2017 年度　卒業研究

Lagrange Lanczos(n)法による拡大画像について

岡山理科大学　総合情報学部　情報科学科

澤見研究室

I13I012 江田　匠

I13I013 大嶋　圭悟

I14I067 錦織　慶亮

目次

1. はじめに.....1

2. 原画像について.....1

3. 画像補間法.....1

3.1 ニアレストネイバー法.....2

3.2 バイリニア法.....2

3.3 バイキュービック法.....3

3.4 sinc Lanczos(n)法.....4

3.5 Lagrange 補間法.....6

4. 補間法ごとの出力画像と処理速度.....7

5. Lagrange Lanczos(n)法.....9

6. 補間の手法について.....11

7.1 ブロック単位補間とポイント単位補間の比較.....12

7.2 ルンゲ現象とギップス現象について.....14

8. 補間法ごとの比較.....14

9.1 原画像の画素情報量が少ない場合の比較.....15

9.2 原画像の画素情報量が少ない場合の比較（カラー版）.....18

10. まとめ.....19

参考文献.....19

1 はじめに

近年ではSNSなどで写真や画像を見たり用いる機会が増えており,さらにその画像を加工するソフトも多く出ている.本研究では画像を加工する中から画像の拡大について焦点をあて,既存のソフトで使われている画像の拡大方法よりに原画像に近い画質で画像を拡大できる方法はないのだろうかと考え,この研究に取り組むことにした.今回は既存の方法からsinc Lanczos(n)法を参考にしてより優れた補間法になると考えられるLagrangeLanczos(n)法を提案することにした.

2 原画像について

原画像としてラスタ画像を使用した.ラスタ画像の主な形式にはPNG(Portable Network Graphics),JPEG(Joint PhotoGraphic Experts Group),BMP(Bitmap)があり,一般的なディジタルカメラやスマートホンではJPEGが用いられている.ラスタ画像は別名ビットマップ画像ともいわれ. ディジタルカメラやインターネットで用いる画像として用いられており,Photoshopなどの編集ソフトでも用いられている.一般に3原色それぞれを8ビットで表しドットピクセルを2次元配列にして構成された画像のことで,一般的なディスプレイを用いてディジタル画像の色や濃淡の変化を自然に見せる場合に適している.ここでは主にBMPとJPEGを用いている.

図1ラスタ画像の見本

3 画像補間法

画像補間法は画像の拡大,縮小,回転,変形などを行うときに用いられており,補間の際にピクセルとその周辺のピクセルの間に新たなピクセルを生成するよう計算し利用するための技術である.

主な補間法にはニアレストネイバー法,バイリニア法,バイキュービック法,sinc Lanczos(n)法があるが,ここでは主としてsinc Lanzos(n)法,Lagrange Lanczos(n)法について研究していく.

3.1　ニアレストネイバー法

補間に 0 次多項式を用いた方法で原画像の画素値をそのまま用いており，補間するピクセルに一番近いピクセルを参照していることから最隣接法とも呼ばれている．拡大時の計算速度は後述するバイリニア法やバイキュービック法よりも早いが，拡大した画像の画質が荒くなりジャギーが発生するデメリットがあるため精度はやや低いといえる．ジャギーとは偽象とも呼ばれ，画像を拡大した際にみられる階段状のギザギザのことである．画像の補間方法は，緑で示した座標ピクセルを補間する場合その座標から一番近い位置にあるピクセル（図 2 の赤いピクセル）一つを参照して補間している．

補間したい画素の座標（Px, Py）における画素値を g とした場合の計算式は以下の通りとなる．補間される緑のピクセルの座標（Px, Py）の値は処理を進めていくにつれて変化する．これは後述する補間法でも同じである．

$$0 \leq Px \leq 1, \qquad 0 \leq Py \leq 1$$
$$g(Px, Py) = f(x + Px, y + Py)$$

・・・・・・(式 1)

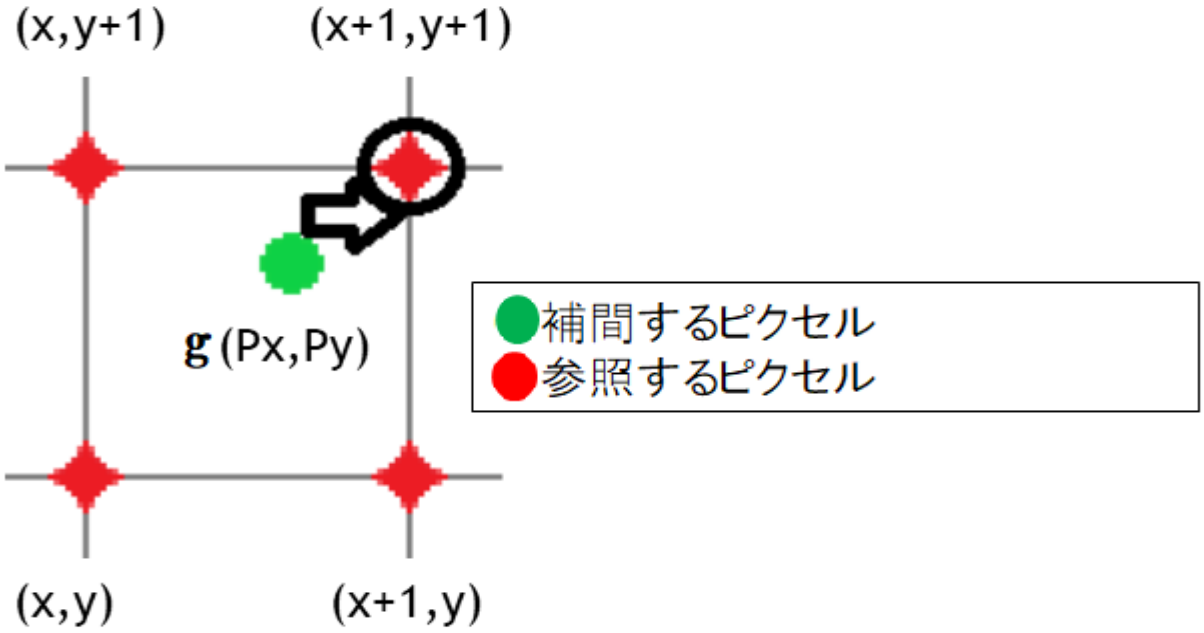


図 2　ニアレストネイバー法の参照ピクセル

3.2 バイリニア法

補間に 1 次多項式を用いた方法で，補間されるピクセルに隣接している四つのピクセルを参照している．前述したニアレストネイバー法よりも多くのピクセルを参照し重み付け計算により必要とする位置で補間計算をし，ピクセル値を生成しているため拡大画像にジャギーが目立たなくなり，よりきれいに見える．しかし，本来は存在していなかった位置にピクセル値を生成するため，境界部分がぼやけて表示されることがあり，ニアレストネイバー法より優れているもののやはりあまり画質はよくない．補間方法は，緑の座標ピクセルを補間する場合その周囲にある四つの赤いピクセルを参照して補間計算している（図 3）．

補間は図のように縦横に一次元の式を用いて二次元格子上で行う(図 3). 補間したい画素の座標(Px, Py)の画素値を g として計算した場合の式は以下の通りとなる.

$$0 \leq Px \leq 1,\quad 0 \leq Py \leq 1$$

$$f(Px) = f(x) \cdot g_0(Px) + f(x+1) \cdot g_1(Px) + f(x+2) \cdot g_2(Px) + f(x+3) \cdot g_3(Px)$$

・・・・・・(式 2)

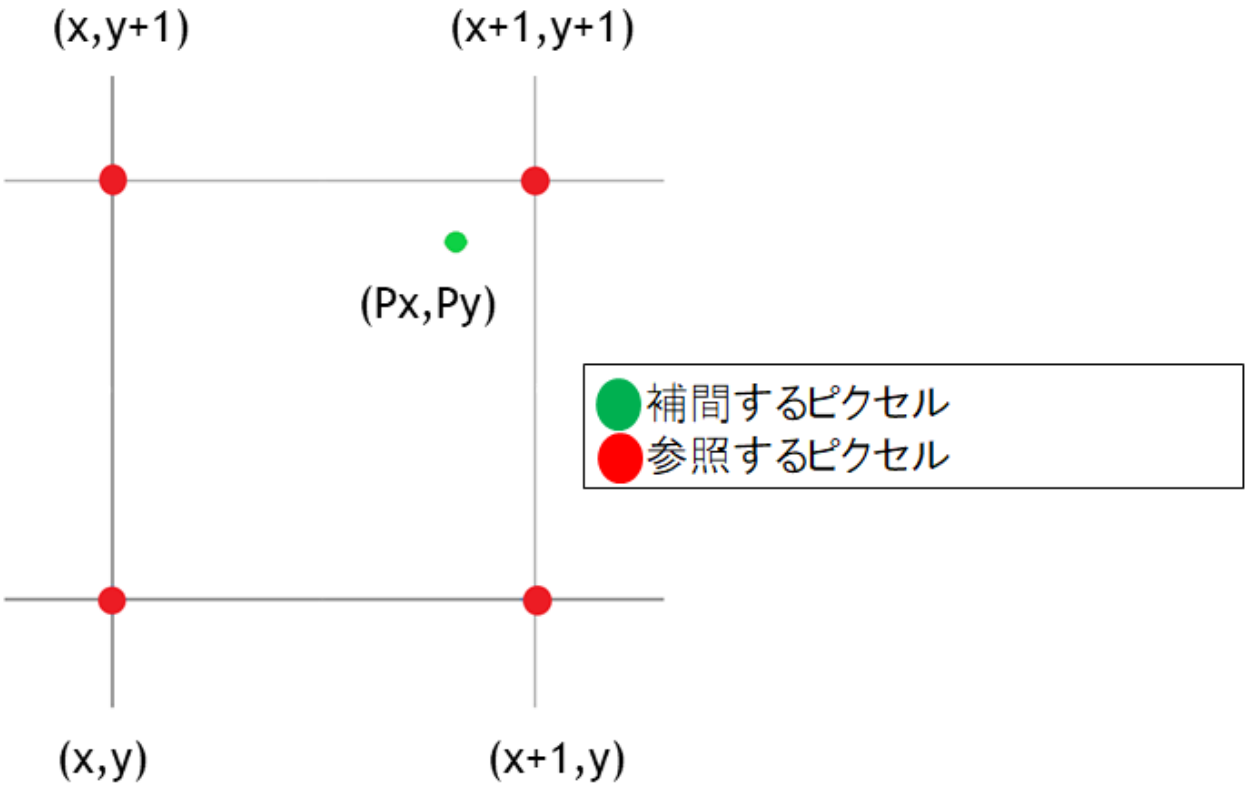


図 3　バイリニア法の参照ピクセル

3.3 バイキュービック法

補間に 3 次多項式を用いた方法で, 隣接しているピクセルだけではなくさらに周辺の 4*4 の 16 ピクセルを参照している. ニアレストネイバー法, バイリニア法, 後述する Lanczos(n)法と比べても, もとの画像に含まれている情報の損失が最も少ないと考えられており, 自然な画質の画像が得られる画像補間法であることが知られている. 計算量が多いため処理速度はさらに遅くなるが, 拡大時の画質の粗さが減少し綺麗に拡大することができる.

補間したい画素の座標(Px, Py)の画素値を g として計算した場合の式は以下の通りとなる.

$$g_{0,3}(x) = \frac{(x_1 - x)(x_2 - x)(x_3 - x)}{(x_1 - x_0)(x_2 - x_0)(x_3 - x_0)}$$

$$g_{1,3}(x) = \frac{(x_0 - x)(x_2 - x)(x_3 - x)}{(x_0 - x_1)(x_2 - x_1)(x_3 - x_1)}$$

$$l_{2,3}(x) = \frac{(x_0 - x)(x_1 - x)(x_3 - x)}{(x_0 - x_2)(x_1 - x_2)(x_3 - x_2)}$$

$$l_{3,3}(x) = \frac{(x_0 - x)(x_1 - x)(x_3 - x)}{(x_0 - x_3)(x_1 - x_3)(x_2 - x_3)}$$

・・・・・・(式 3)

また h(t), x1, …, x4, y1, …, y4 は以下により与えられる.

$$h(x_0) = l_{0,3}(x)\ , h(y_0) = l_{0,3}(y)$$

$$h(x_1) = l_{1,3}(x)\ , h(y_1) = l_{1,3}(y)$$

$$h(x_2) = l_{2,3}(x)\ , h(y_2) = l_{2,3}(y)$$

$$h(x_3) = l_{3,3}(x)\ , h(y_3) = l_{3,3}(y)$$

・・・・・・(式 4)

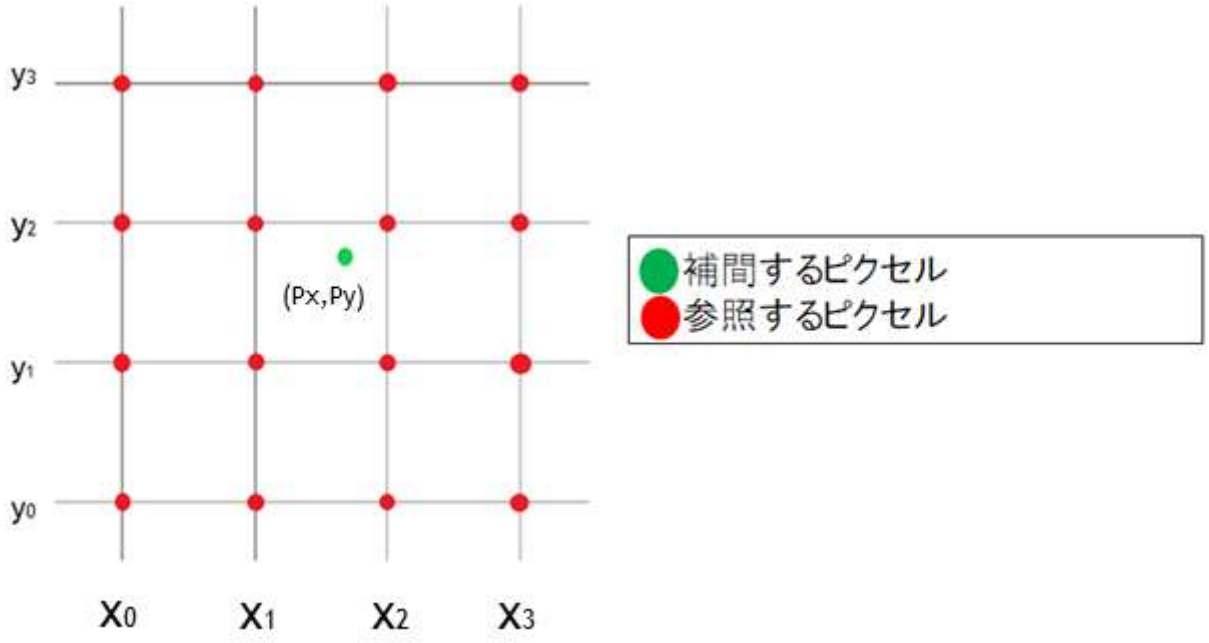


図 4　バイキュービック法の参照ピクセル

3.4　sinc Lanczos(n)法

ローパスフィルタのインパルス応答としても知られている sinc 関数は, フーリエ変換を用いた補間方法に使われており, 正弦関数をその変数で除して得られる初等関数のことをいう. 画像の拡大をする際に画質が劣化してしまう原因はいくつかあるが, そのひとつに補間処理の過程で起こる誤差振動がある. この誤差振動が起こることによって画質は劣化し, その振動が大きいほど画質の劣化も大きくなる. sinc 関数はグラフで見ると中心点から遠ざかるにつれて振動の幅が小さく緩やかになってい

くが,振動がなくなることはない(図 5).これがギップス現象とよばれる画像劣化の原因となっている.sinc Lanczos(n)法は sinc 関数に重みづけ関数 Lanczos を組み合わせて補間誤差の振動を極力減らした sinc Lanczos(n)関数を用いてピクセルを補間計算し画像を拡大する方法である.sinc 関数のままで使う場合と比べても誤差振動は少なくなっておりギップス現象の影響は軽減されている.Lanczos(n)法はピクセル数を 2n*2n 個単位のブロックで補間するため,ニアレストネイバー法などと比べると計算量が多くなることから処理速度はかなり遅くなる.しかし次数 n を大きな数値に変更することで拡大時の画像の粗さが減少し,画像を高画質で拡大することができる. sinc Lanczos(n)法による拡大は主に n=3 が用いられている.n の値が大きくなりすぎると誤差も大きくなっていくため,かえって画質が低下してしまう.今回のように n=3 の場合参照するピクセル数は(2*3)*(2*3)=36 となり,周囲にある 36 の赤いピクセルを参照して補間計算することになる.(図 6).

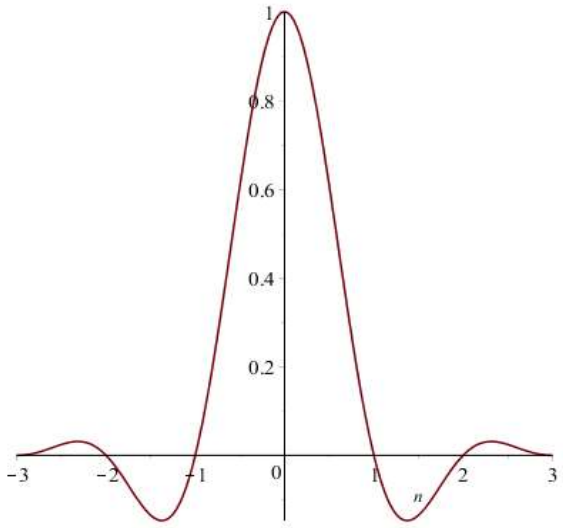


図 5 sinc Lanczos(3)法の関数

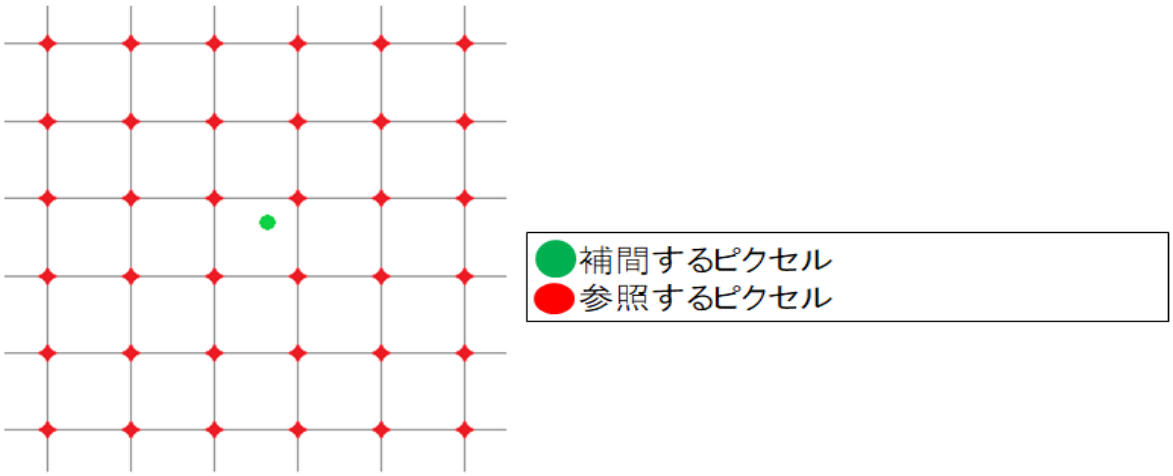


図 6 sinc Lanczos(3)法の参照ピクセル

3.5 Lagrange 補間法

Lagrange 多項式を用いた補間法であり Lagrange の公式を用いることで多項式の次数 n を容易に変更することができる. 本来であれば画像を拡大する際に偽象が発生するが, Lagrange 補間法の場合は次数をより高次の多項式にしていくことで偽象が目立たない画像を出力することができる. 以下に公式と 3 次の Lagrange 補間法の関数を示す. 赤い色の関数が Lagrange30, x=0 で値が 0 となっている青い関数が Lagrange31, 緑の関数が Lagrange32, x=1 で値が 0 となっている青い関数が Lagrange33 となっている(図 7.). 3 次の Lagrange 補間を画像に適用したものはバイキュービック法となる.

$$\mathrm{fn}(x) = \sum_{k=0}^{n} f(x_k) L_{k,n}(x)$$

・・・・・・(式 5)

$$L_{k,n}(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

・・・・・・(式 6)

$$\mathrm{Lagrange3,0} = \frac{(1-x)(2-x)(3-x)}{(1-0)(2-0)(3-0)}$$

$$\mathrm{Lagrange3,1} = \frac{(0-x)(2-x)(3-x)}{(0-1)(2-1)(3-1)}$$

$$\mathrm{Lagrange3,2} = \frac{(0-x)(1-x)(3-x)}{(0-2)(1-2)(3-2)}$$

$$\mathrm{Lagrange3,3} = \frac{(0-x)(1-x)(2-x)}{(0-3)(1-3)(2-3)}$$

・・・・・・(式 7)

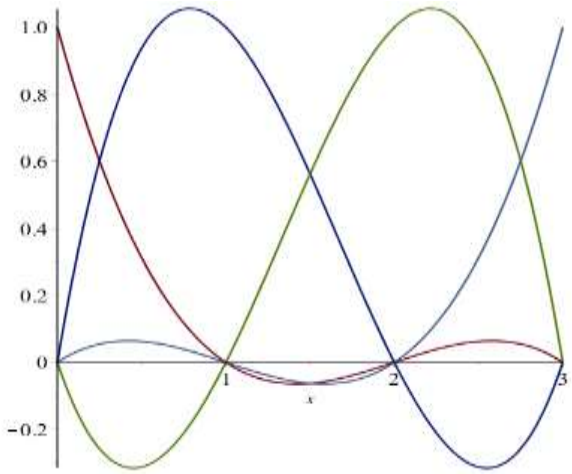


図 7　3 次 Lagrange 補間法の関数

これが 4 次 Lagrange 補間法になると 3 次 Lagrange 補間法に対してグラフの関数がひとつ増え, 数

式も以下のようになる.原点から始まる赤い関数が Lagrange44 である.

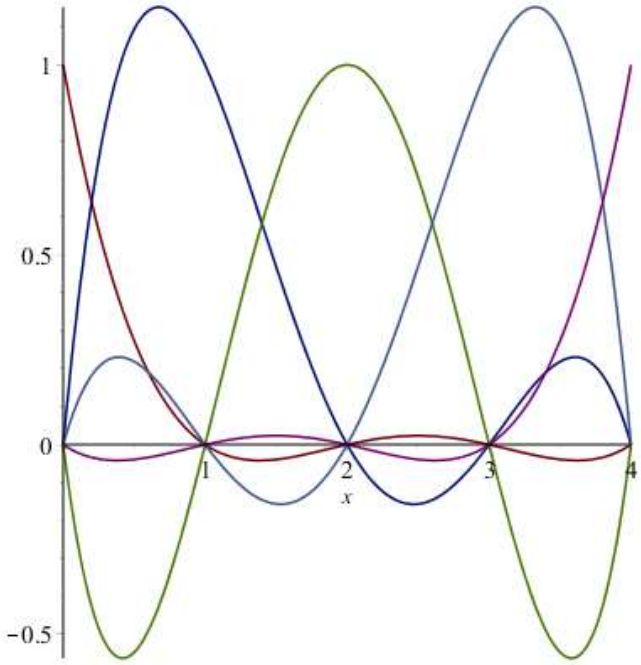


図 8 4 次 Lagrange 補間法の関数

$$\text{Lagrange40} = \frac{(1-\text{x})(2-\text{x})(3-\text{x})(4-\text{x})}{(1-0)(2-0)(3-0)(4-0)}$$

$$\text{Lagrange41} = \frac{(0-\text{x})(2-\text{x})(3-\text{x})(4-\text{x})}{(0-1)(2-1)(3-1)(4-1)}$$

$$\text{Lagrange42} = \frac{(0-\text{x})(1-\text{x})(3-\text{x})(4-\text{x})}{(0-2)(1-2)(3-2)(4-2)}$$

$$\text{Lagrange43} = \frac{(0-\text{x})(1-\text{x})(2-\text{x})(4-\text{x})}{(0-3)(1-3)(2-3)(4-3)}$$

$$\text{Lagrange44} = \frac{(0-\text{x})(1-\text{x})(2-\text{x})(3-\text{x})}{(0-4)(1-4)(2-4)(3-4)}$$

・・・・・・(式 8)

4　補間法ごとの出力画像と処理速度

ラスタ画像をニアレストネイバー法,バイリニア法,バイキュービック法,sincLanczos(n)法を用いて拡大してみる. 画像にある丸い対象物の輪郭の曲線部分は画像の劣化が分かりやすくなるため,赤枠で囲まれた目の部分を拡大する.前述したとおり参照するピクセル数が少ないニアレストネイバー法は画像が粗く,バイリニア法やバイキュービック法などと参照するピクセル数が多くなるにつれて画像が綺麗になり画質が向上しているのがわかる.

図 9 拡大前の画像

図 10 ニアレストネイバー法

図 11 バイリニア法

図 12 バイキュービック法

図 13 sinc Lanczos(4)法

図 14　4 次 Lagrange 法

画像の拡大時の処理速度は補間法によって違いがある.ニアレストネイバー法やバイリニア法は四則演算の回数が少なく,sinc Lanczos(n)法や Lagrange(n)法は参照する範囲が 2n*2n と n*n になることから四則演算の回数が多くなるためである.四則演算の回数が多いほど拡大時の処理時間は長くなるが,参照するピクセル数が多く計算量が多くなるほどより滑らかな画質となる.

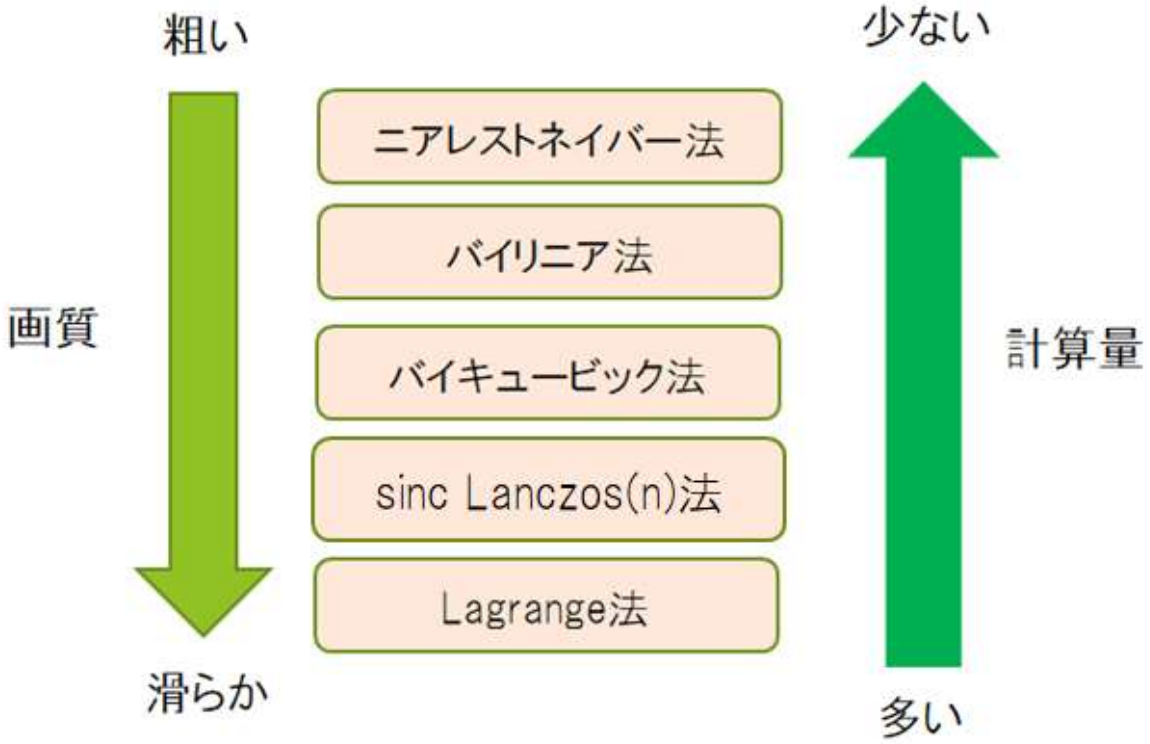


図 15 補間法ごとの計算量と画質の関係

5 Lagrange Lanczos(n)法

Lagrange 補間法で使用した n 次 Lagrange 多項式と sinc Lanczos(n)法で用いた重みづけ関数 Lanczos を組み合わせた画像の補間法を Lagrange Lanczos(n)法と呼ぶことにする. sinc Lanczos(n)法と Lagrange 補間法は主観評価でもあまり差は出なかったことから, 上記のふたつの補間法を組み合わせることで既存の補間法の画質を超えることができるのではと期待している.
sincLanczos(n)法と同じように, 次数 n=3 程度であれば偽象によって画質が大きく劣化するということは少ない. しかし次数 n が大きい多項式を用いる場合では, ブロック単位で処理すると画素ブロックの端のほうの計算において大きな誤差振動が発生する場合があり, その結果画像が粗くなって画質が低下してしまうことがある. 以下に n=4 とした場合の関数を示す. 前述した Lagrange 補間法に Lanczos を重みづけしているため Lagranege40L と表記する. 始めに山なりになっている赤い関数が Lagrange40L. x=3 で値が 0 となっている青い関数が Lagrange41L. 緑の関数が Lagrange42L. x=1 で値が 0 となっている青い関数が Lagrange43L. 残った赤い関数が Lagrange44L である. 前述した Lagrange 補間法の式に Lanczos 法の重みづけのための式 sincF を乗算するようにした式をソースコードに組み込むことで重みづけを行う(図 17). これらを用いて作成した Lagrange Lanczos(4)法による拡大画像はこれまでのものと大きな変化は見られなかった(図 18).

$$sincF = \frac{2\sin(2\left(\frac{1}{4}x - \frac{1}{2}\right)\pi)}{\pi(x-2)}$$

$$\text{Lagrange40L} = \frac{(x-1)(x-2)(x-3)(x-4)}{(0-1)(0-2)(0-3)(0-4)}sincF$$

$$\text{Lagrange41L} = \frac{(x-0)(x-2)(x-3)(x-4)}{(1-0)(1-2)(1-3)(1-4)} sincF$$

$$\text{Lagrange42L} = \frac{(x-0)(x-1)(x-3)(x-4)}{(2-0)(2-1)(2-3)(2-4)} sincF$$

$$\text{Lagrange43L} = \frac{(x-0)(x-1)(x-2)(x-4)}{(3-0)(3-1)(3-2)(3-4)} sincF$$

$$\text{Lagrange44L} = \frac{(x-0)(x-1)(x-2)(x-3)}{(4-0)(4-1)(4-2)(4-3)} sincF$$

・・・・・・(式 9)

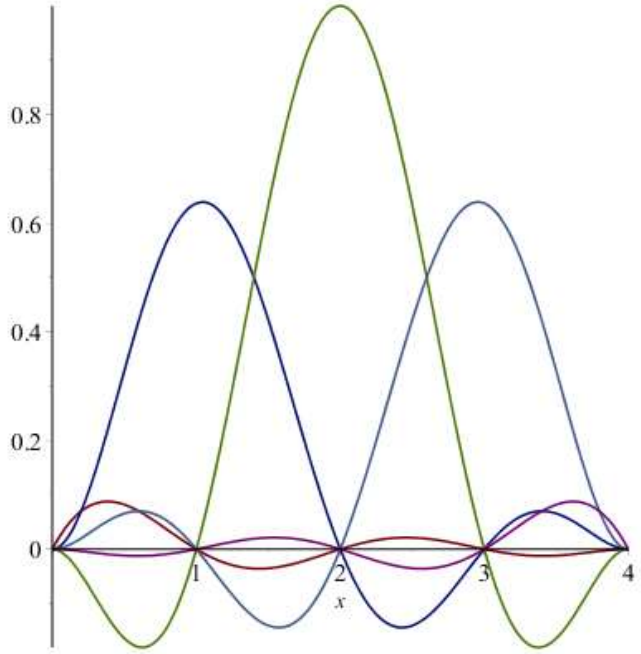


図 16 Lagrange Lanczos(4)の関数

```
for (row = 0; row<NN - 1; row++) {                              //カーネルの計算
    for (col = 0; col<LagN; col++) {
        LGnk = 1;
        for (n = 0; n<LagN; n++) {
            if (n != col) { LGnk = LGnk*(LagN / 2 - 1 + (row + 1.) / NN - n) / (col - n); }
            LG[row][col] = LGnk;
        sin( dblPi*( (row+1.)/NN -col-1+LagN/2 )/sincLn )/ ( dblPi*( (row+1.)/NN -col-1+LagN/2 )/sincLn );}
    }
}
```

図 17 Lagrange Lanczos(n)法ソースコードの一部

上記のソースコードは Lagrange(n)法を用いて補間計算の準備を行い, 色を付けている部分で sincF 関数を使い, Lanzos 法としての重み付けをする乗算を行って補間計算に用いる係数を求めている.

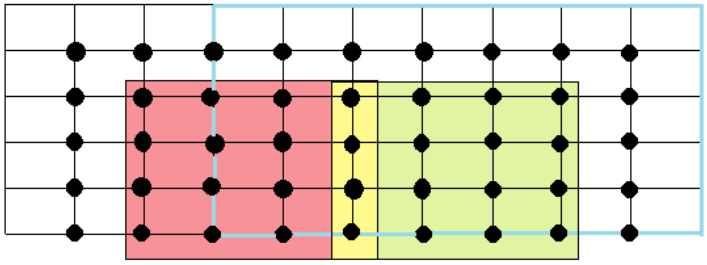
図 18 Lagrange Lanczos(4)法による拡大画像

6　補間の手法について

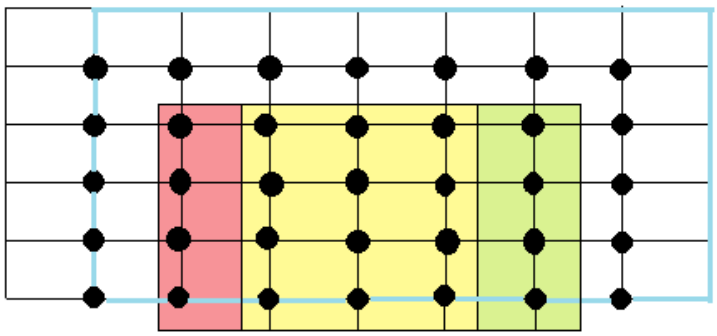
補間の手法にはブロック単位ごとの補間とポイント単位ごとの補間がある. ブロック単位ごとの補間では, まず原画像から大きさが n*n の画素ブロックを取り出し, このブロック内の点上ですべての画素を補間したら次のブロックに移り, これを画像全体に対して繰り返して行う. このとき 1 次多項式なら 2*2 の画素ブロックから 4 ピクセル単位, 3 次多項式なら 4*4 の画素ブロックから 16 ピクセルの画素値を用いてブロック内の補間計算をする.

ポイント単位ごとの補間の場合は, ブロックの移動や範囲の指定がブロック単位の場合と異なり, ブロック境界のピクセルを 1 ピクセルずらしながら 1*1 の画素ブロック内で補間計算していく.

3 次多項式の場合のそれぞれの補間方法を示す. 赤いブロックが 1 回目の補間, 緑のブロックが 2 回目の補間, 黄色のブロックが重複して補間された部分である (図 19).



(a)ブロック補間



(b)ポイント補間

図 19　横方向にブロックを重複させた補間

7.1 ブロック単位補間とポイント単位補間の比較

原画像をブロック単位とポイント単位の補間によって拡大し,比較と評価を行う.評価の方法としては,目視による主観的評価と,評価値を求めて行う客観的評価がある.ここでは主観的評価と,エントロピとして計算した数値に比較による客観的評価を行う.

表 1(a)　ブロック単位補間の画素情報量

補間次数	画素情報量(bit)
4	6.8526738
8	6.5371206
16	5.9499044
原画像	6.7541581

表 1(b)　ポイント単位補間の画素情報量

補間次数	画素情報量(bit)
4	6.6943993
8	6.4396536
16	5.8842059
原画像	6.7541581

図 20　4 次 Lagrange ブロック

図 21　4 次 Lagrange ポイント

図 22　8 次 Lagrange ブロック

図 23　8 次 Lagrange ポイント

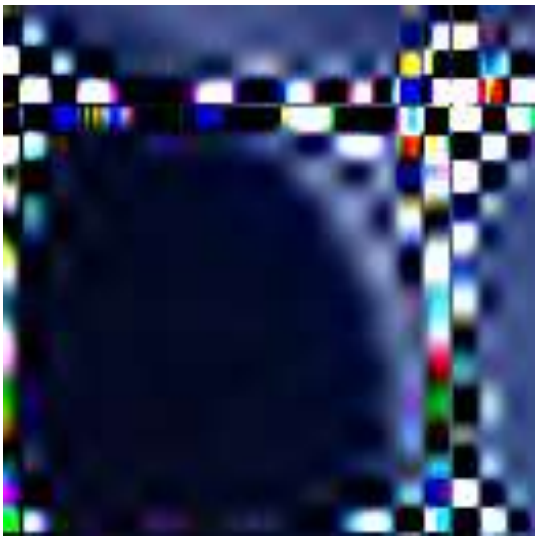


図 24　16 次 Lagrange ブロック

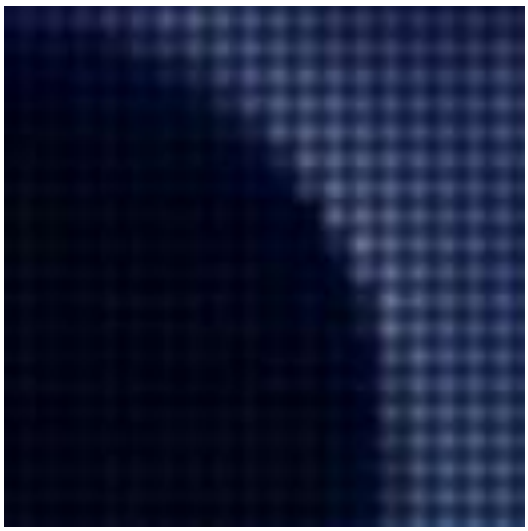


図 25　16 次 Lagrange ポイント

原画像にそれぞれ 4 次, 8 次, 16 次のブロック間またはポイント間の補間を行い, その一部を切り取った画像を示す(図 20, 21, 22, 23, 24, 25). 画像はすべて Lagrange 法で拡大している.

画素情報量による客観的評価を行う. 画素情報量の数値が原画像の数値に近いほど原画像の画質を維持しつつ拡大できているということになる. 数値の比較では 4 次ではポイント単位の補間がより原画像に近い数値となっているが, 8 次と 16 次ではブロック単位の補間のほうが原画像により近い数値となった(表 1a). 目視による主観的評価では. 同じ次数での補間を比較すると, 4 次の補間ではそれほど大きな差は見受けられないが(図 20, 21), 次数が 8 次, 16 次と高くなるにつれ画質の劣化の差が大きくなっていくのが分かる(図 22, 23, 24, 25). 特にブロック単位では画像の劣化が格子状に大きく表れており, ルンゲの現象が起きていることが分かる(図 20, 22, 24). 特に 16 次のブロック単位の補間では色の偏りが激しくかなりの劣化となっている(図 24). 主観評価ではややポイント単位の補間が優れていると分かった. これらの結果からエントロピの数値はブロック単位の補間が多少上回っているものの, 実際の拡大画像の画質としてはポイント単位の補間は計算量が同じで画質が優れていると結論づけることができる.

7.2 ルンゲ現象とギップス現象について

ルンゲの現象とは高次多項式を補間するときに, 次数 n が大きくなるにつれて誤差が大きく増大していく現象のことである. 多項式を用いる補間法の際に起こる現象のため sinc 関数, sincLanczos 関数を用いる補間の際には発生しない. ギップス現象とは sinc 関数と sincLanczos 関数で補間する場合に関数値が急激に変化する場所があり, その場所に生じる誤差振動のことである. これらの現象が補間のときに起こることによって画像が劣化する原因となっている.

8 補間法ごとの比較

Lagrange Lanczos(n)法の画像と画素情報量を用いて既存の補間法である sinc Lanczos(n)法との比較を行う. どちらも原画像に一番近い数値が出る条件となる sinc Lanczos(3)法と Lagrange Lanczos(4)法で比べている. この比較も手法による比較の時と同じように画素情報量が原画像に近ければ近いほど原画像の画質を維持しつつ画像を拡大できていることとなる. この時画素情報量の単位は bit となっている. 拡大画像の黒い枠は処理できていない端の部分であり, 端の処理には幾つかの手法があることから, 今回の比較ではこの部分については考えないものとする.

図 26-a sinc Lanczos(3)法での拡大

図 26-b　Lagrange Lanczos(4)法での拡大

表 2　補間法と画素情報量

画像補間法	画素情報量（bit）
sinc Lanczos(3)法	6.6261339
Lagrange Lanczos(4)法	6.7546139
原画像	6.7541581

目視による主観的評価を行った結果,画質に大きな違いを見受けることはできなかった.画素情報量による客観的評価ではLagrange Lanczos(4)法の画素情報量が原画像にかなり近い数値を出す結果となった(表 2).これらの結果からLagrange Lanczos(4)法がより優れていると結論付けることができる.

9.1　原画像の画素情報量が少ない場合の比較

今回提案したLagrange Lanczos(n)法は画素情報量が多い場合,既存の補間法と比べて原画像に近い数値が出ていた.そこで今度は画素情報量が少ない画像の場合でも既存の補間法よりも優れているのかを検証する. これまでと同様に原画像に一番近い数値となるsinc Lanczos(6)法と 4 次Lagrange法とLagrange Lanczos(4)法で比べている.また拡大画像の黒い枠は処理できていない端の部分であり,端の処理には幾つかの手法があることから今回の比較ではこの部分については考えないものとする.

図 27　原画像

図 28　sinc Lanczos(6)法

図 29　4 次 Lagrange 法

図 30　Lagrange Lanczos(6)法

表 3　補間法と画素情報量

画像補間法	画素情報量(bit)
sinc Lanczos(6)法	2.0244004
4 次 Lagrange 法	2.0441379
Lagrange Lanczos(4)法	1.1397756
原画像	0.2545134

目視による主観的評価では sinc Lanczos(6)法での拡大画像で線の近くにぶれが生じている(図 26). しかしそれ以外には特に違いは見受けられなかった. 画素情報量による客観的評価では Lagrange Lanczos(4)法が原画像に一番近い数値となった. しかし原画像の画素情報量が多い時と比べると, その精度は落ちていた(表 3).

そこでこれら以外の補間法を用いて比較を行うことにした.

図 31　ニアレストネイバー法

図 32　バイリニア法

図 33 バイキュービック法

表 4 補間法と画素情報量

画像補間法	画素情報量(bit)
ニアレストネイバー法	0.6571255
バイリニア法	1.1170777
バイキュービック法	0.9855584
原画像	0.2545134

目視による主観的評価では,やはりニアレストネイバー法でジャギーが発生しているが,バイリニア法やバイキュービック法に大きな差は見受けられない(図 31).画素情報量による客観的評価ではニアレストネイバー法が原画像に一番近い数値となり,Langrange Lanczos(6)法よりも原画像により近い数値となった(表 4).

9.2　原画像の画素情報量が少ない場合の比較（カラー版）

拡大の処理の際, カラー画像と黒のみの画像では計算量が異なるため, 画素情報量が少ないカラー画像を拡大した場合でも同じような結果が得られるのかを検証する．これまでと同様に原画像に一番近い数値となる sinc Lanczos(6)法と 4 次 Lagrange 法と Lagrange Lanczos(4)法で比べている. また拡大画像の黒い枠は処理できていない端の部分であり, 端の処理には幾つかの手法があることから, 今回の比較ではこの部分は考えないものとする.

図 34　原画像

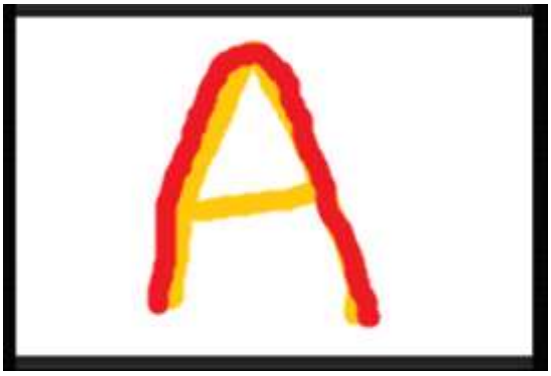


図 35　sinc Lanczos(6)法

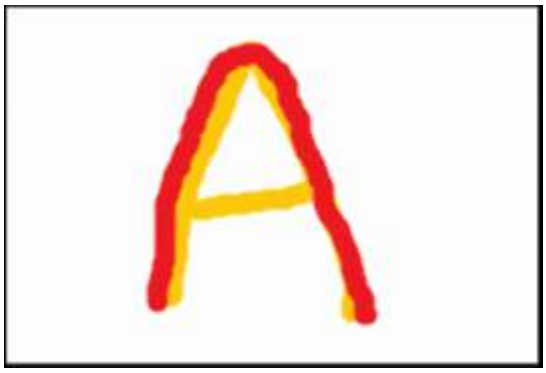


図 36　4 次 Lagrange 法

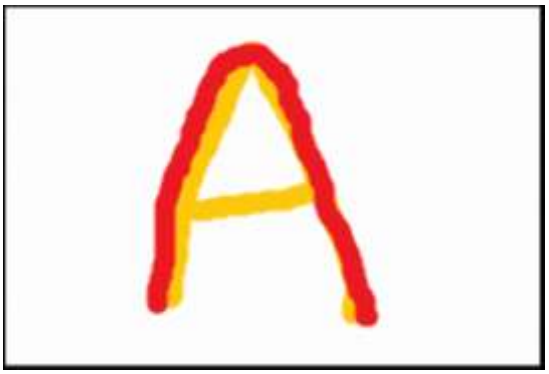


図 37　Lagrange Lanczos(4)法

表 5 補間法と画素情報量

画像補間法	画素情報量(bit)
sinc Lanczos(6)法	1.9910668
4 次 Lagrange 法	2.2223803
Lagrange Lanczos(4)法	1.3947199
原画像	0.8595197

目視による主観的評価では sinc Lanczos(6)法での拡大画像で,赤い線の外側にミシン目のようなジャギーが発生している(図 35).しかしそれ以外の違いは特に見受けられなかった.画素情報量による客観的評価では Lagrange Lanczos(4)法が原画像に一番近い数値となった(表 5).画素情報量の多い画像を拡大した場合と比べてやはり精度は落ちているものの,先ほどの黒のみの画像の拡大の場合よりは高い精度で拡大できていた.

これらのことから今回提示する Lagrange Lanczos(n)法は sinc Lanczos(n)法や Lagrange 法よりも優れた拡大画像を作り出せる.しかし画素情報量が少ない画像やシンプルな画像の場合はバイリニア法やバイキュービック法の方がよい結果となった.またニアレストネイバー法は性質上原画像を忠実に再現することができるがジャギーが発生する粗い画像であるため,どちらを良いとするかは個人の好みによると思われる.

10　まとめ

ブロック単位の補間とポイント単位の補間では,ポイント単位での補間の方が優秀であるという結論が出た.今回の補間に用いたプログラムでは,画像の端の部分の計算処理をしておらず,端を切り取って画像の比較を行っている．実用化するためには,端の部分も正確に処理する必要がある．また既存の補間との比較によって Lagrange Lanczos(n)法による補間が,一番原画像に近い画質を維持しつつ拡大できることが分かった.しかし客観的評価として画素情報量が一番近いということでしか評価しておらず主観的評価も数人でしか行っていないため,精度を上げるためにより多くの人に主観的評価を行うことやほかの評価方法を利用した比較をすることを今後の課題とする.

参考文献

[1]Visual Studio2017 に OpenCV3.2.0 と opencv_contrib を導入する方法
https://qiita.com/tomochiii/items/fa26404ebc5fcd4481b9
[2] Visual Studio 2017 + OpenCV 3.2.0 + x64 の初期設定
http://phst.hateblo.jp/entry/2016/11/30/081749

[3]画像補間法による拡大,田中・仲,岡山理科大学卒業論文 2015 年度
[4]補間法による拡大画像の比較, 神崎・大霜・坂本,岡山理科大学卒業論文 2016 年度
[5]画像の解像度変換について,三宅勇輔,岡山理科大学修士論文 2016 年度