Ⅰ.序論

今回の画像修復のプログラムでは、画像が AR モデルで近似できると仮定して、線形モデルとして見立てて処理を行う。これは画像のそれぞれのピクセルに対して周辺のピクセル情報から元情報を計算で求め修復を行う方法である。この方法を用いて、試行回数、既知の画素値の割合、画像パッチの値を変化させて画像修復の精度について考察していく。また、パラメータ a が既知であるかどうかによる画像修復の精度の違い、修復方法の違いによる画像修復の精度の違いについても考察していく。修復方法はランダムスキャンとランダムラインスキャンを採用する。

Ⅱ.本論

この条件下での画像修復アルゴリズムは以下のようになり、修復画像を得ることができる。

1.未知の画素の値を 0 とする
2.以下を繰り返す（適切な回数繰り返す）
　2-1.画像からXとbを生成する
　2-2.Xとaを用いてbを計算する
　2-3.得られたbのうち、画素値が既知の要素は既知の値を代入
　2-4.上記で得られた b を並び替えて、画像を生成

図 1 a が既知である場合の画像修復アルゴリズム

1.未知の画素の値を 0 とする
2.以下を繰り返す（適切な回数繰り返す）
　2-1.画像からXとbを生成する
　2-2.Xとbからaを計算
　2-3.Xと上記で得られたaを用いてbを計算する
　2-4.得られたbのうち、画素値が既知の要素は既知の値を代入
　2-5.上記で得られた b を並び替えて、画像を生成

図 2 a が未知である場合の画像修復アルゴリズム

今回の実験では、これらのアルゴリズムを用いて縦横比 199×300 の画像に対して画像修復を行った。また、以下に示す結果はすべて、各実験につき 5 回実行して得られたエラー値の平均をとったものをその結果のエラー値とした。

1 .試行回数を変化させた場合

はじめに試行回数を変化させて実験を行った。既知の画素値の割合を r=0.3、画像パッチを p=5 で固定し、パラメータ a を未定として試行回数を 1 から 35 回に変化させた。アルゴリズムは図 2 を使用した。

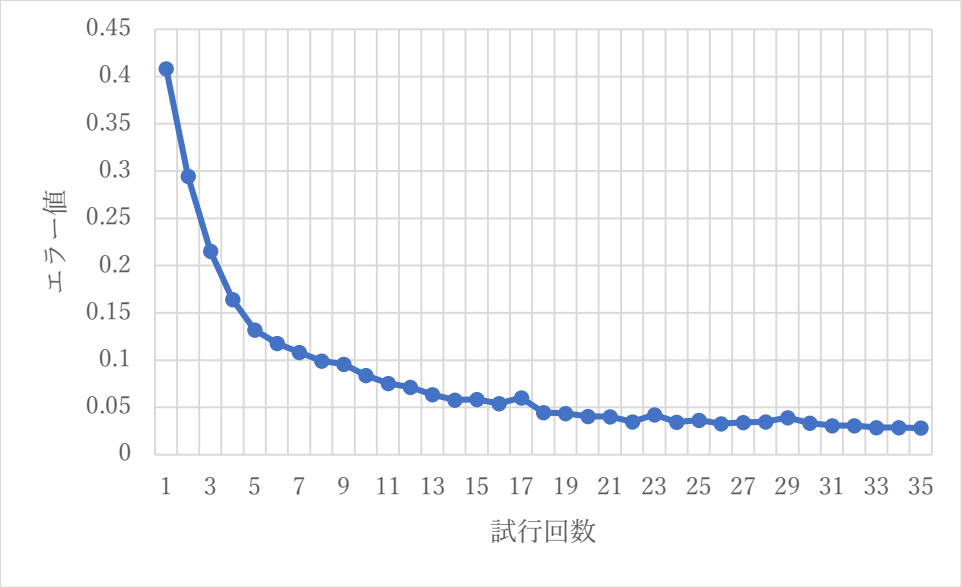

図 3 試行回数とエラー値

図 3 では横軸を試行回数、縦軸をエラー値とした。試行回数が 18 回以降ではエラー値の変化量が小さくなり、その値は約 0.040 に収束した。そのため、試行回数は約 20 回が妥当であるといえる。

2.既知の画素値の割合を変化させた場合

次に既知の画素値の割合を変化させて実験を行った。試行回数を k=20、画像パッチを p=5 で固定し、既知の画素値の割合を 0.1 から 1.0 まで 0.1 ずつ変化させた。アルゴリズムは図 2 を使用した。

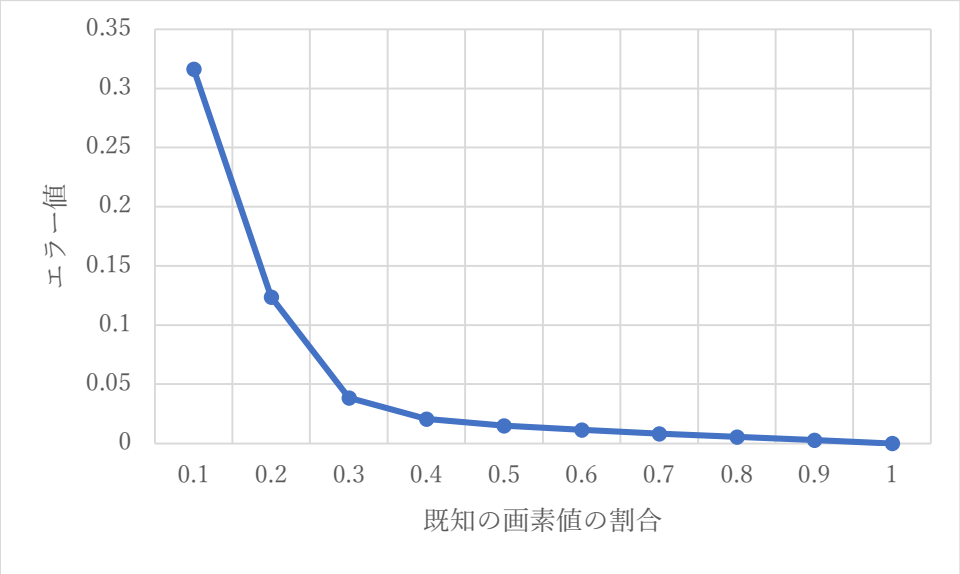

図 4 既知の画素値の割合とエラー値

図 4 では横軸を既知の画素値の割合、縦軸をエラー値とした。既知の画素値の割合が大きくなるほどエラー値が小さくなった。これは既知の画素値の割合が大きくなるほど、修復する必要がある画素値の割合が小さくなっているからと考えられる。そのため、既知の画素値の割合が 1 のときはすべての画素値が既知であるため、エラー値が 0 となっている。

3．画像パッチを変化させた場合

次に画像パッチを変化させて実験を行った。試行回数を k=20、既知の画素値の割合を r=0.3 で固定し、画像パッチを 3 から 21 まで 2 ずつ変化させた。アルゴリズムは図 2 を使用した。

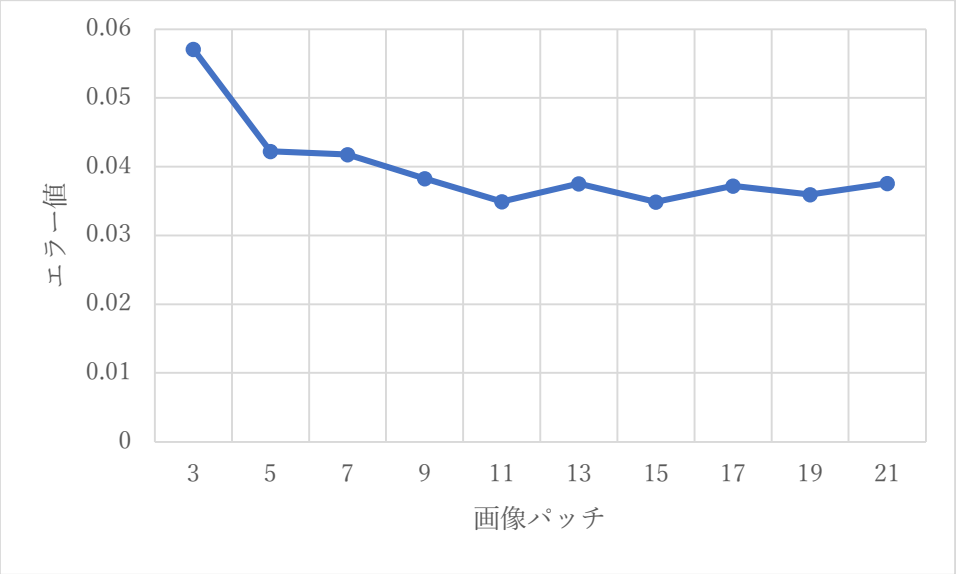

図 5 画像パッチとエラー値

図 5 では横軸を画像パッチ、縦軸をエラー値とした。画像パッチの値が p=3 を除き、エラー値はほぼ横ばいとなった。よって、画像パッチの値はエラー値に影響しないことが考えられる。また、画像パッチが大きくなるほど、実行時間が大きくなった。

4．パラメータ a が既知であるかどうかを変化させた場合

次にパラメータ a が既知であるかどうかを変化させて実験を行った。試行回数を k=20、既知の画素値の割合を r=0.3、画像パッチを p=5 で固定し、パラメータ a が既知である場合図 1 のアルゴリズムをパラメータ a が未知である場合図 2 のアルゴリズムを使用した。

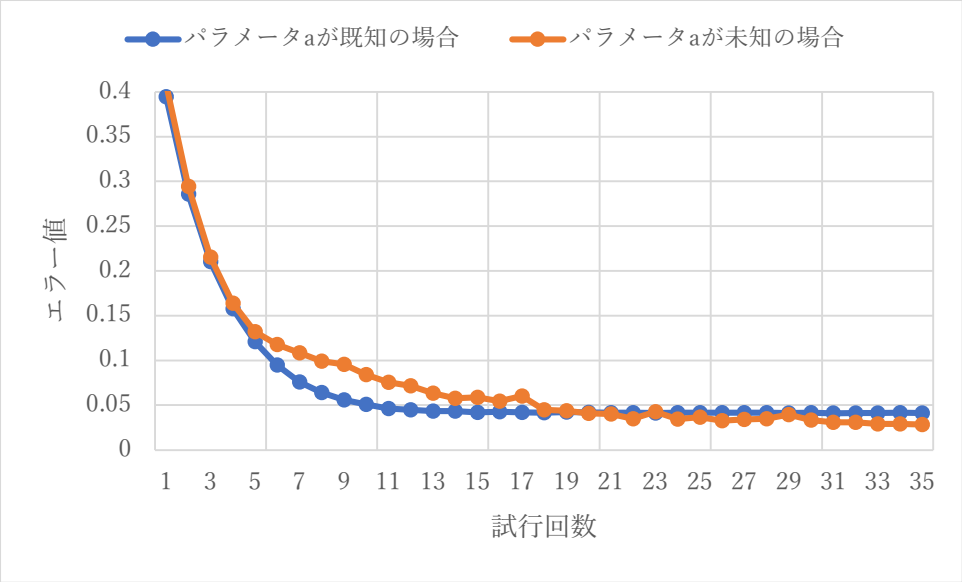

図 6 パラメータ a が既知であるかパラメータ a が未知であるか

図 6 では横軸を試行回数、縦軸をエラー値とした。青線はパラメータ a が既知である場合、オレンジ線はパラメータ a が未知である場合を表している。試行回数が 18 回以降ではどちらもエラー値の変化量が小さくなり、その値は約 0.040 に収束した。しかし、パラメータ a が未知である方がエラー値の変化量が小さく、パラメータ a が既知である方が先に収束値に達した。図 2 のアルゴリズムはパラメータ a を推定する計算を行うため、図 1 のアルゴリズムよりも誤差が大きくなるといえる。

5．修復方法を変化させた場合

最後に修復方法を変化させて実験を行った。試行回数を k=20、既知の画素値の割合を r=0.3、画像パッチを p=5 で固定し、既知の画素の位置の生成方法をランダムスキャンとランダムラインスキャンを採用した。アルゴリズムは図 2 を使用した。ランダムスキャンでは、ランダムに選択された画素のみを修復し、ランダムラインスキャンではランダムに引かれた直線上の画素のみを修復した。直線の数は 400 本引いた。

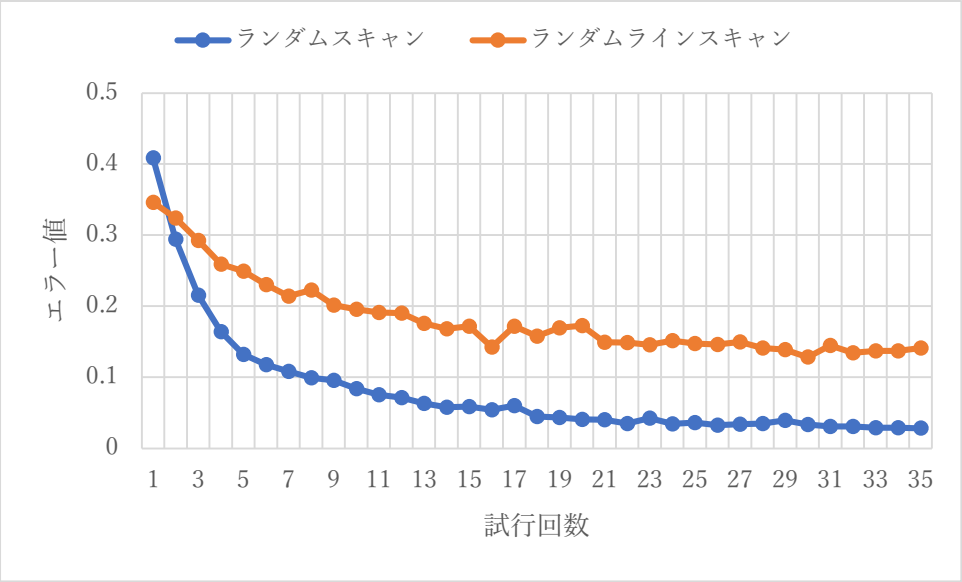

図 7 ランダムスキャンとランダムラインスキャン

図 7 では横軸を試行回数、縦軸をエラー値とした。青線はランダムスキャンによる修復、赤線はランダムラインスキャンによる修復を表している。どちらも試行回数が 18 回以降ではエラー値の変化量が小さくなった。しかし、ランダムスキャンは約 0.040 に、ランダムラインスキャンは約 0.140 に収束した。ランダムスキャンは画像全体を満遍なく修復できるのに対し、ランダムラインスキャンは直線の引き方によって修復の仕方に偏りが生じる。そのためランダムスキャンに比べて、ランダムラインスキャンはエラー値が大きい値で収束している。また、エラー値も安定せずランダムスキャンに比べて、ランダムラインスキャンの方がきれいな曲線が得られなかった。

Ⅲ.考察

MATLAB による画像修復の精度を上げるには、試行回数はある程度確保するべきである。しかし、一定数の試行回数を重ねるとエラー値は変化しなくなる。そのため、試行回数は 20 回程度行うべきである。既知の画素値の割合は、大きければ大きいほど修復する必要がある画素の数が減少するので大きい方がよい。しかし、実際の画像修復では既知の画素値を自分で設定することはできないので今回は考えない。画像パッチの大きさによる画像修復の精度に大きな違いは見られなかった。しかし、パッチ数が大きいほど実行にかかる時間が長かったため、素早く修復を行うにはパッチ数は最小の 3 にするべきである。パラメータ a は既知であることが望ましい。なぜなら、パラメータ a が未知である場合、パラメータ a を求めるときに誤差が余分に発生し、その結果得られるエラー値が大きくなったからである。修復方法はランダムスキャンの方がよい。なぜなら、ランダムスキャンは修復する画像の位置を満遍なく得られることができるので高い修復精度が得られる。ランダムラインスキャンは引く直線の数や位置に影響する。直線の数を増やすことで高い修復精度を得ることができるが、引く直線の位置によって修復精度にばらつきが生じる。そのため、安定した修復結果を得るためにはランダムスキャンを採用するのがよいと考えた。