2014-02
20

画像処理を始めよう　～エッジ2～

画像処理

前回に続いてエッジの話をしていきましょう。

エッジのとらえ方には

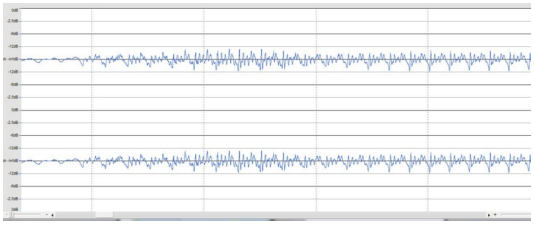
1. 微分値が高い
2. 高周波
3. ゴリ

の３種類があるといいました。今回は高周波のお話です。

画像は２次元の信号

信号理論を学んだことがない人にはハイパスフィルタと言われてもぴんとこないかもしれません。そもそも、画像が信号というのも、ぴんとこないのかもしれません。ていうか、信号って青黄赤以外に何かあったっけとか思われるかもしれません。残念ながら交通の信号機のことではありません。

ここでの信号は情報の変化を伝える波です。中学高校の理科でオシロスコープで電流の波形とか、見たことありませんか？他には、下の図のような音の波形信号なども身近な例ですね。



（キャプチャ画面:Sound Engine Free）

電流や音の場合、信号は時間で変化する電圧、音圧の波f(t)です。画像の場合、空間で情報が変化する色の波f(x,y)です。動画だとf(x,y,t)になります。時間で変化する信号を扱ってた私の知人にとっては、最初取っつきにくかったようです。なお、「色情報をどのように数値化するのか？」という事も重要な問題ですが、この記事ではそこは特に問題にしないでおきます。グレーだけ扱うとか、RGBで表現するのが一般的です。

高周波、低周波についても軽く説明しておきます。高周波というのは、波の繰り返し単位が短いことを言います。逆に低周波というのは波の繰り返し単位が長いことを言います。

プロフィール

クソニート
なんかJavaとか、JavaFXとかについて書いてるっぽい。もう画像処理を書くことはないんじゃないかな。

読者になる 12

@nodamushiさんをフォロー

検索

記事を検索

最新記事

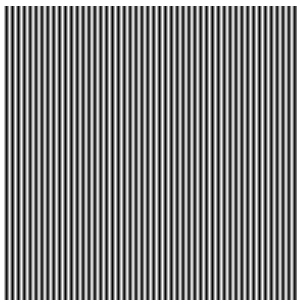
Eclipseプラグイン開発: 拡張ポイントの定義
Eclipseプラグイン開発: 非UIプラグインのテスト
Eclipseプラグイン開発: 非同期実行
Eclipseプラグイン開発: バンドルリソース関連
Eclipseプラグイン開発: リソースパス関連

月別アーカイブ

▶ 2017 (9)
▶ 2016 (11)
▶ 2015 (17)
▼ 2014 (11)
2014 / 12 (2)
2014 / 10 (3)
2014 / 9 (1)
2014 / 7 (1)
2014 / 2 (1)
2014 / 1 (3)
▶ 2013 (27)
▶ 2012 (8)
▶ 2011 (46)
▶ 2010 (37)
▶ 2009 (16)

カテゴリー

C/C++ (1)
CoffeeScript (5)
eclipse (5)
Eclipse CDT (3)
Eclipseプラグイン開発 (7)
Emacs (1)



（高周波な画像）

（低周波な画像）

上の画像は非常に単調な、一つの波しか持たない極端な例です。一般の信号というのは、様々な高周波や低周波の足し合わせからなります。この中から高周波成分だけ取ってくれば（＝ハイパスフィルタをかければ）、エッジ画像ができあがるというわけです。上の画像見ても、高周波な画像はエッジっぽいけど、低周波な画像はぼけていてエッジっぽくないよね。

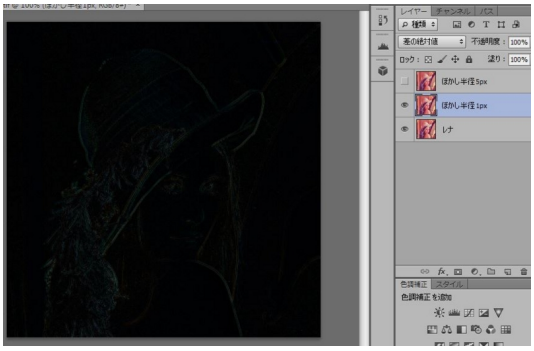
高周波成分を取り出す

ん～、納得できないけど、画像は信号だって事にしておいてやろう。しておいてやってください。話進まないので。

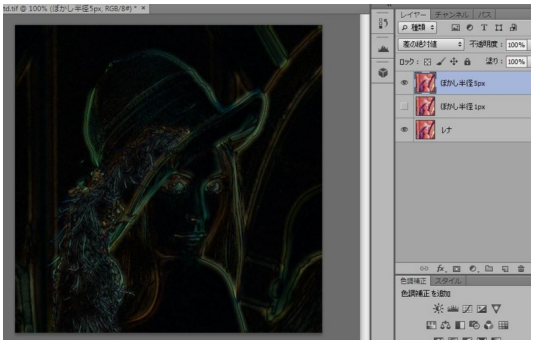
で、どうやって高周波成分取り出すの？

まぁ～、最も単純なのはぼかしを使うことですね。「元の画像」-「ぼかした画像」＝「エッジ画像」になるのです。え？納得できない？え～っとですね、ぼかすという行為は**低周波**な画像を作成することに他なりません。つまり、「元の画像」-「ぼかした画像」＝「エッジ画像」というのは「全部の周波」-「低周波」＝「高周波」といってるのと同じ事なんです。

実際にPhotoShopでガウスぼかしした画像を、元画像から引いてみました。



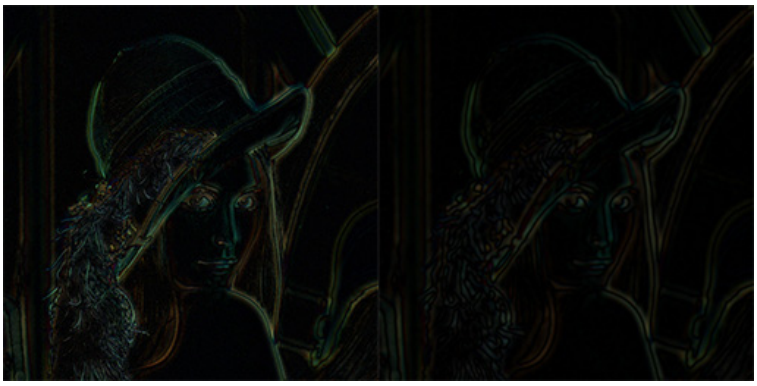
（ぼかし半径1px）



（ぼかし半径5px）

見事にエッジ画像が得られていますね。なお、ぼかし半径が小さい方がより高周波なエッジ画像を得られています。ぼかし半径が小さいほどぼかし画像にはかなりの高周波が残ります。よって、全部の周波数からぼかした画像をひくと、ぼかし画像に残らなかった非常に高い高周波だけが結果として出てきます。よって、ぼかし半径が小さいほど、差分を取ったときに高周波成分を取り出せるのです。

ぼかし半径の違いを利用して、ぼかし画像から、より大きなぼかし画像を引くことで狙った周波数のエッジだけを取り出すことが出来ます。



左は原画から5pxぼかした画像を引いた物、右は2.5pxぼかした画像から5pxぼあかした画像を引いた物です。髪の毛の高周波や、ノイズが少なくなってるのが分かると思います。これは、いわゆるDoG(Difference of Gaussian)で、LoG(Laplacian of Gaussian)の近似であるということは、SIFT特徴量の記事で解説しました。

さて、なんとなく、原理は理解していただけたかと思います。

前回のコメントで、文系の方でも読む奇怪な人もいると知ったので、基礎の基礎までは極力数式を避けておきました。が、ここまでの説明で回避してきた畳み込みとフーリエ変換を使って、もう一回同じ話をしておきます。説明を数学とか宇宙語に頼るのは、説明力が低いことに他ならないので悔しい限りですが。

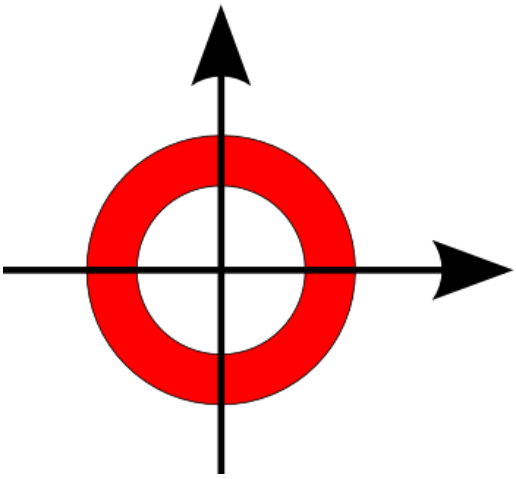
ハイパスフィルタ

がっつり低周波成分をカットしたいなら、周波数空間で「1 - シリンダ(円形)関数」を掛け合わせれば良い。が、シリンダ関数を逆フーリエ変換したらベッセル関数なんて負数は出るわ振動するわ収束しないわでややこしい。シリンダ関数の代わりに矩形関数を使ったとしても、出てくるsinc関数が厄介。なので、シンプルにガウス関数を用いるのが一般的です。（ガウスも収束はしないが…）

ガウス関数をフーリエ変換すると、以下の式のように、標準偏差が実空間での標準偏差の逆数になります。したがって、実空間で「1-ガウス関数」と計算されるフィルタで残る周波数は、ガウス関数の標準偏差が小さいほど、高周波だけが残るようになります。逆に、標準偏差が大きいと、低周波まで残るようになります。

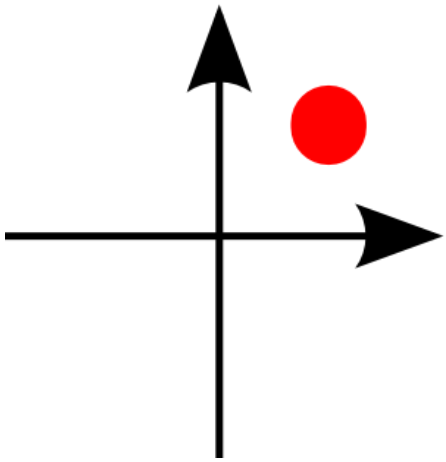
$$F(\exp(-\frac{x^2}{\sigma^2})) = \frac{\sqrt{\pi}}{\sigma}\exp(-\frac{\sigma^2\omega^2}{4})$$

さて、実際にはガウス関数は滑らかな関数ですが、ここでは簡単のため、ガウス関数＝シリンダ関数とみなしてしまいましょう。すると、「ガウス関数（σ1) - ガウス関数（σ2）」（σ1<σ2）というフィルタを畳み込むことで残る周波数帯域は以下の図のような領域になります。軸は、x方向の周波数およびy方向の周波数です。



ガボールフィルタ

先の「ガウス関数（σ1) - ガウス関数（σ2）」フィルタでは、ｘ軸、y軸関係なくある周波数帯域の周波数をすべてとってくることになります。そうすると、一つの興味が自然とわいてきます。特定の領域の周波数だけ残すことはできないのだろうか？たとえば、下図の赤い部分だけを取ってくることはできないだろうか？



答えは単に赤いところだけ値が1で、そうでないところは0という関数を周波数空間で掛け算すればいい。0,1だけの関数はフーリエ変換で非常に扱いにくいから、赤い部分はいつも通りガウス関数で代用すればいいだろう。単純には

$$\exp\{-(\omega_x - a)^2 - (\omega_y - b)^2\}$$

という形状でこの関数を表現できるだろう。しかし、この形ではフーリエ変換で扱いにくい。そこで、この赤い領域の中心がx軸に乗るように座標をθ回転させる。その座標系では

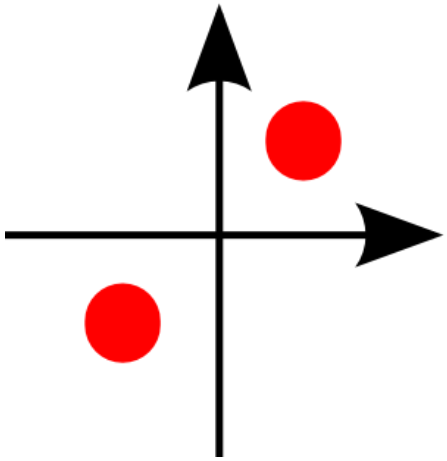
$$\exp\{-(\omega'_x - k)^2 - \omega'^2_y\}$$

となる。これを逆フーリエ変換すると

$$\cos(2\pi kx)\exp + i\sin(2\pi kx)\exp$$

となる。（ガウス関数の中身は省略しました。）

虚数部はフィルタとして扱いにくいので、無視をします。虚数部を無視すると、周波数空間では下図の様に、原点に対して反対方向に同じ形の領域が出てきます。



　あとはガウスフィルタを円形ではなく、楕円形にしたり、位相を考慮するなどして、最終的に次のフィルタが得られます。

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right)$$

$$x' = \cos\theta x - \sin\theta y$$

$$y' = \sin\theta x + \cos\theta y$$

　これがいわゆるガボールフィルタです。さまざまなλ、θ、Ψ、γ、σの値のガボールフィルタの集合をガボールフィルタバンクといいます。ガボールフィルタはエッジ検出のためのフィルタではありませんが、画像処理において、エッジとの少なからぬ関係性があります。

nodamushi 3年前

コメントを書く

« -Π≦θ<Πに納める CoffeeScript v1.7 »