プログラミング課題（5/19）

氏名：園山佳典
学籍番号：26002201991

1. はじめに

本課題では、MATLAB を用いてカラー画像を読み込んで、x 方向（横方向）を 1.5 倍、y 方向（縦方向）を 1.2 倍に拡大した画像をそれぞれニアレストネイバー補間法とバイリニア補間法で生成する。

2. 方法

● ニアレストネイバー補完法

元の画像を imread で読み込み image に代入、拡大後の画像を new_image_nearest とし、拡大倍率を(a,b)とする。
また、元の画像の幅と高さを(width,height)とする
拡大後のサイズは(new_width,new_height)とし、

```
new_width = round(width * a);
new_height = round(height * b);
```

というふうに元の画像の幅と高さをそれぞれ a 倍、b 倍し拡大する。
round は四捨五入を行う関数です。
拡大後の画像の各ピクセル(x,y)を求めるために、元画像の座標(orig_x,orig_y)を下記のコードで逆変換する。

```
orig_x = round(x / a);
orig_y = round(y / b);
```

その後、求めた(orig_x,orig_y)が元の画像の範囲内にあるか if 文を用いて判定する。
元の画像の範囲内にある場合、拡大後の画像のピクセル(x,y)に元の画像の最も近い隣接ピクセルの値を代入する。この手順を for 文を用いて拡大後の画像の全てのピクセルに対して行い、ニアレストネイバー補完法による拡大後の画像を生成する。

● バイリニア補完法

拡大後の座標を**(x, y)**とし、元の座標を**(orig_x, orig_y)**とする。
floor、ceil 関数により元の座標の数値の切り捨てと切り上げを行い、最も近い隣接ピクセルの座標を求める。

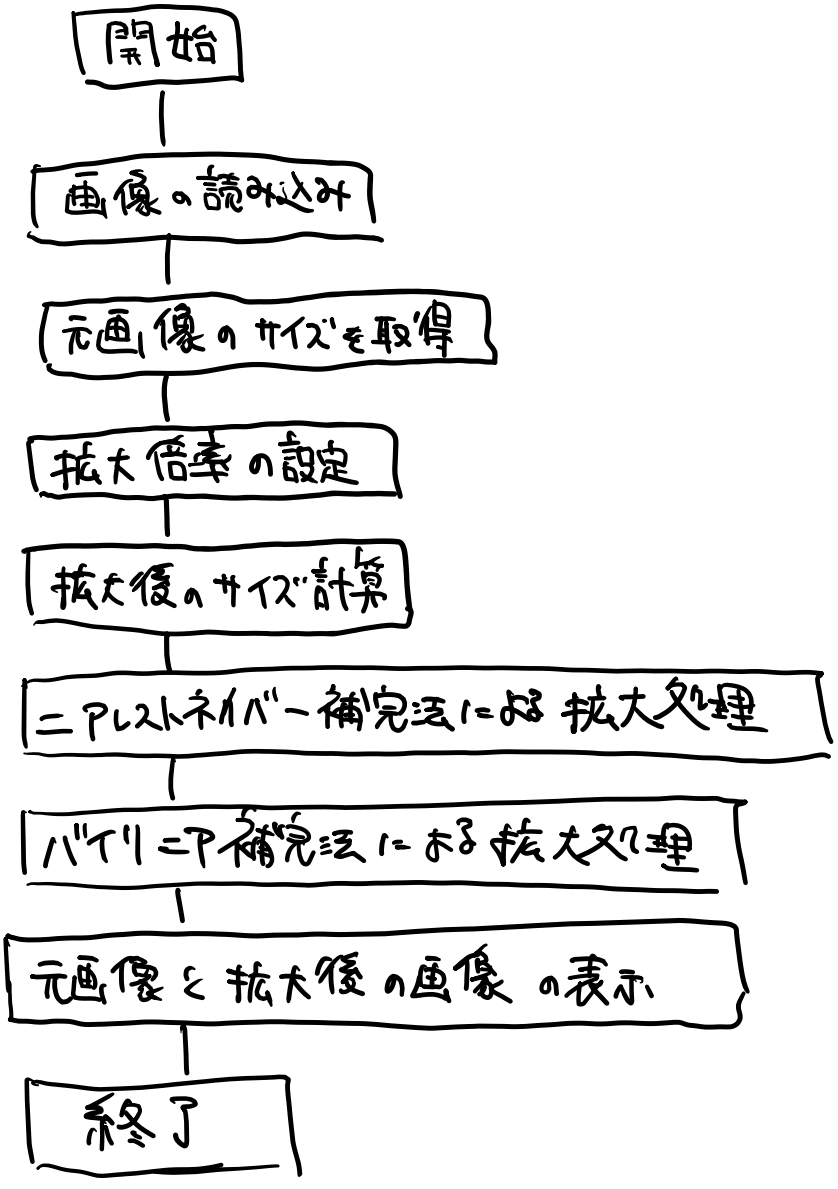
```
x1 = floor(orig_x);
y1 = floor(orig_y);
x2 = ceil(orig_x);
y2 = ceil(orig_y);
```

そして、各隣接ピクセルの値を取得する際に元の画像の範囲を超えないように if 文をもちいて処理する。取得した 4 つのピクセルをそれぞれ pixel1、pixel2 、pixel3、 pixel4 に代入する。その後 pixel1 から pixel4 までの重みを計算し各ピクセルの値に重みをかけて足し合わせる。

```
weight1 = (x2 - orig_x) * (y2 - orig_y);
weight2 = (orig_x - x1) * (y2 - orig_y);
weight3 = (x2 - orig_x) * (orig_y - y1);
weight4 = (orig_x - x1) * (orig_y - y1);
interpolated_pixel = weight1*pixel1 + weight2*pixel2 + weight3*pixel3 +
weight4*pixel4;
```

これにより拡大後の画像の各ピクセルの値が求まる。

３．アルゴリズム



4.結果

入力画像

出力画像(ニアレストネイバー補完法)

出力画像(バイリニア補完法)

５．考察

下の２つの図からもわかるように、

ニアレストネイバー補完法では求めた位置に最も近い画素一の値をそのまま利用するためピクセルの配置が粗くなるのに対し、バイリニア補完法では周りの４点の画素値の重み付の平均値を求めるため滑らかな補完ができる。

ニアレストネイバー補完法

バイリニア補完法

６．付録：プログラムリスト

```
% 画像読み込み
image = imread('manzi.jpg');

% 拡大倍率
a = 1.5; % x 方向の拡大倍率
b = 1.2; % y 方向の拡大倍率

% 元画像のサイズ
[height, width, ~] = size(image);

% 拡大後のサイズ
new_width = round(width * a);
new_height = round(height * b);

% 拡大後の画像を生成（ニアレストネイバー補間法）
new_image_nearest = zeros(new_height, new_width, 3, 'uint8');
for y = 1:new_height
```

```
    for x = 1:new_width
        % 拡大前の座標を計算
        orig_x = round(x / a);
        orig_y = round(y / b);

        % 拡大前の座標が元画像の範囲内であるか
        if orig_x >= 1 && orig_x <= width && orig_y >= 1 && orig_y <= height
            % 拡大後の画像に値を代入
            new_image_nearest(y, x, :) = image(orig_y, orig_x, :);
        end
    end
end

% 拡大後の画像を生成（バイリニア補完）
new_image = zeros(new_height, new_width, 3, 'uint8');
for y = 1:new_height
    for x = 1:new_width
        % 拡大後の座標を逆変換して元の座標を求める
        orig_x = (x-0.5) / a + 0.5;
        orig_y = (y-0.5) / b + 0.5;

        % 元の座標に最も近い隣接ピクセルの座標を求める
        x1 = floor(orig_x);
        y1 = floor(orig_y);
        x2 = ceil(orig_x);
        y2 = ceil(orig_y);

        % 各隣接ピクセルの値を取得する
        if x1 < 1
            x1 = 1;
        end
        if x2 > width
            x2 = width;
        end
        if y1 < 1
            y1 = 1;
```

```
            end
            if y2 > height
                y2 = height;
            end

            pixel1 = double(image(y1, x1, :));
            pixel2 = double(image(y1, x2, :));
            pixel3 = double(image(y2, x1, :));
            pixel4 = double(image(y2, x2, :));

            % バイリニア補完で新しいピクセルの値を求める
            weight1 = (x2 - orig_x) * (y2 - orig_y);
            weight2 = (orig_x - x1) * (y2 - orig_y);
            weight3 = (x2 - orig_x) * (orig_y - y1);
            weight4 = (orig_x - x1) * (orig_y - y1);
            ip = weight1*pixel1 + weight2*pixel2 + weight3*pixel3 +
weight4*pixel4;

            % 拡大後の画像に値を設定する
            new_image(y, x, :) = uint8(ip);
        end
    end

    % 元の画像と拡大後の画像を表示
    figure;
    imshow(image);
    title('元の画像');
    figure;
    imshow(new_image_nearest);
    title('ニアレストネイバー補間法');
    figure;
    imshow(new_image);
    title('バイリニア補完');
```