情報デザイン専攻

画像情報処理論及び演習II

-周波数分解-

FFT、Gaussianフィルタと周波数分解

第4回講義
水曜日１限
教室6218

吉澤　信
shin@riken.jp, 非常勤講師
大妻女子大学　社会情報学部
独立行政法人 理化学研究所

今日の授業内容

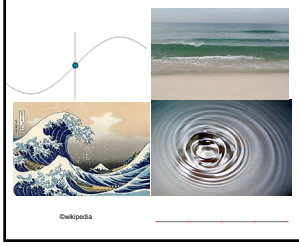
www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec16.pdf

レポート04〆切10月21日(水)

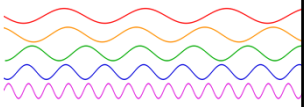
1. 前回の続き.
2. Gaussianフィルタと周波数分解.
3. 演習：前回・前々回の続き＋高速離散コサイン変換によるGaussianフィルタと周波数分解.

復習：周波数(Frequency)

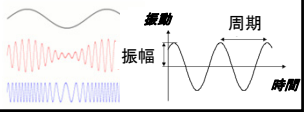
- ✓ 周波数・振動数：波動・振動周期の逆数(1/周期).
- ✓ 周期(Period)：１循環するまでの時間.
- ✓ 振幅（Amplitude）：振動の大きさ.



低周波：ゆるやか＝大きな特徴

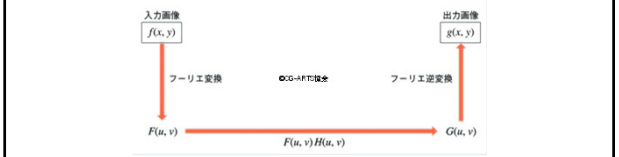


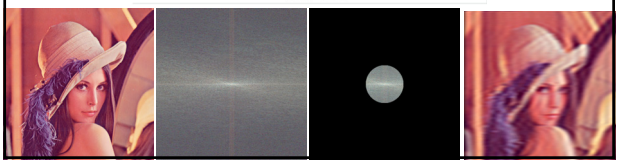
高周波：こまやか＝シャープな特徴



復習：周波数操作

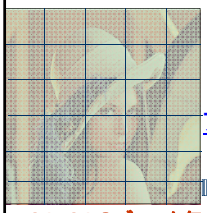
入力画像 → 変換 → 周波数 → 処理 → 処理後の周波数 → 逆変換 → 出力画像





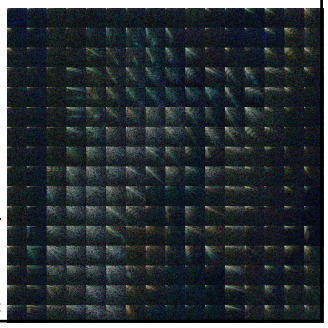
復習：離散コサイン変換(DCT)

- ✓ 非常に処理が重いので、FFTを使わない簡単な実装は画像を部分画像(ブロック)に分割してブロック毎に変換する:



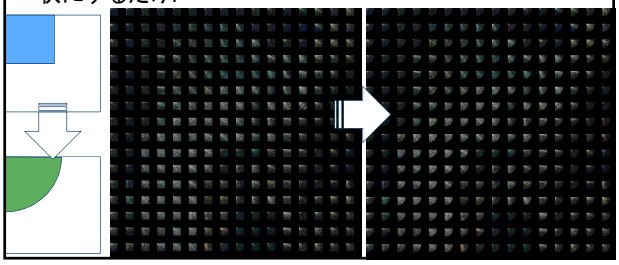
ブロック DCT →

32x32のブロック毎のDCT例:



前回の演習：周波数フィルタ

- ✓ testDCT.cxxを編集して円状に高周波をゼロにするローパスフィルタを作ってみましょう！
- ✓ 16x16のブロックで半径1,2,3,4,8で実行してみてください.
- ✓ ヒント：testDCT.cxxは四角に低周波を残しているので、円状にするだけ.

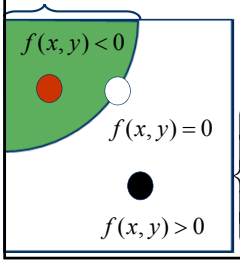


復習:円の式

✓ 半径rの円の式は、

$$x^2 + y^2 = r^2$$

r つまり、座標(x,y)が上記式を満たす点の集合が円.



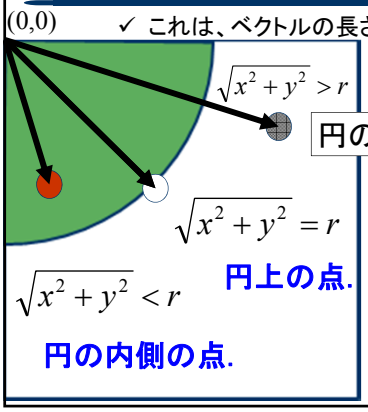
$$f(x,y) = x^2 + y^2 - r^2$$

とすると(x,y)は、

$$\begin{cases} f(x,y)=0 & \text{円上の点.} \\ f(x,y)>0 & \text{円の外側の点.} \\ f(x,y)<0 & \text{円の内側の点.} \end{cases}$$

復習:円の式

✓ これは、ベクトルの長さで考えても同じ:



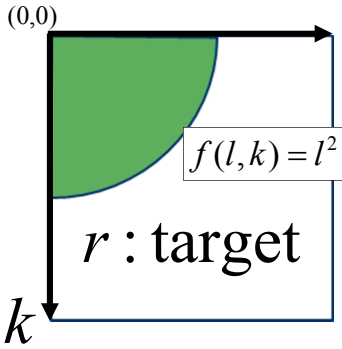
$$f(x,y) = x^2 + y^2 - r^2$$

とすると(x,y)は、

$$\begin{cases} f(x,y)=0 & \text{円上の点.} \\ f(x,y)>0 & \text{円の外側の点.} \\ f(x,y)<0 & \text{円の内側の点.} \end{cases}$$

復習:円の式

✓ プログラム内では、



$$f(l,k) = l^2 + k^2 - \text{target}^2$$

$$f(x,y) = x^2 + y^2 - r^2$$

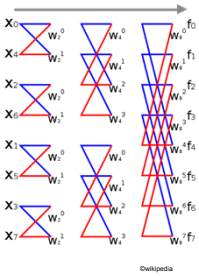
とすると(x,y)は、

$$\begin{cases} f(x,y)=0 & \text{円上の点.} \\ f(x,y)>0 & \text{円の外側の点.} \\ f(x,y)<0 & \text{円の内側の点.} \end{cases}$$

高速フーリエ変換(FFT)

✓ 再帰的に N = A*Bサイズの変換を、より小さいAとBの変換を行う事で高速化(バタフライ演算):計算量は $N^2 \to 2N\log_2 N$

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & W^0 & 0 \\ 0 & 1 & 0 & W^1 \\ 1 & 0 & W^2 & 0 \\ 0 & 1 & 0 & W^3 \end{bmatrix} \begin{bmatrix} W_2^0 & W_2^0 & 0 & 0 \\ W_2^0 & W_2^1 & 0 & 0 \\ 0 & 0 & W_2^0 & W_2^0 \\ 0 & 0 & W_2^0 & W_2^1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

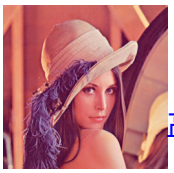


http://www.wisdom.weizmann.ac.il/~naor/COURSE/fft-lecture.pdf

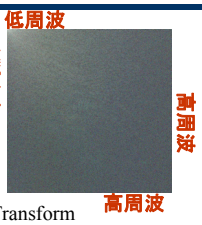
✓ FFT(高速DCT/DSTを含む)のCライブラリ:
- FFTW(n次元対応): www.fftw.org
- fftsg(1-3次元:京大、大浦さん):演習Ex09.zipのfftsg.hとfftsg2d.h

www.kurims.kyoto-u.ac.jp/~ooura/fft-j.html

高速離散コサイン変換(FDCT)

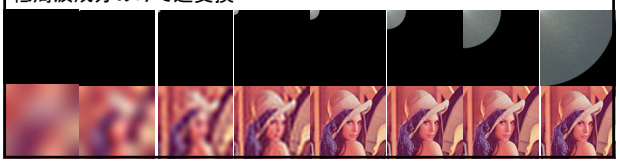


高速DCT



FDCT: Fast Discrete Cosine Transform

低周波成分のみで逆変換 ⟷ 高周波成分も使って逆変換



輝度値の(線形補間による)正規化

✓ DCTによる周波数は実数(負の値も含む浮動小数).

✓ パワースペクトルの可視化はlog()を使う、

$$|F(u,v)|^2 \to \log(1 + |F(u,v)|^2)$$

✓ どちらにしろ8bit画像(0-255)で表示するには正規化が必要:

$$I(u,v) \leftarrow 255\frac{I(u,v) - \min}{\max - \min}$$

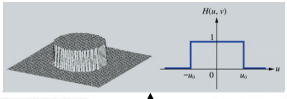
ここで、max,minはそれぞれ輝度値の最大・最小値.

✓ 注意点:周波数を用いた処理は正規化しない
→可視化のときだけ正規化やlog()を使う.

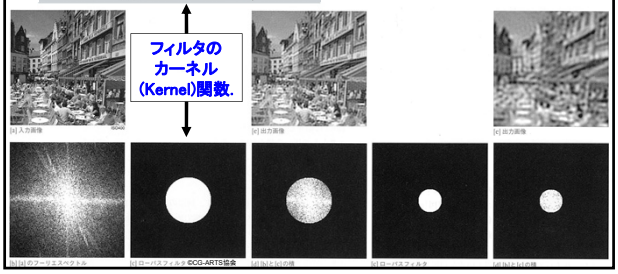
Shin Yoshizawa: shin@riken.jp

復習：ローパスフィルタ(Box関数)

✓ −u0から、u0までの低周波数成分だけ残す.



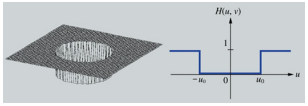
周波数の高い横方向の波（縦縞）を消す.



Shin Yoshizawa: shin@riken.jp

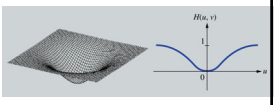
復習：ハイパスフィルタ

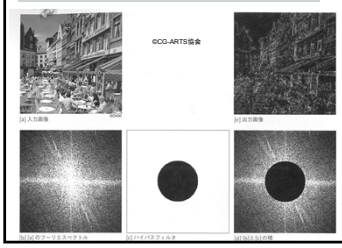
✓ −u0から、u0までの高周波数成分だけ残す.



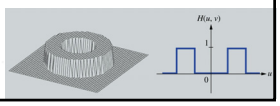
$$H_{high}(u,v) = 1 - H_{low}(u,v)$$

✓ 1からロウパスを引く:



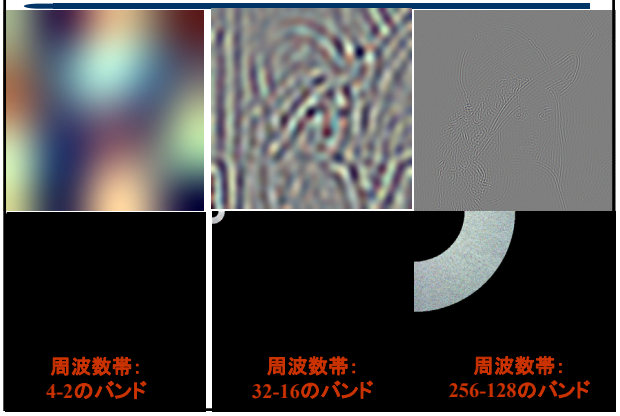


✓ バンドパスフィルタ：特定周波数成分の抽出.



Shin Yoshizawa: shin@riken.jp

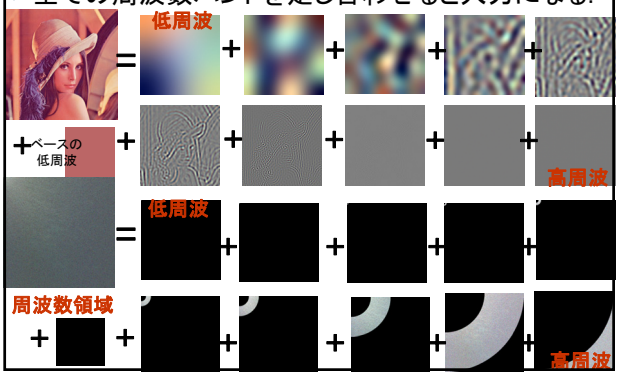
FDCTによるバンドパスフィルタ



Shin Yoshizawa: shin@riken.jp

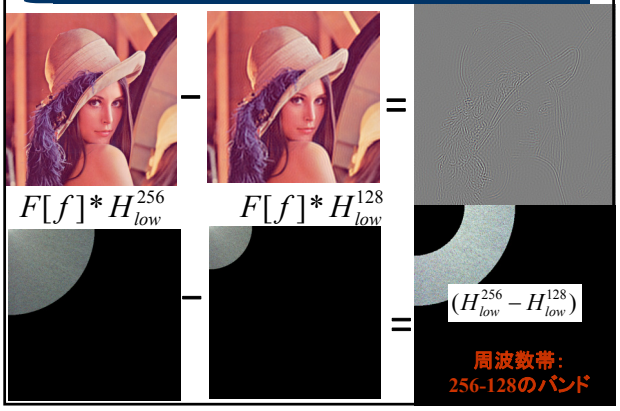
FDCTによる周波数分解

✓ 全ての周波数バンドを足し合わせると入力になる.



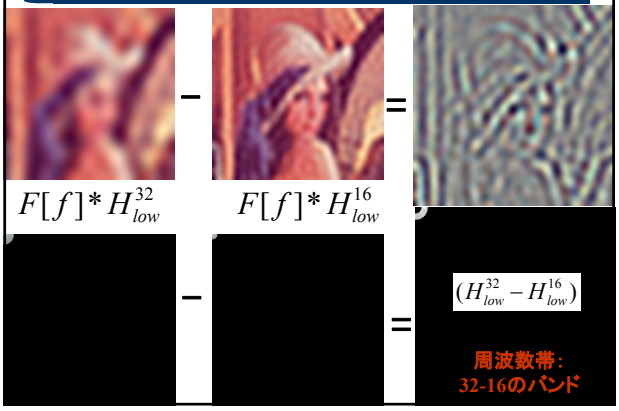
Shin Yoshizawa: shin@riken.jp

差分によるバンドパス



Shin Yoshizawa: shin@riken.jp

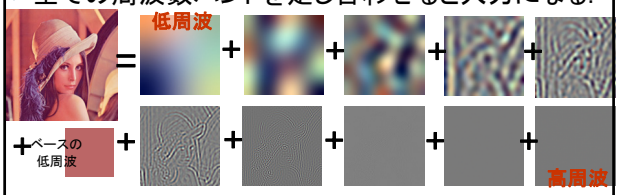
差分によるバンドパス2



Shin Yoshizawa: shin@riken.jp

FDCTによる周波数分解2

✓ 全ての周波数バンドを足し合わせると入力になる.



$$f = F^{-1}[(F[f]*H^2_{low} - F[f]*H^1_{low}) + (F[f]*H^4_{low} - F[f]*H^2_{low}) + (F[f]*H^8_{low} - F[f]*H^4_{low}) + \ldots + (F[f]*H^{1024}_{low} - F[f]*H^{512}_{low}) + F[f]*(1-H^{1024}_{low}) + F[f]*H^1_{low}].$$

$$= F^{-1}[F[f]]$$

$$= F^{-1}[\sum_{\sigma=1}^{N} F[f]*(H^{2\sigma}_{low} - H^{\sigma}_{low})] + \overbrace{F^{-1}[F[f]*(1-H^{2N}_{low})]}^{\text{高周波の余り}} + \overbrace{F^{-1}[F[f]*H^1_{low}]}^{\text{ベースの低周波}}.$$

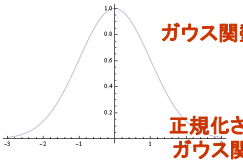
f :入力信号

$F[]$:変換　$F^{-1}[]$:逆変換　$*$:掛け算　H^{σ}_{low} :スケール σ のロウパスフィルタ.

バンド幅は任意でよいが、実用上は2倍のスケール幅をよく使う(Wavelet・多重解像度解析との関係).

Shin Yoshizawa: shin@riken.jp

ガウス関数



ガウス関数:

1次元:

$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} G_\sigma(x)$$

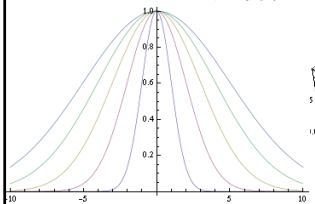
2次元:画像

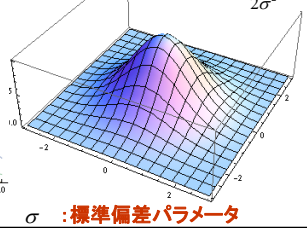
$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} G_\sigma(x,y)$$

正規化されたガウス関数:

$$G_\sigma(x) = \exp(-\frac{x^2}{2\sigma^2})$$

$$G_\sigma(x,y) = \exp(-\frac{x^2+y^2}{2\sigma^2})$$



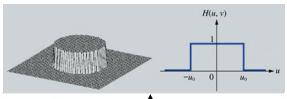


σ が小さい→尖った形状　σ :標準偏差パラメータ

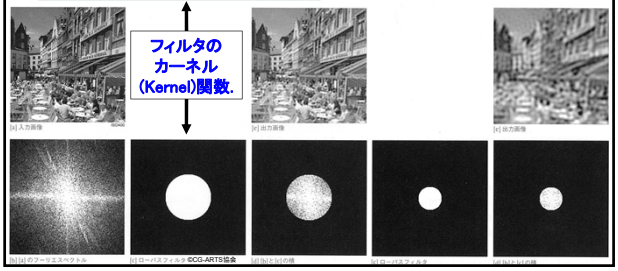
Shin Yoshizawa: shin@riken.jp

復習:ローパスフィルタ(Box関数)

✓ −u0から、u0までの低周波数成分だけ残す.



周波数の高い横方向の波(縦縞)を消す.



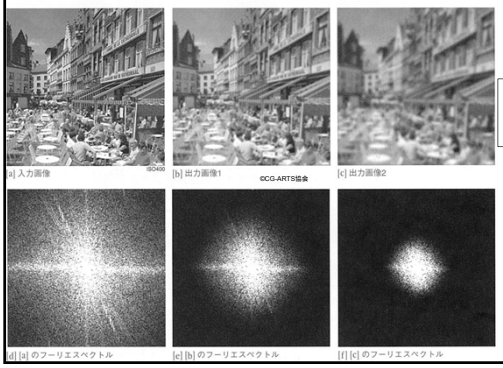
フィルタのカーネル(Kernel)関数.

Shin Yoshizawa: shin@riken.jp

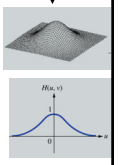
復習:ローパスフィルタ(ガウス関数)

✓ Gaussianフィルタとも呼ぶ.

画像の平均的な明るさが保持される.

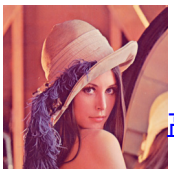


フィルタのカーネル(Kernel)関数.

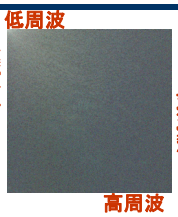


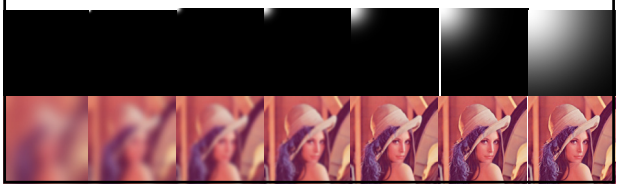
Shin Yoshizawa: shin@riken.jp

FDCTによるGaussianフィルタ



高速DCT





Shin Yoshizawa: shin@riken.jp

GaussianとBox関数によるロウパスフィルタ

Shin Yoshizawa: shin@riken.jp

DoGとバンドフィルタ

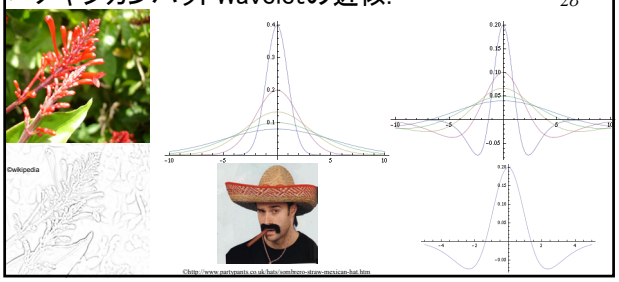
✓ DoG: Difference of Gaussian.

$$DoG_\sigma(x,y) = g_\sigma(x,y) - g_{2\sigma}(x,y).$$

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} G_\sigma(x,y)$$

$$G_\sigma(x,y) = \exp(-\frac{x^2+y^2}{2\sigma^2})$$

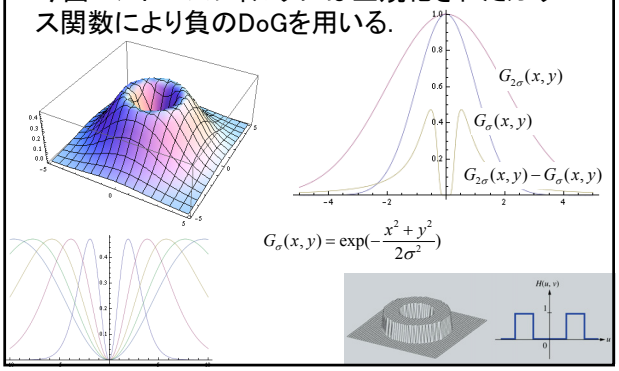
✓ メキシカンハットWaveletの近似.



Shin Yoshizawa: shin@riken.jp

DoGとバンドフィルタ2

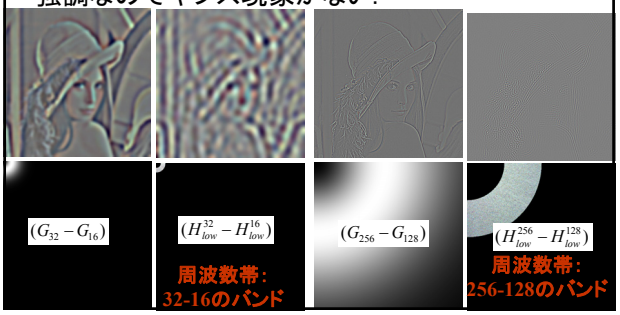
✓ 今回バンドパスフィルタには正規化されたガウス関数により負のDoGを用いる.



Shin Yoshizawa: shin@riken.jp

DoGとバンドフィルタ3

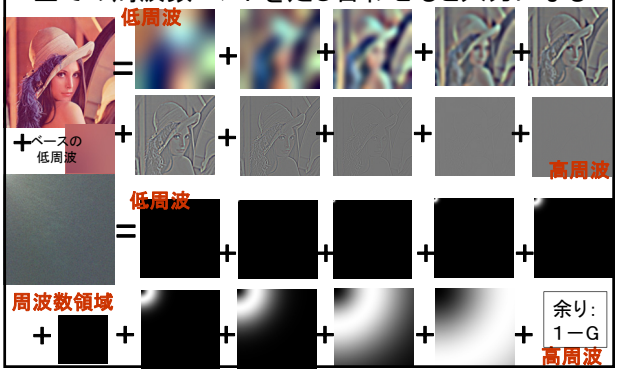
✓ ガウス関数を用いた場合の周波数分解は(理論的には)全ての周波数を使った帯域(バンド)強調なのでギプス現象がない.



Shin Yoshizawa: shin@riken.jp

DoG+FDCTによる周波数分解

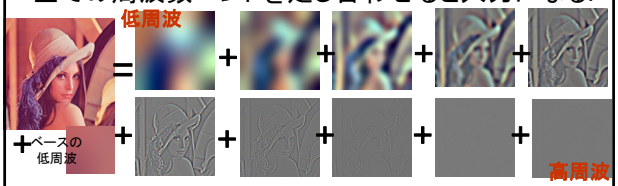
✓ 全ての周波数バンドを足し合わせると入力になる.



Shin Yoshizawa: shin@riken.jp

DoG+FDCTによる周波数分解2

✓ 全ての周波数バンドを足し合わせると入力になる.



$$f = F^{-1}[(F[f]*G_2 - F[f]*G_1) + (F[f]*G_4 - F[f]*G_2) + (F[f]*G_8 - F[f]*G_4) + \ldots + (F[f]*G_{512} - F[f]*G_{256}) + F[f]*(1-G_{512}) + F[f]*G_1].$$

$$= F^{-1}[F[f]]$$

$$= F^{-1}[\sum_{\sigma=1}^{N} F[f]*(G_{2\sigma} - G_\sigma)] + \overbrace{F^{-1}[F[f]*(1-G_{2N})]}^{\text{高周波の余り:Nが}\infty\text{ならゼロ}} + \overbrace{F^{-1}[F[f]*G_1]}^{\text{ベースの低周波}}.$$

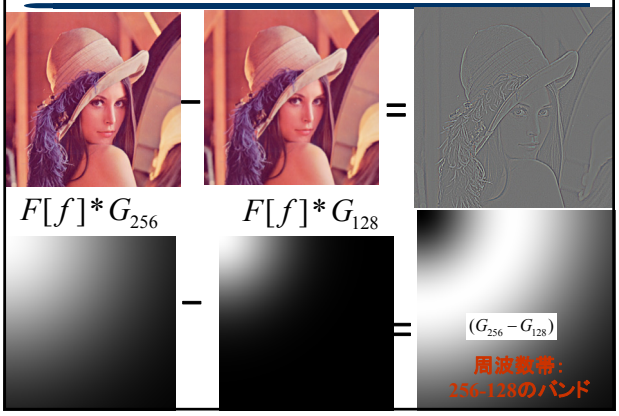
f :入力信号

G_σ :スケール σ の正規化ガウス関数.

$F[]$:変換　$F^{-1}[]$:逆変換　$*$:掛け算

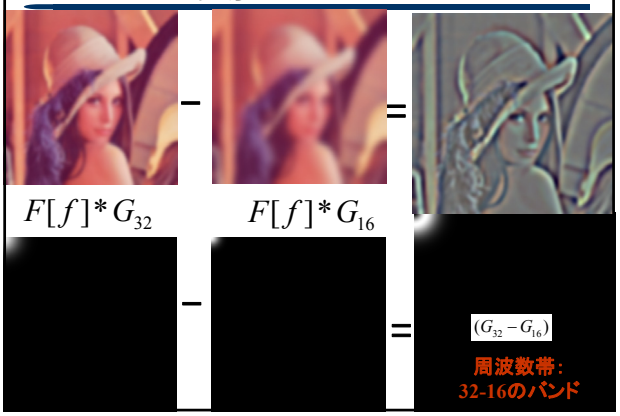
Shin Yoshizawa: shin@riken.jp

差分によるDoG



Shin Yoshizawa: shin@riken.jp

差分によるDoG2



Shin Yoshizawa: shin@riken.jp

平滑化と差分による周波数分解

✓ つまりGaussianフィルタと差分を繰り返し適用する事で周波数分解を近似出来る.

$$f = F^{-1}[\sum_{\sigma=1}^{N} F[f]*(G_{2\sigma}-G_{\sigma})] + F^{-1}[F[f]*(1-G_{2N})] + F^{-1}[F[f]*G_1].$$

f :入力信号

G_σ :スケール σ の正規化ガウス関数.

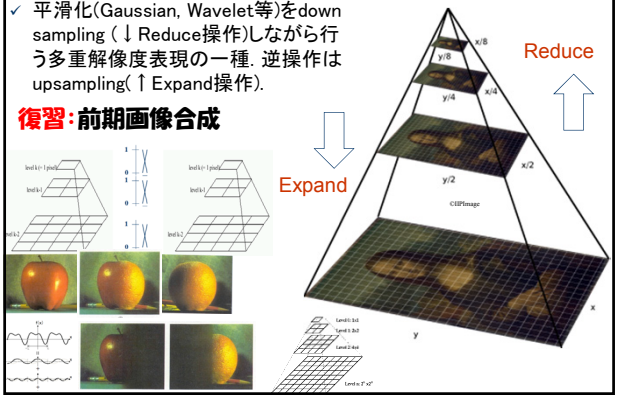
$F[]$:変換　$F^{-1}[]$:逆変換　$*$:掛け算

Shin Yoshizawa: shin@riken.jp

Pyramid表現

✓ 平滑化(Gaussian, Wavelet等)をdown sampling (↓Reduce操作)しながら行う多重解像度表現の一種. 逆操作はupsampling(↑Expand操作).

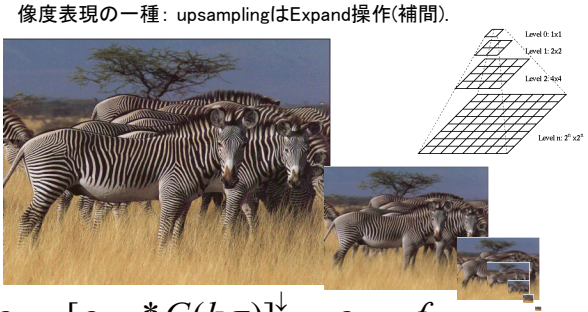
復習:前期画像合成



Shin Yoshizawa: shin@riken.jp

Gaussian Pyramid

✓ Gaussian平滑化をdown sampling (Reduce操作)しながら行う多重解像度表現の一種: upsamplingはExpand操作(補間).



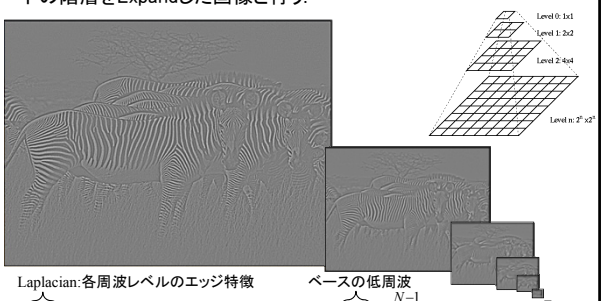
$$g_k = [g_{k-1} * G(k\sigma)]^{\downarrow}, \quad g_0 = f.$$

f :入力信号　$*$ 畳み込み:ここではGaussianフィルタと同義.

Shin Yoshizawa: shin@riken.jp

Laplacian Pyramid

✓ Difference of Gaussian (DoG)をGaussian Pyramidの各階層で一つ下の階層をExpandした画像と行う.

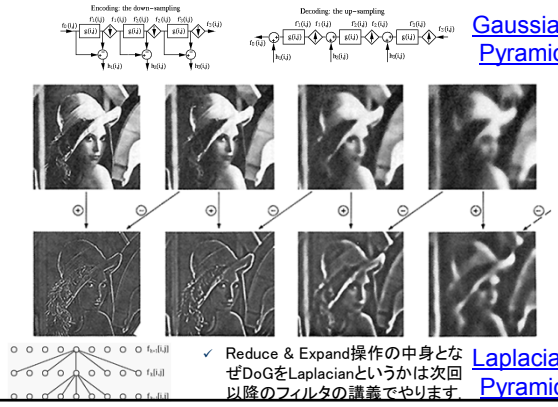


Laplacian:各周波レベルのエッジ特徴　ベースの低周波

$$L_k = g_k - [g_{k+1}]^{\uparrow} \qquad f = g_N + \sum_{i=0}^{N-1} L_i^{\uparrow}$$

Shin Yoshizawa: shin@riken.jp

Gaussian & Laplacian Pyramid



✓ Reduce & Expand操作の中身となぜDoGをLaplacianというかは次回以降のフィルタの講義でやります.

演習: FDCT

www.riken.jp/brict/Yoshizawa/Lectures/index.html
www.riken.jp/brict/Yoshizawa/Lectures/Lec16.pdf

高速離散コサイン変換

www.riken.jp/brict/Yoshizawa/Lectures/Ex09.zip

前々回の演習(BMPとPPMの相互変換)と前回の演習(四角→円)が分からなかった or 出来ていない or 欠席した人は、前回・前々回の演習から始める事！

演習:Ex09.zipの説明

- ✓ fftsg.h,fftsg2d.h,alloc.h:FFT用ヘッダーファイル.
- ✓ FDCT.h:高速離散コサイン変換用ヘッダーファイル
- ✓ testFDCT.cxx: Box関数のローパスフィルタを実行.
- ✓ testFrequency.cxx:周波数分解をBox関数で実行.
- ✓ testGauss_FDCT.cxx:Gaussianフィルタを実行.
- ✓ testGauss_Frequency.cxx:周波数分解をガウス関数で実行.
- ✓ まずは、makeでコンパイルして上記4つのプログラムを実行してみよう！

演習17-1:ローパスフィルタと周波数分解

- ✓ testFDCT.cxx:の実行:引数3

./testFDCT 入力bmp 出力名(.bmpなし) 周波数の閾値(int)
閾値＝4, 16, 64, 128で実行してみましょう.

- ✓ testFrequency.cxxの実行: 引数3

./testFrequency 入力bmp 出力名(.bmpなし) バンドの最大閾値(int)
閾値=1024で実行してみましょう.

- ✓ testGauss_FDCT.cxxの実行:引数3

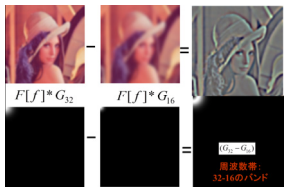
./testGauss_FDCT 入力bmp 出力名(.bmpなし) ガウス関数の標準偏差(double)
閾値=4.0, 16.0 64.0 128.0で実行してtestFDCTの結果と比べてみよう.

- ✓ testGauss_Frequency.cxx:の実行:引数3

./testGauss_Frequency 入力bmp 出力名(.bmpなし) バンドの最大値(double)
閾値=1024.0で実行してtestFrequencyの結果と比べてみよう.

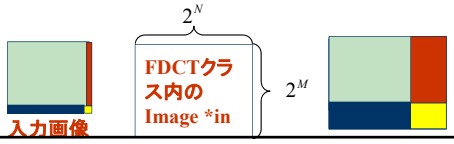
演習17-2:差分により周波数分解を作成！

- ✓ 2つの異なるパラメータでGaussianフィルタ適用した結果を差分する事でバンドパスフィルタの結果を出力するプログラムを作成してみよう！
- ✓ ヒント:
  testGauss_FDCT.cxxを改造する.
- ✓ 異なるパラメータ用に2種類のFDCTクラスとその出力画像クラスを作る→平滑化(変換→ガウス関数との積→逆変換)×2→差分画像を計算→正規化しBMP画像として出力.



演習:FDCTクラスの説明

- ✓ fftsg.h,fftsg2d.h,alloc.h:FFT用ヘッダーファイル.
- ✓ FDCT.h:高速離散コサイン変換用ヘッダーファイル
  - fftsg.h,fftsg2d.h,alloc.hを中で使ってFDCTクラスを定義:
  - 内部でImage *in;として周波数<->画像変換のためのデータを持つ.宣言の時点でコンストラクタのImageクラスoriginalの中身がFDCTクラス内のImage *inにコピーされる.
  - 注意点:クラス内部の画像サイズin->sx,in->syは2の乗数で元の画像サイズが2の乗数でない場合はoriginal->sx,original->syに最も近い2の乗数になり、余りはoriginalの端のデータが入る.



演習:FDCTクラスの説明2

- FDCTクラスのメソッドは、
- void DCT(): 高速離散コサイン変換を実行:Image *inの中身がコサイン変換後の周波数成分.
- void InverseDCT(Image *out): 高速離散コサイン逆変換を実行:Image *inの中身が逆変換後の画像で、outにoriginalと同じ画像サイズを入れる事でinの中身を出力としてコピー.
- 注意点:InverseDCT()を実行するにはDCT()を先に実行する事.
- void Normalize(Image *out): outの輝度値を0-255に正規化.
- void toSpectrum(Image *out): パワースペクトル画像log(1+F*F)をoutへ保存.
- void CopyTo(Image *din): inの中身をdinにコピー.
- void CopyFrom(Image *din): dinの中身をinにコピー.

Shin Yoshizawa: shin@riken.jp

演習：FDCTクラスの説明3

- FDCTクラスの使い方は、testFDCT.cxxを参照.

1. 宣言+メモリ確保：FDCT *fft_R = new Image(R); ただしRはImageクラスで中身が入っている事(宣言+メモリ確保+readBMP()).
2. 変換:fft_R->DCT();
3. 周波数の操作：fft_R->in->img[i][j] = …
4. 逆変換：fft_R->InverseDCT(out); ただし、outはRと同じ大きさでメモリが確保されたImageクラス.
5. メモリ開放：delete fft_R;

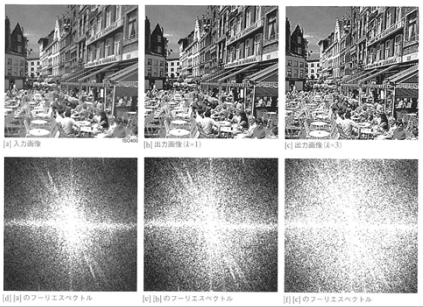
Shin Yoshizawa: shin@riken.jp

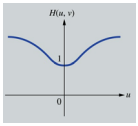
復習：高域強調フィルタ

✓ ハイパスフィルタから作る事が出来る.

$$H_{h-emph}(u,v) = 1 + kH_{high}(u,v)$$

エッジ強調！





Shin Yoshizawa: shin@riken.jp

演習17-3：高域強調フィルタ

✓ 高域強調フィルタを作ってみよう！

✓ ヒント：

1. Gaussianフィルタ(testGauss_FDCT.cxx)を改造する.
2. フィルタのカーネルは周波数領域で

$$H_{h-emph}(u,v) = 1 + kH_{high}(u,v) = 1 + k(1 - H_{low}(u,v))$$

3. ここでロウパスフィルタのカーネル $H_{low}(u,v)$ はプログラム内ではgauss->img[i][j]というガウス関数の画像を用いる.
4. フィルタ後は正規化なし(輝度値が0以下は0、255以上は255).
5. ロウパスフィルタのガウス関数のパラメータ(標準偏差)と上の強調フィルタのパラメータk二つのパラメータをプログラムの引数で与える.
6. testGauss_FDCT.cxxは $H_{low}(u,v)$ だけのフィルタ.