動しながら$f(x)$に掛けあわせ，得られた関数を積分するイメージ
1次元の線形フィルタとほぼ同じ処理をしている

2次元Convolution(畳み込み)

2つの2次元関数$f(x,y)$ $g(x,y)$ より、一つの2次元関数を出力する演算

連続関数
$$(f * g)(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)g(x-u,y-v)dudv$$

離散関数
$$(f * g)(x,y) = \sum_{j=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} f(i,j)g(x-i,y-j)$$

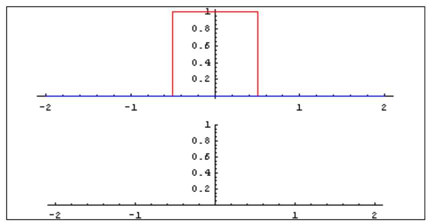
$f(x,y)$を固定し，$g(x,y)$をラスタスキャン順に移動しながら$f(x,y)$に掛けあわせ，得られた関数を積分

※長々と説明しましたが、線形フィルタとほぼ同じ、とおもってOKです。
※違いは『積分域が (-∞, ∞)であること』と『フィルタgの引数がマイナスになった』ところです

まとめ：線形フィルタと畳み込み

- 線形フィルタについて復習
- 畳み込み積分を紹介
  - 2つの関数 f(x)g(x)から一つの関数(f*g)(x)を出力する演算
  - 練習問題を実施
  - 離散系では線形フィルタとほぼ同じ処理

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$



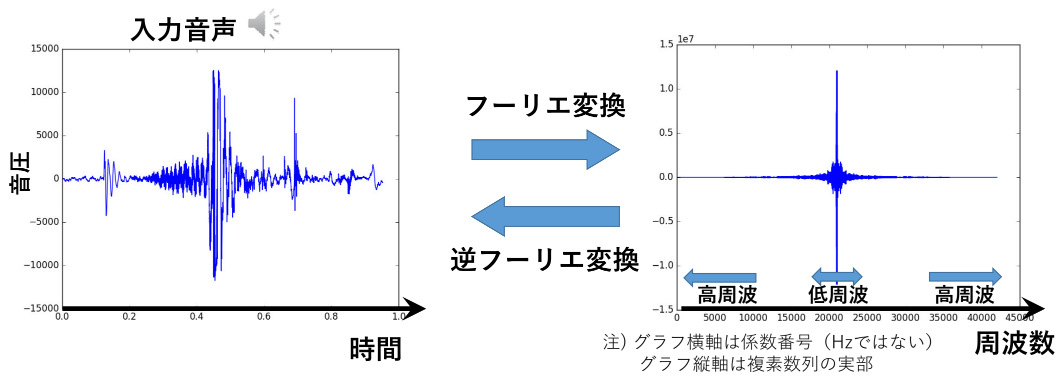
By Lautaro Carmona [CC-BY-SA] from wikipedia

フーリエ変換と畳み込み

FourierSound.py

離散フーリエ変換とは（音）

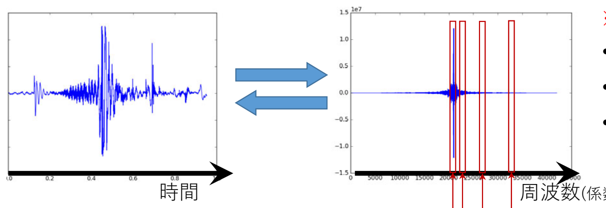
時間に関する離散信号　（横軸が時間の複素数データ）を、
周波数に関する離散信号（横軸が周波数の複素数データ）に変換する



注）グラフ横軸は係数番号（Hzではない）
グラフ縦軸は複素数列の実部

FourierSound.py

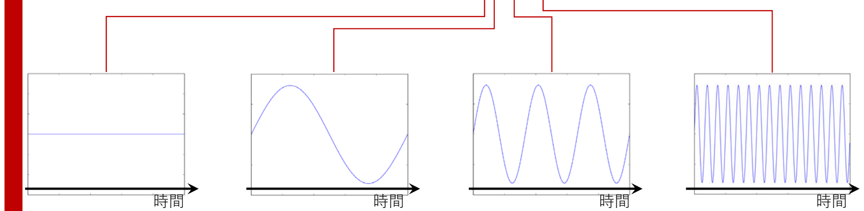
離散フーリエ変換とは（音）



- 離散フーリエ変換後の信号は元信号に含まれる正弦波の量を示す

※正確には正弦波の大きさと位相ずれを複素数で表現

- 中央付近は低周波に対応
- 外側に行くほど高周波に対応
- 中央は直流成分と呼ばれ、定数項に対応
  - 直流成分により全体を上下にオフセットした信号を表現できる

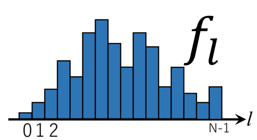


※下の波はイメージ
※本来はもっともっと細かいです。

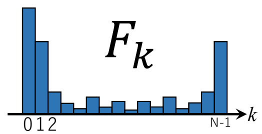
離散フーリエ変換（1D）

フーリエ変換　$F_k = \frac{1}{N}\sum_{l=0}^{N} f_l e^{-i2\pi l \frac{k}{N}}$

逆フーリエ変換　$f_l = \sum_{k=0}^{N-1} F_k e^{i2\pi k \frac{l}{N}}$



長さNの複素数列f_l (実数列のことも多い)



長さNの複素数列F_k

フーリエ変換：『長さNの複素数の列f_l』を『長さNの複素数列F_k』に変換する
逆フーリエ変換：『長さNの複素数の列F_k』を『長さNの複素数列f_l』に変換する

※ f_lが実数の場合**共役対称性**『$F_k = \overline{F_{N-k}}$』を持つ
※わかりやすさのため長さNとしたが正確には周期N

フーリエ変換とは（連続）

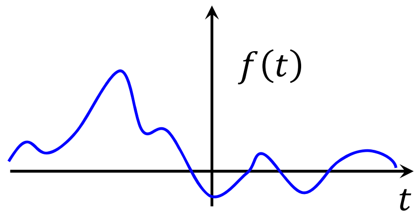
※ この講義では初見です
（詳細は信号処理の教科書へ）

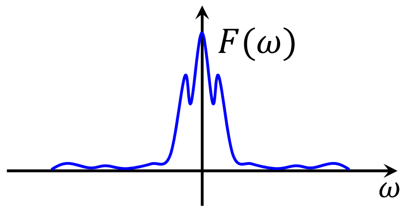
フーリエ変換：

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

逆フーリエ変換：

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega t}\, d\omega$$





フーリエ変換：時間tの関数 $f(t)$ を、周波数ωの関数$F(\omega)$に変換する
逆フーリエ変換：周波数ωの関数$F(\omega)$を、時間tの関数 $f(t)$ に変換する
→ $f(t)$と$F(\omega)$は複素数関数

ガウス関数のフーリエ変換

導出も大切だけど結論も大切。覚えてほしい！

- ガウス関数をフーリエ変換するとガウス関数になる
  - 虚部はゼロになる
  - 標準偏差は逆数になる（裾の広いガウス関数は裾の狭いガウス関数に）

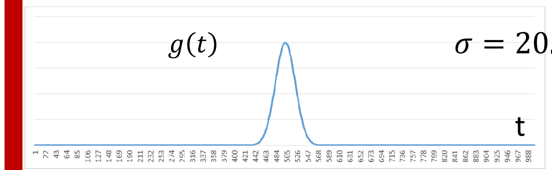
ガウス関数：

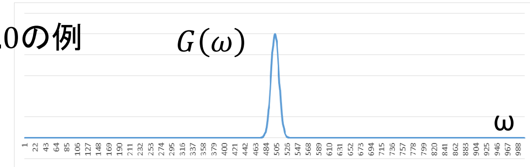
$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}}$$

フーリエ変換：

$$G(\omega) = e^{-\frac{\omega^2}{2\,(1/\sigma^2)}}$$

$\sigma = 20.0$の例





ガウス関数のフーリエ変換(導出)

$$G(\omega) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}} e^{-i\omega t} \mathrm{d}t \quad \text{（定義より）}$$

$$\frac{dG(\omega)}{d\omega} = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}} \frac{de^{-i\omega t}}{d\omega} \mathrm{d}t \quad \text{（両辺微分）}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}} (-it) e^{-i\omega t} \mathrm{d}t \quad \text{（微分実行）}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \sigma^2 \int_{-\infty}^{\infty} \left(-\frac{2t}{2\sigma^2}\right) e^{-\frac{t^2}{2\sigma^2}}\, i e^{-i\omega t} \mathrm{d}t \quad \text{（整理・部分積分準備）}$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \sigma^2 \left(\left[e^{-\frac{t^2}{2\sigma^2}}\, i e^{-i\omega t} \right]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}}\, \omega e^{-i\omega t} \mathrm{d}t \right) \quad \text{（部分積分）}$$

$$= -\omega\sigma^2 G(\omega) \quad \text{（第一項はゼロ、第二項はG(w)なので）}$$

$$\frac{dG(\omega)}{d\omega} = -\omega\sigma^2 G(\omega)$$

以下を利用

$$\frac{d}{dt} e^{-\frac{t^2}{2\sigma^2}} = \left(-\frac{2t}{2\sigma^2}\right) e^{-\frac{t^2}{2\sigma^2}}$$

$$\int \left(-\frac{2t}{2\sigma^2}\right) e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t = e^{-\frac{t^2}{2\sigma^2}}$$

ガウス関数のフーリエ変換(導出)

$$\frac{dG(\omega)}{d\omega} = -\omega\sigma^2 G(\omega) \quad \text{（これは一階の微分方程式なので変数分離で解ける）}$$

$$\frac{dG(\omega)}{G(\omega)} = -\omega\sigma^2 d\omega \quad \text{（変数分離）}$$

$$\log G(\omega) = -\frac{1}{2}\omega^2\sigma^2 + C \quad \text{（両辺を積分、Cは積分定数）}$$

$$G(\omega) = e^C e^{-\frac{1}{2}\omega^2\sigma^2} \quad \text{（整理）}$$

$$G(0) = e^C = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{t^2}{2\sigma^2}} \mathrm{d}t = 1 \quad \text{（積分定数を決める、有名な積分公式利用）}$$

$$G(\omega) = e^{-\frac{\omega^2\sigma^2}{2}} \quad \text{（求まった積分定数を代入して，フーリエ変換が得られた）}$$

畳み込み積分のフーリエ変換

導出も大切だけど、結論も大切。
覚えてほしい！導出は次項の補足資料へ

$h(x), f(x), g(x)$ のフーリエ変換を $H(\omega), F(\omega), G(\omega)$ とすると以下が成り立つ,

$h(x) = (f * g)(x)$ ⟺ $H(\omega) = F(\omega)G(\omega)$

- **$h(x)$ が $f(x)$ と $g(x)$ の畳み込みにより得られるとき、$h(x)$ のフーリエ変換 $H(\omega)$ は、フーリエ変換した $F(\omega)$ と $G(\omega)$ の積になる**
- 畳み込みの高速化が可能
  - Step1 フーリエ変換して $f(x), g(x) \rightarrow F(\omega), G(\omega)$　　O(N log N)
  - Step2 周波数空間で積を計算し $H(\omega) = F(\omega)G(\omega)$　　O(N)
  - Step3 逆フーリエ変換する $H(\omega) \rightarrow h(x)$　　O(N log N)

※ 単純な畳み込み $f * g$ を計算すると O(N^2)

畳み込み積分のフーリエ変換（導出）

導出も大切だけど、結論も大切。
覚えてほしい！

$$h(x) = \int_{-\infty}^{\infty} f(t)g(x-t)\mathrm{d}t \quad \text{(畳み込み積分の定義)}$$

$$H(\omega) = \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} f(t)g(x-t)\mathrm{d}t \right) e^{-ix\omega}\, dx \quad \text{(hをフーリエ変換)}$$

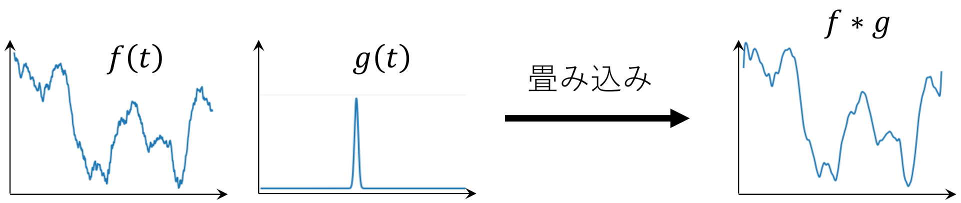
$$= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(x-t)\, e^{-ix\omega} dx \right) dt \quad \text{(整理)}$$

$$= \int_{-\infty}^{\infty} f(t) \left(\int_{-\infty}^{\infty} g(u)\, e^{-i(u+t)\omega} du \right) dt \quad (u = x-t, dx = du \ \text{と変数変換})$$

$$= \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \left(\int_{-\infty}^{\infty} g(u)\, e^{-i\omega u} du \right) dt \quad \text{(整理)}$$

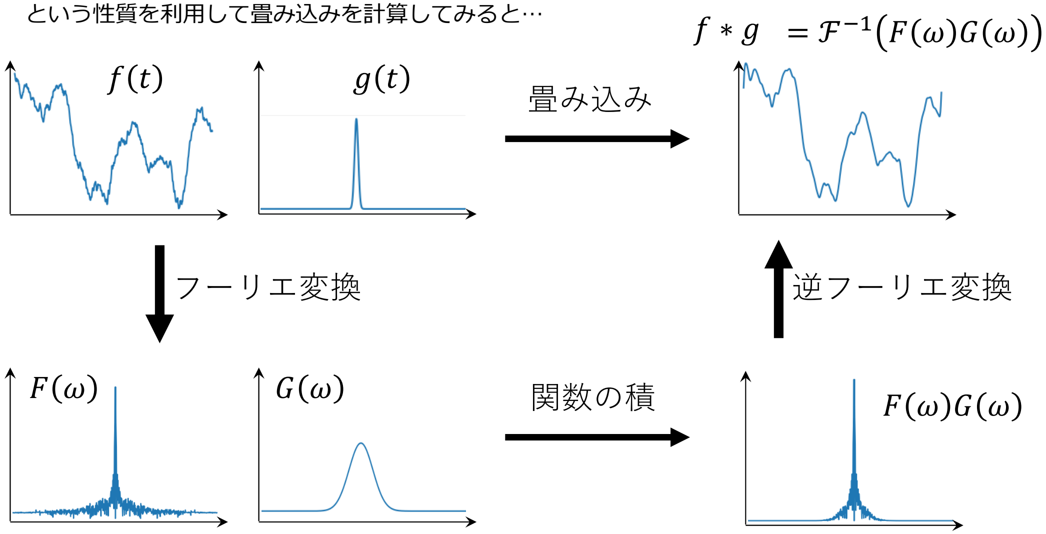
$$= F(\omega)G(\omega)$$

畳み込みと周波数フィルタリング

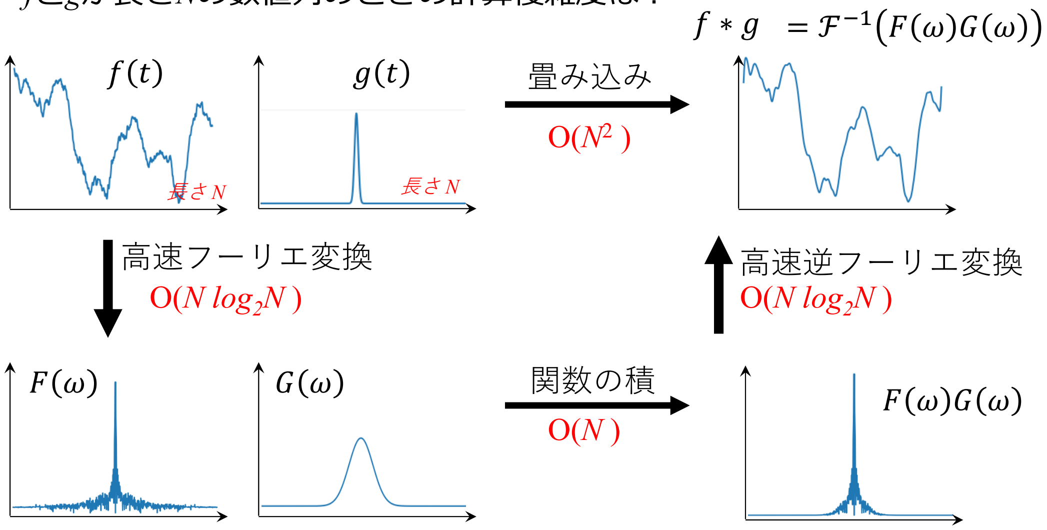


※ 関数 $f(t)$ に Gaussian g(t)を畳み込むことで、細かな変化が平滑化されている

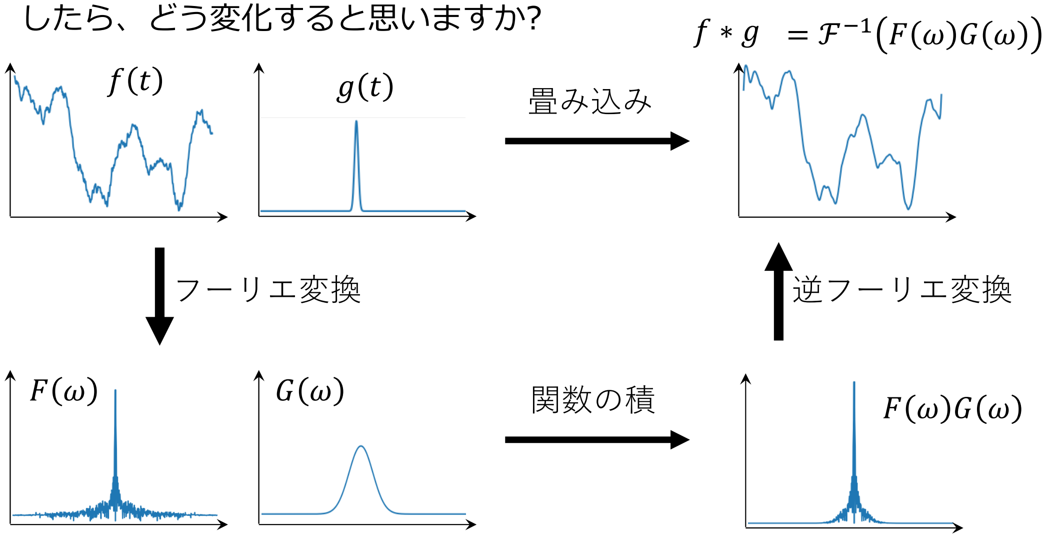
「畳み込みのフーリエ変換は、周波数空間では関数の積になる」
という性質を利用して畳み込みを計算してみると…



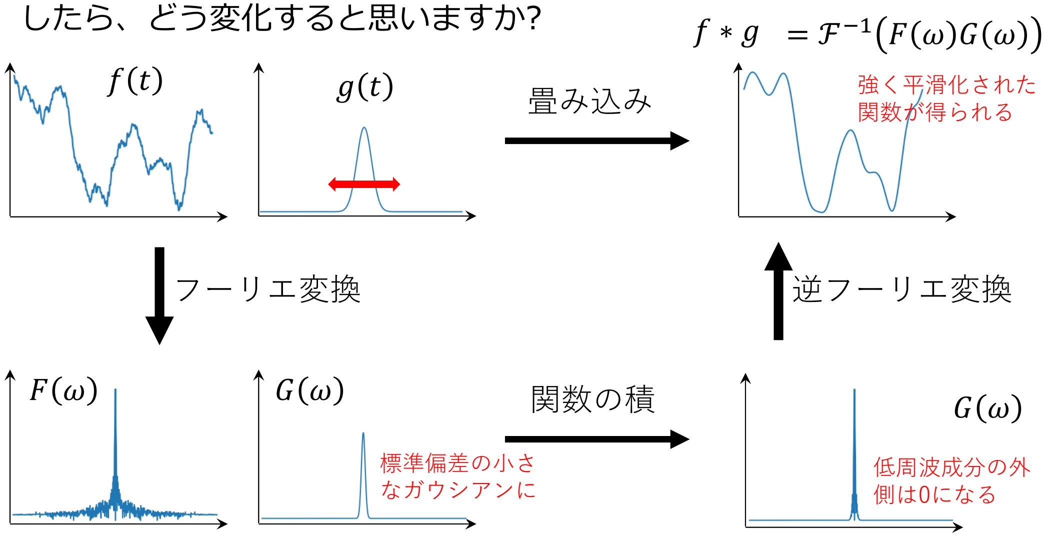
fとgが長さNの数値列のときの計算複雑度は？



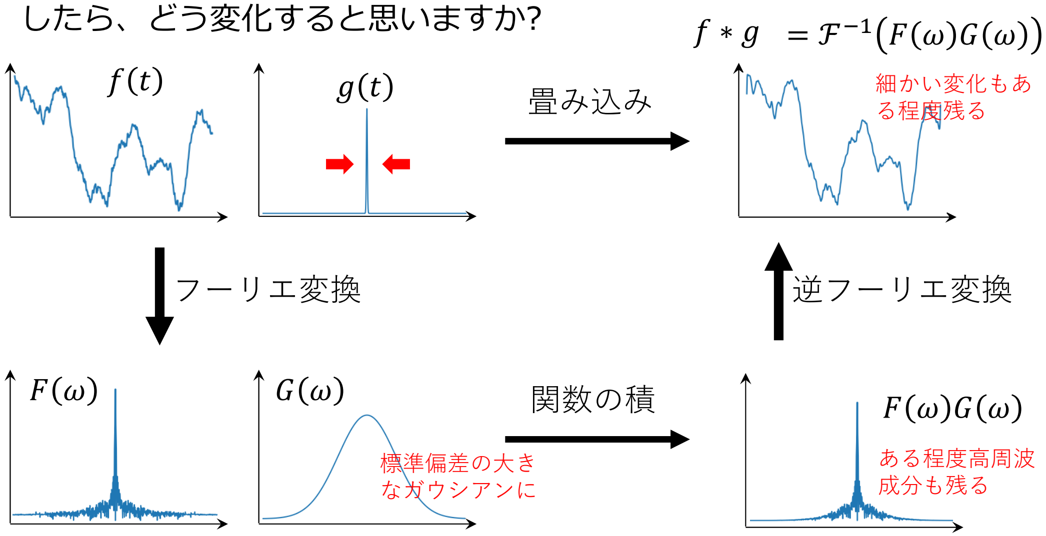
畳み込むガウシアンの標準偏差を大きく
したら、どう変化すると思いますか?



畳み込むガウシアンの標準偏差を大きく
したら、どう変化すると思いますか?



畳み込むガウシアンの標準偏差を小さく
したら、どう変化すると思いますか?



まとめ：フーリエ変換と畳み込み

フーリエ変換に付いて復習し、2つの重要な性質に付いて紹介した

1. ガウス関数をフーリエ変換するとガウス関数になる

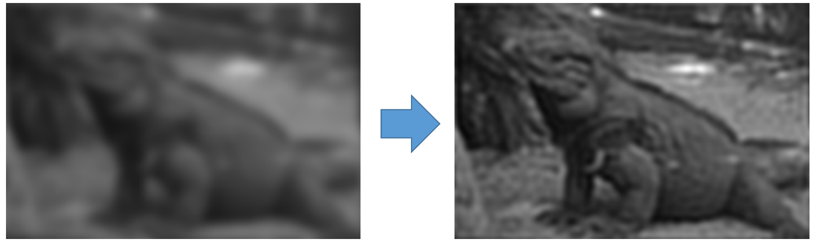
$$g(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \qquad G(\omega) = e^{-\frac{\omega^2}{2(1/\sigma^2)}}$$

2. 畳み込み積分のフーリエ変換はフーリエ変換の積になる

$$h(x) = \int_{-\infty}^{\infty} f(x-t)g(t)dt \Leftrightarrow H(\omega) = F(\omega)G(\omega)$$

$H(\omega), F(\omega), G(\omega)$は, $h(x), h(x), h(x)$のフーリエ変換

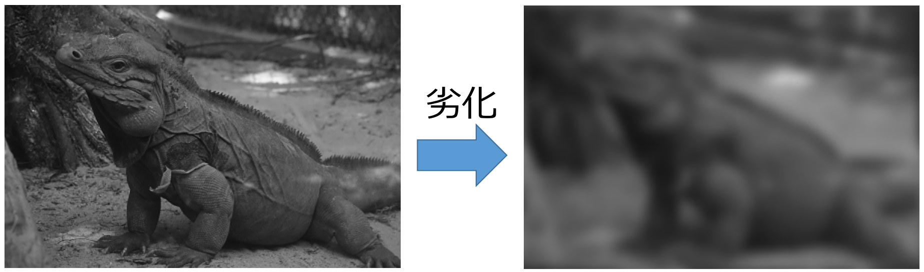
この性質を利用し周波数空間にて畳み込みを計算する
方法を紹介した



逆畳み込み（Deconvolution）

画像の劣化モデル（1/2）

- 写真撮影の際、手ブレ・ピンボケ・機器ノイズ等のため劣化した画像が取得される
- 劣化前の画像を取得したい
- 劣化前画像復元のため劣化課程をモデル化（数式表現）する



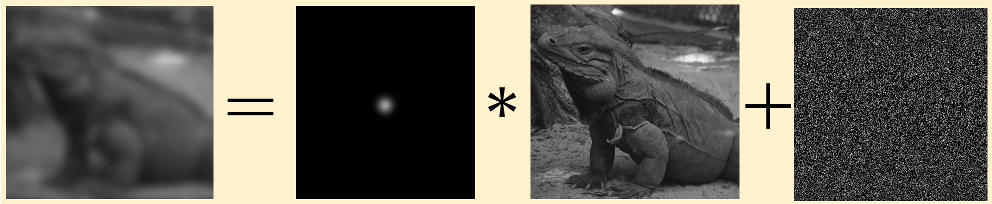
$f(x,y)$：劣化の無い理想画像
※実際にこれを撮影するのは困難

$g(x,y)$：劣化画像
※ 手ブレ・ピンボケ・ノイズを含む

画像の劣化モデル（2/2）

ここでは画像の劣化モデルを以下のとおり定義する

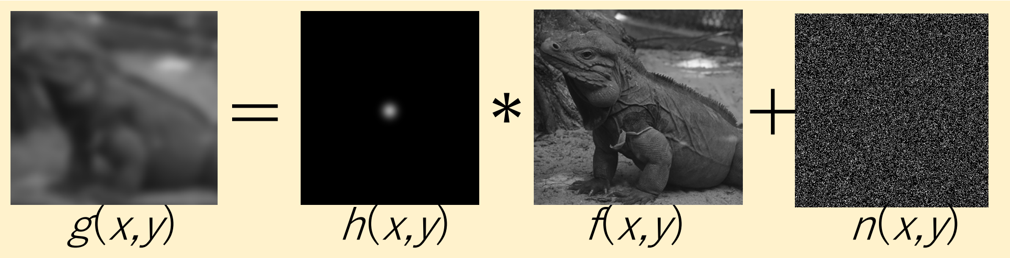
$$g(x,y) = h(x,y) * f(x,y) + n(x,y)$$



$g(x,y)$ $h(x,y)$ $f(x,y)$ $n(x,y)$

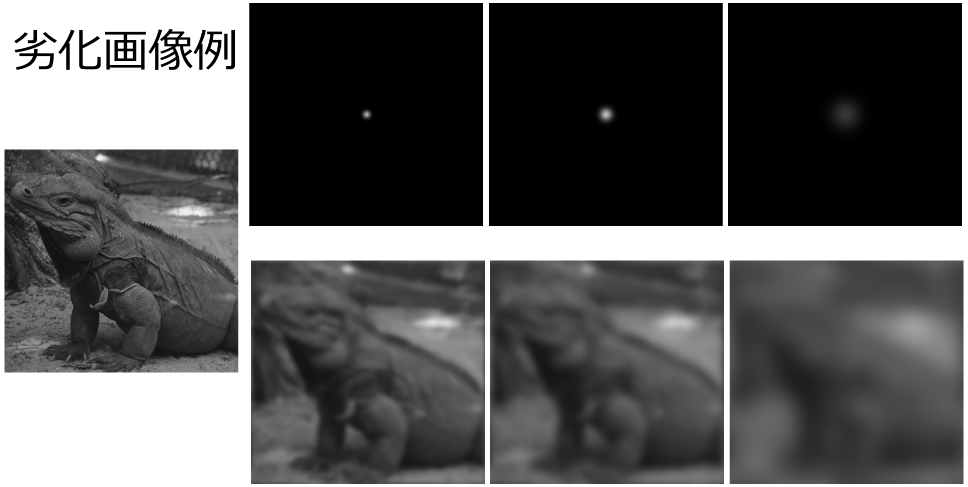
元画像に関数h(x,y)が畳み込みこまれ、ノイズ画像n(x,y)が加算されて 画像が劣化
観測されるのは g(x,y), 取得したいのはf(x,y)
h(x,y)はボケの様子を表すもので **点広がり関数** と呼ばれる

点広がり関数 (PSF: point spread function)

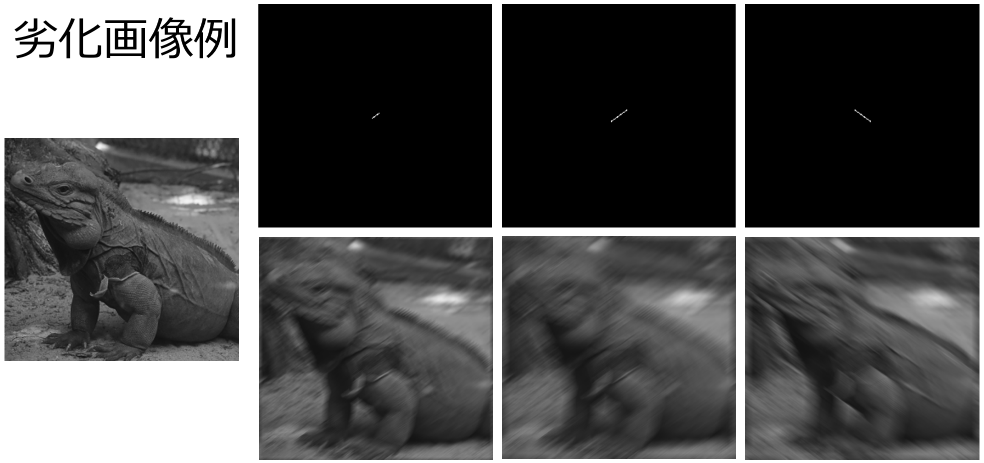


- 画像劣化時に元画像に畳み込まれている関数$h(x,y)$のこと
- 劣化の特性を表す
- 元画像が点光源のとき、劣化画像に表れる応答を表すため、点広がり関数（PSF）やインパルス応答と呼ばれる

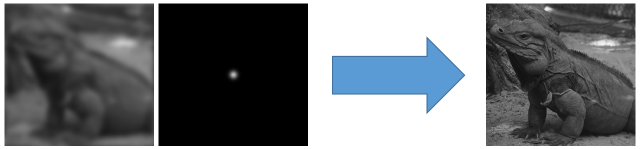
劣化画像例



劣化画像例



劣化画像の復元　(単純な手法)



- **劣化画像**と**点広がり関数**から元画像を取得する問題を考える

$g(x,y) = h(x,y) * f(x,y) + n(x,y)$　　**gと hは既知で fがほしい**

$G(u,v) = H(u,v)F(u,v) + N(u,v)$　　両辺をフーリエ変換（大文字で表現）

$G(u,v) \approx H(u,v)F(u,v)$　　ちょっと強引だけどノイズの影響を無視

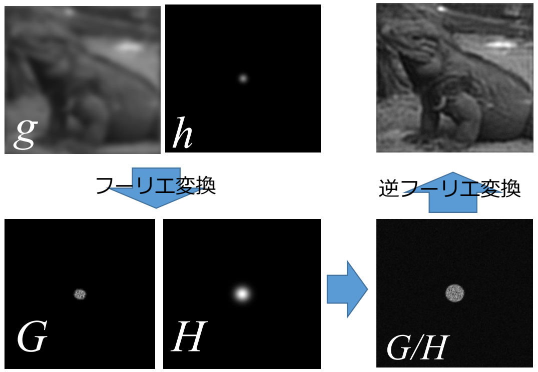
$F(u,v) \approx \frac{1}{H(u,v)} G(u,v)$　　Fについて整理

$f(x,y) \approx \mathcal{F}^{-1}\left(\frac{1}{H(u,v)} G(u,v)\right)$　　*両辺逆フーリエ変換*

劣化画像の復元 （単純な手法）

$$f(x,y) \approx \mathcal{F}^{-1}\left(\frac{1}{H(u,v)}G(u,v)\right)$$

f : 元画像
G : 劣化画像のフーリエ変換
H : 点広がり関数のフーリエ変換



この手法の問題

- ノイズを無視
- Hは高周波部分でほぼゼロ

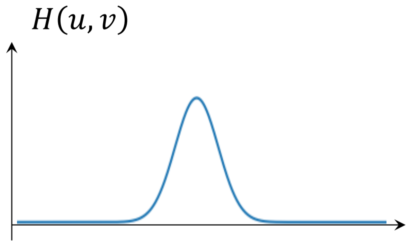
→ 単純に G/H を実装するとノイズが強調される

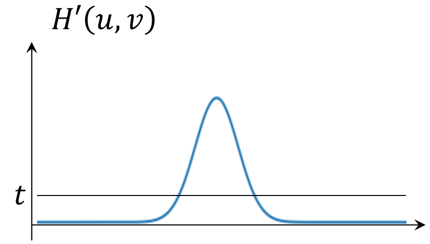
左の例ではHを以下の通り拡張した

$$H'(u,v) = \begin{cases} t & |H(u,v)| < t \\ H(u,v) & otherwise \end{cases}$$

ここでは t=0.01を利用

$$\mathrm{H}'(u,v) = \begin{cases} t & |H(u,v)| < t \\ H(u,v) & otherwise \end{cases}$$

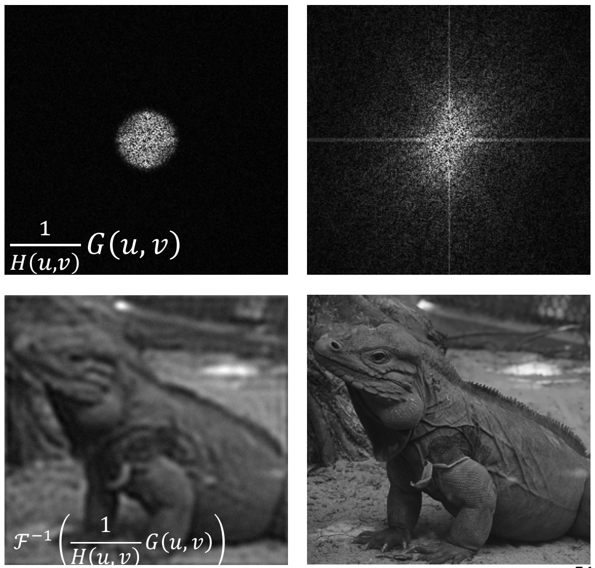




正解との比較

- 右が正解
- 左が復元手法

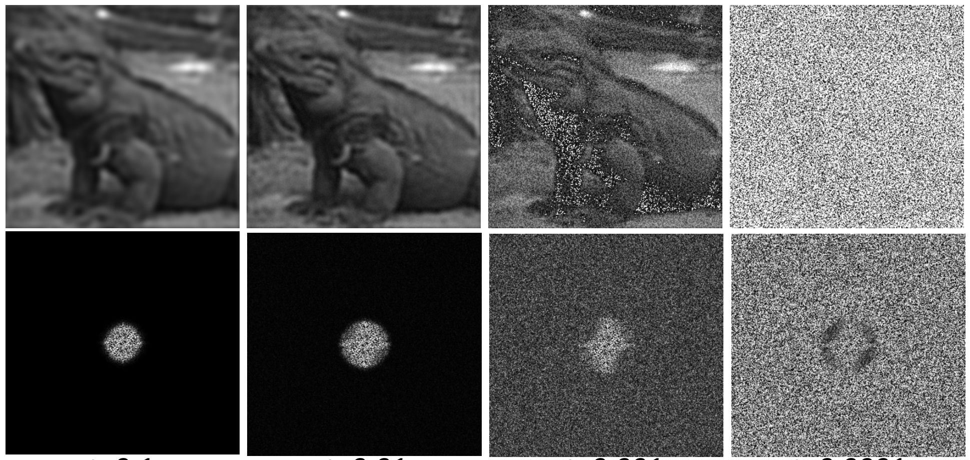
※Hに対する閾値処理の影響が見られる
※閾値処理を行なわないと高周波成分に存在するノイズが強調され画像はうまく復元されない



tの影響

$$H'(u,v) = \begin{cases} t & H(u,v) < t \\ H(u,v) & otherwise \end{cases}$$

sigma = 5.0を利用



t=0.1　t=0.01　t=0.001　t=0.0001

劣化画像の復元：Wiener filter

先の単純な手法の問題点（ノイズ無視・Hがゼロに近い場所で困る）を改善したい

$$f(x,y) \approx \mathcal{F}^{-1}\left(\frac{H^*(u,v)}{|H(u,v)|^2+\epsilon}G(u,v)\right)$$

f：元画像
G：劣化画像のフーリエ変換
H：点広がり関数のフーリエ変換
H^*: 共役複素数

周波数空間において観測画像$G(u,v)$に$M(u,v)$をかけて復元画像が得られると仮定
元画像のフーリエ変換$F(u,v)$ と 復元画像のフーリエ変換 $G(u,v)M(u,v)$の差を最小化

$$\underset{M}{\operatorname{argmin}} \sum_u \sum_v \left(F(u,v) - M(u,v)G(u,v)\right)^2$$

この解として以下が得られる

$$M(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2+|N(u,v)|^2/|F(u,v)|^2}$$

ここで、ノイズNと元画像Fは不明なので$|N(u,v)|^2/|F(u,v)|^2 = \epsilon$ 置いて上の式が得られる

導出の参考 [http://web.engr.oregonstate.edu/~sinisa/courses/OSU/ECE468/lectures/ECE468_13.pdf]

Wiener filter 適用例

劣化画像

Wiener filter

単純な手法 1/H

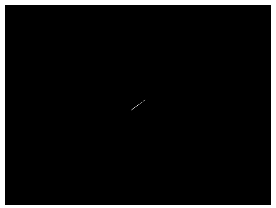
σ = 6のガウシアン
ε = 0.00001

Wiener filter 適用例

劣化画像

Wiener filter

単純な手法 1/H



w = 20, θ=0.8π の線分カーネル
ε = 0.00001

まとめ: Deconvolution

- Deconvolution（逆畳み込み）とは,
  - 『畳み込み(convolution)は，周波数空間では関数どうしの積になる』という特徴を利用し，劣化画像 g（畳み込み後）と **点広がり関数** h から，元画像を復元する処理のこと
- 以下二つのフィルタを紹介した
  - 点広がり関数の逆数を利用した単純なフィルタ
  - ノイズを考慮し**復元画像と元画像の誤差を最小化する Wiener filter**
- 今回は点広がり関数を既知としたが，これも同時に推定する手法もある

$$f(x,y) \approx \mathcal{F}^{-1}\left(\frac{1}{H(u,v)}G(u,v)\right)$$

$$f(x,y) \approx \mathcal{F}^{-1}\left(\frac{H^*(u,v)}{|H(u,v)|^2+\epsilon}G(u,v)\right)$$

f：元画像
G：劣化画像のフーリエ変換
H：点広がり関数のフーリエ変換
H^*: 共役複素数