卒業論文

ＦＰＧＡを用いた液晶用ガラスの欠損検出
画像処理の高速化（Ⅱ）

氏名　　：前田　峻
学籍番号：2260070084-0
指導教員：山崎　勝弘　教授
提出日　：2011 年 2 月 17 日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

本論文では株式会社 KDE から提供された液晶ガラス基板の画像データを対象に、C 言語による画像処理プログラムを作成し、実験を行った。

まず TDI による画像のノイズ除去を行い、ラプラシアンフィルタによるエッジの抽出、さらに 2 値化とラベリングによってノイズ量を検出することで、ノイズ除去効果の検証を行った。640*480 サイズの bmp ファイルに 128 回の TDI 処理を行うことで、画像のノイズを元データの約 7 分の 1 に減少させることが確認された。

また、それらの画像処理プログラムの高速化として、Verilog-HDL による FPGA ボードへの実装の検討も行った。

目次

図目次

1　はじめに

近年の半導体製造技術により、LSI の小型化、高速化、省電力化などは恐るべき速度で進歩を続けている。特に、複数の機能を 1 つのチップ上に集積したシステム LSI は携帯電話や、デジタルカメラ、ゲーム機など、我々が日常で手にしている様々な電子機器に広く利用され、高い付加価値を与えている。一方でこれらに用いられるアプリケーションやアーキテクチャの多様化、大規模化は半導体設計の複雑化という問題をもたらしている。また、使用される部品の微細化に伴い、わずかな欠陥がシステム全体に大きな影響を及ぼすことがある。中でも大型の液晶ディスプレイや、小さな半導体部品の一部等の広い範囲にわたって用いられている液晶ガラスはその光景が顕著に表れている。製造工程において目に見えないほどのわずかな傷や気泡が含まれることで性能の低下、あるいは故障に繋がるほどであり、より設計品質の向上が求められるようになった。

そのため、製造工程において厳重なチェックを行い、わずかでも欠損が確認されたものは取り除かれる。ここで必要となるのが高精度な画像処理技術である。微細化された部品の欠損を肉眼で確認することはもはや不可能であり、その数も膨大である。従ってカメラで画像を取り込み、コンピュータによって確認・判別作業を自動化させることが必須となる。しかし余りにも微小な欠損を検出する必要があるため、撮影に際に混じったノイズを欠損として誤検出してしまう場合がある。そのようなことを防ぐために、判別画像におけるノイズを取り除く画像処理が必要となる。

本研究では、液晶用ガラスの欠損検出のための FPGA を用いた画像処理の高速化を目的としている。本研究では、TDI（Time Delay Integration）処理による撮影画像のノイズ除去、ラプラシアンフィルタ、及びラベリング処理を用いたガラスの傷の検出を行う。また、生産量に対して欠損検査に時間がかかってしまうと作業効率を下げることにつながるため、FPGA ボードへの実装による画像処理プログラムの高速化を目標としている。

画像処理はまず TDI 処理によってノイズ除去を行った上で、ラプラシアンフィルタをかけて画像の 2 値化を行い、ラベリング処理によって欠損部分の検出行う。それらのプログラムを C 言語によって作成し、PC 上で効果を確認する。また、ノイズ除去に用いる TDI 処理の回数を 32 回、64 回、128 回と回数を変えることでどれほどの効果が表れるかの検証も行う。さらに Xilinx 社のハードウェアデザインツール ISE Design Suite11 によってハードウェア記述、及び ModelSim XEⅢ6.4b によりシミュレーションを行う。以上の流れから設計したソフトウェアと回路の評価を行う。

本論文では TDI、ラプラシアンフィルタ、ラベリングの 3 つの画像処理プログラムのアルゴリズムについて説明し、実験結果によって効果を検証する。そしてソフトウェアの verilog-HDL を用いたハードウェア記述について述べる。

２　ガラス欠損検出画像処理のためのアルゴリズム

2．１　TDI(Time Delay Integration)

TDI とは、同じ画像の画素をずらして撮影を繰り返し、その共通部分を重ね合わせ平均値を取ることで、ノイズの影響が小さい画像を得る手法である。

画像は撮影ごとにノイズの量や含まれる場所が変わり、ノイズによって実際の画素値がプラスされたりマイナスされたりする。そこで目標の画像を複数枚撮影し、全てを足し合わせて平均値を取る。それにより、ノイズによってプラスされた画素とマイナスされた画素の平均を取ることで、実際の画素値に近い値を得ることができる。しかし、同条件で撮影を繰り返すとノイズの入る場所も似通ってしまう。そのため、撮影する画像を 1 画素ずつずらしていくことでノイズの条件を変えていく。図 1 に画像のイメージを用いて例を示す。

図１．得たい画像イメージ

例えば、このような横 6 ピクセル、縦 7 ピクセルの画像があったとする。この数値を画像の輝度値としてみなすと、3 から 9 までの値が並んだ画像といえる。この画像を得るためにカメラで撮影する。

図２．実際に撮影した画像

実際には図 2 のように撮影の際にノイズが混じり、綺麗な画像とはいかなくなる。赤い字の部分がノイズであり、実際の画像と比べると値が違っていることがわかる。

ここで、欲しい画像よりも少し縦の範囲を広げて画像の撮影を行う。この例では縦に 3 ピクセル伸ばした画像を用い、そのイメージ図を図３に示す。本来は縦 7 ピクセルのところを 10 ピクセルまで広げ、3～9 までだった画像が、0～9 までとなっている。当然、この画像にもところどころノイズが混じる。

0	0	0	0	0	0
1	1	1	0	0	0
2	2	2	2	1	2
3	3	3	2	3	3
4	4	4	4	3	4
5	5	5	4	6	5
6	6	6	6	6	6
7	7	7	7	7	7
8	8	8	8	8	8
9	9	9	9	9	9

図３．縦３ピクセル分を増やして撮影

これを縦に１画素ずつずらして撮影した画像を４枚用意する。今回の場合は 1～10、2～11、3～12 までの画像である。全体的に、右上を中心にノイズが混じっている。

0	0	0	0	0	0
1	1	1	0	0	0
2	2	2	2	1	2
3	3	3	2	3	3
4	4	4	4	3	4
5	5	5	4	6	5
6	6	6	6	6	6
7	7	7	7	7	7
8	8	8	8	8	8
9	9	9	9	9	9

1	1	1	2	1	0
2	2	2	2	1	2
3	4	3	4	3	3
4	4	4	5	4	4
5	4	5	5	5	6
6	6	6	5	6	6
7	7	7	7	7	7
8	8	8	8	8	8
9	9	9	9	9	9
10	10	10	10	10	10

2	2	2	1	3	3
3	3	3	3	4	2
4	4	4	5	4	4
5	5	5	6	5	5
6	6	6	6	5	6
7	7	7	7	7	6
8	8	8	8	8	8
9	9	9	9	9	9
10	10	10	10	10	10
11	11	11	11	11	11

3	3	3	4	2	4
4	4	4	3	4	5
5	5	5	5	6	4
6	6	6	5	5	6
7	7	7	6	7	7
8	7	8	8	7	8
9	9	9	9	9	9
10	10	10	10	10	10
11	11	11	11	11	11
12	12	12	12	12	12

図４．TDI 処理のため、画像を４枚用意

ここで各画像の画素値が横に並ぶよう、図５のように画像の高さを１ずつずらしていく。4 枚の共通部分であり、本来求めていた 3～9 の部分を紫色で網がけしている。

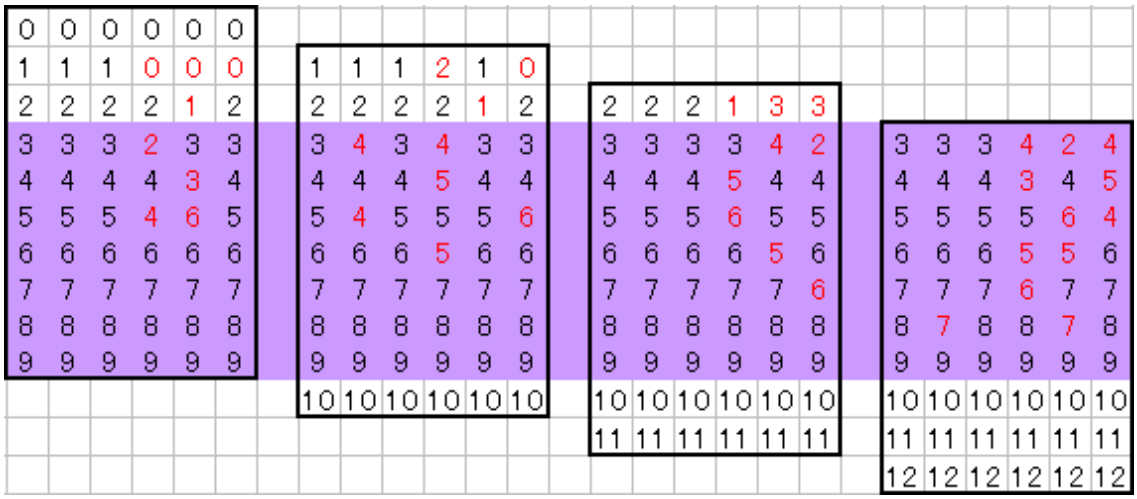


図５．画像の高さを合わせる

この共通部分だけを切り取るとこのような図 6 のような 4 枚ができあがる。

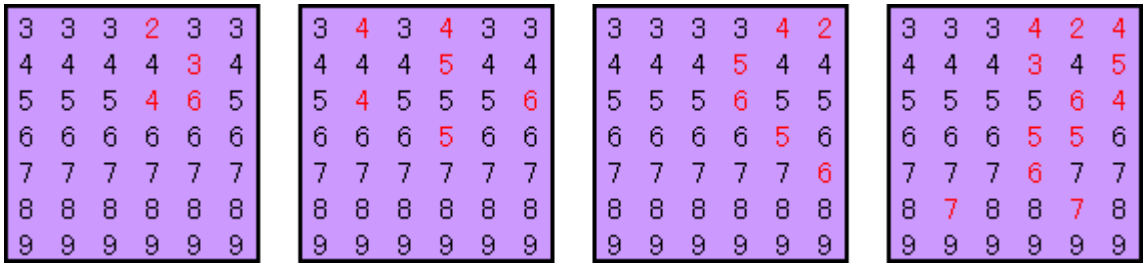


図 6．共通部分だけを取り出した

図 6 の画像を全て足し合わせ、足した枚数で割り、平均値を求めると以下のようになる。

3	3.25	3	3.25	3	3
4	4	4	4.25	3.75	4.25
5	4.75	5	5	5.5	5
6	6	6	5.5	5.5	6
7	7	7	6.75	7	6.75
8	7.75	8	8	7.75	8
9	9	9	9	9	9

図 7．TDI 処理後の画像

3～9 が並んだ綺麗な画像とはいかないが、普通に撮影した場合の元画像（図 8）に比べてノイズの影響が小さくなっていると言える。

図 8．元画像

今回の例では 4 枚の画像を足し合わせて平均値を取ったので、TDI 処理を 4 回実行したことになる。4 回だけではノイズの影響を小さくするのに十分な効果があったとは言い難いが、この実行回数を増やせば増やすほどノイズの影響を小さくすることが期待できる。

2．2　TDI 処理の実行結果

本研究では図 9 の画像を用いて実験を行う。実際の液晶ガラスの一部分を撮影したものであり、画像には多量のノイズが含まれている。また、図中の赤で囲っている部分が製造工程でできた傷である。

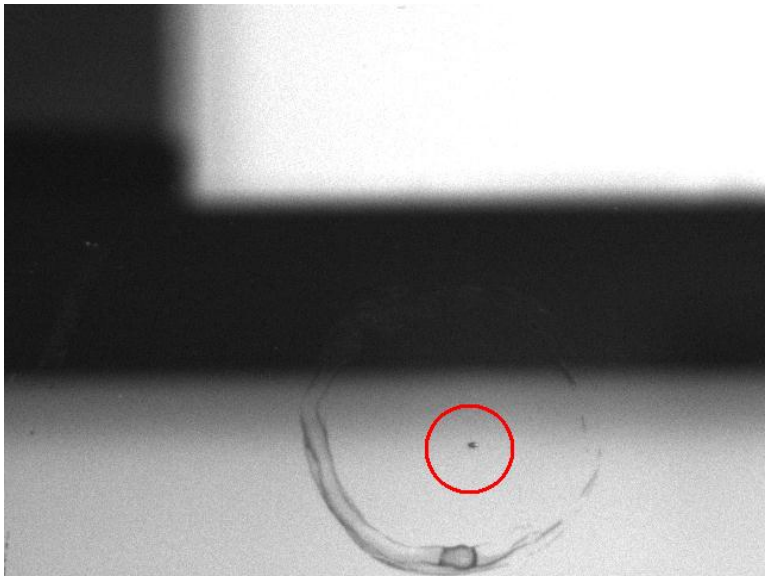


図 9．研究で用いる画像

これに TDI 処理を 32 回かけたものが図 10 である。下 32 ピクセル分は画像が黒ずんでしまうため、わかりやすいように黒く塗りつぶしてある。

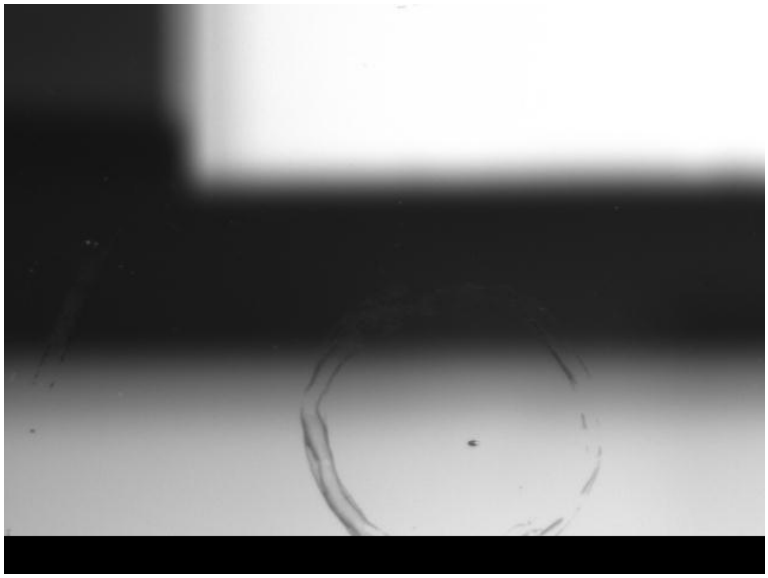
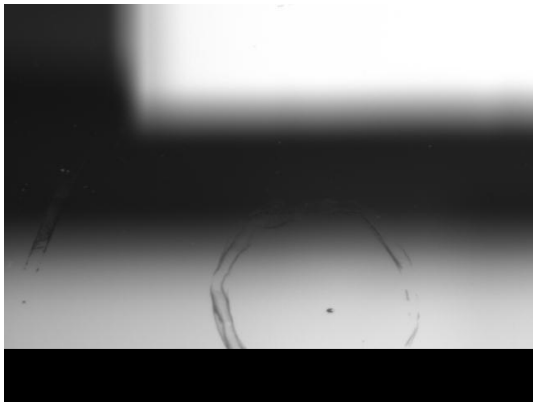


図 10．TDI を 32 回実行

また、図 11(a)(b)は同様に TDI をそれぞれ 64 回、128 回行ったものである。実行回数が増えるにつれ、画像は小さくなってしまう。



(a). TDI64 回

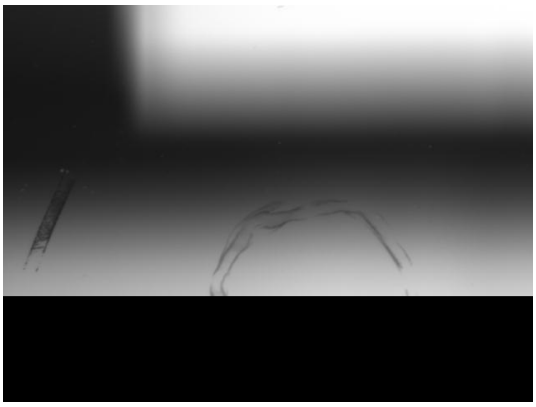


図 10(b). TDI128 回

図 11. それぞれ TDI を 64 回,128 回実行

元画像はノイズの影響でガラスの表面部分が荒く写っているため、傷の検出処理をかけても誤検出を行ってしまう可能性がある。それに対し、TDI 実行後はノイズの影響が減って画像が鮮明になっているため、傷の検出処理も正常に行われると考えられる。ただし、これらの画像からもわかるように、TDI 処理を多くするほど画像は小さくなるという欠点もある。また、TDI 処理 32 回、64 回、128 回の間ではそれほど大きな違いも見られないため、処理回数を増やしすぎると時間が無駄になっていることも考えられる。これらのことから、TDI 処理は適切な処理回数を行うことが求められる。現時点では 32 回で十分な結果が得られているため、32 回で十分なのではないかということが挙げられる。

2．3　ラプラシアンフィルタ

ラプラシアンフィルタは画像中に含まれる物体の輪郭部分（エッジ）を抽出するフィルタである。ラプラシアン∇_2は空間二次微分を行うオペレータであり、次のような演算を行う。

$$\nabla^2 f(x,y) = f_{xx}(x,y) + f_{yy}(x,y)$$

これを差分形式であらわすと

$$\nabla^2 f(x,y) = f_{xx}(x,y) + f_{yy}(x,y)$$

$$= f(x-1,y) - 2f(x,y) + f(x+1,y) + f(x,y-1) - 2f(x,y) + f(x,y+1)$$
$$= f(x-1,y) + f(x+1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)$$

これを係数で表現すると

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

となる。これを４近傍のラプラシアンフィルタという。また、４５°方向も含めると

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

という形式になり、８近傍のラプラシアンフィルタといい、この形も良く使われる。
これらの係数をマスクパターンとして注目画素の近傍に以下の 3*3 のオペレータを使って微分を行う。処理の流れを図 12 に示す

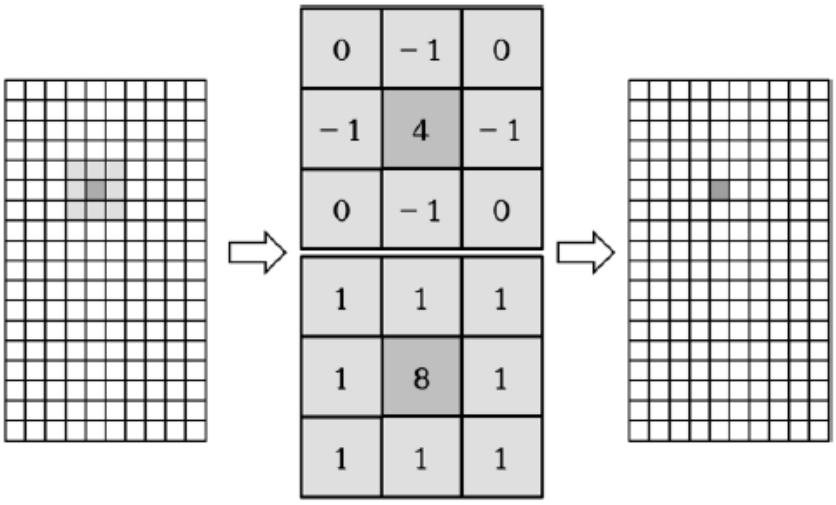
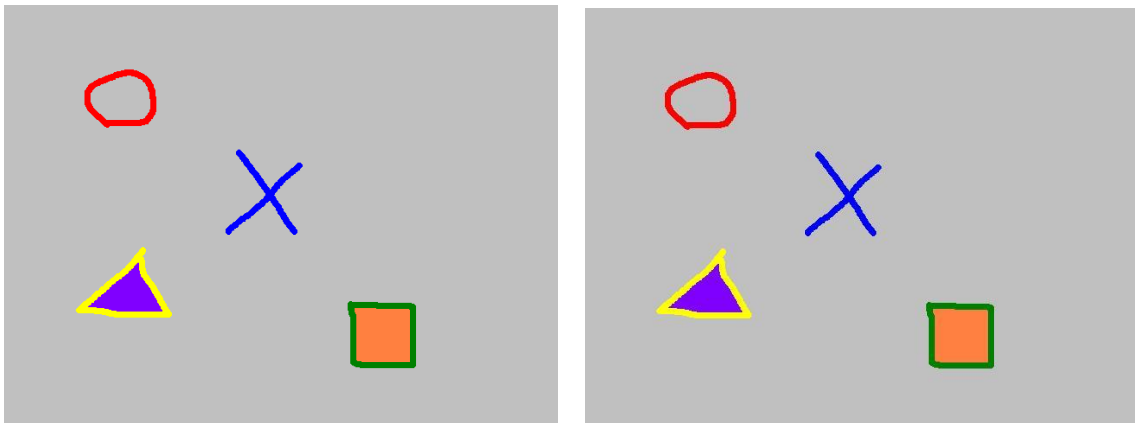


図 12. ラプラシアンフィルタ処理

2．4 ラプラシアンフィルタの実行例

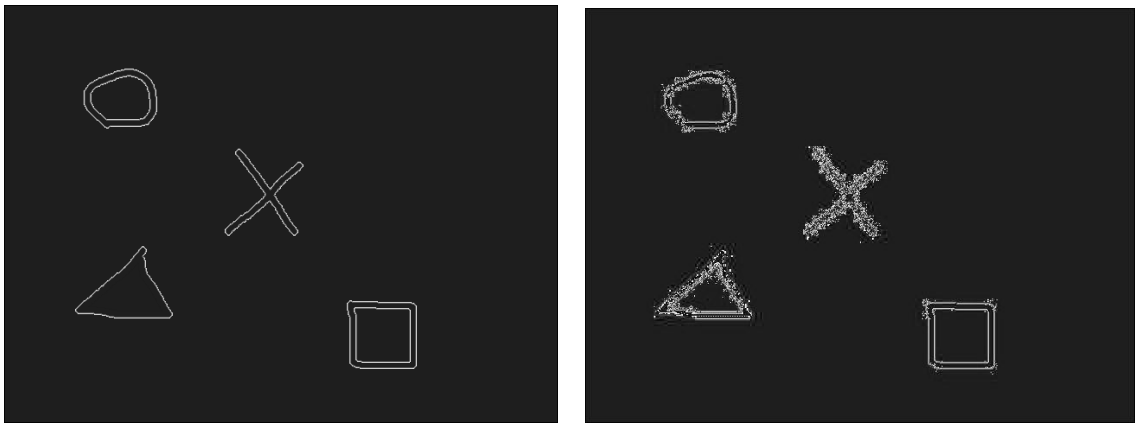
ラプラシアンフィルタの実行例を以下に示す。



(a)ノイズなし　　(b)ノイズあり

図 13．ラプラシアンフィルタ実行例

図 13 の 2 枚を見比べても何の違いも見られないが、右側の図にはノイズが含まれている。この 2 枚にそれぞれラプラシアンフィルタをかけると図 14 のようになる。

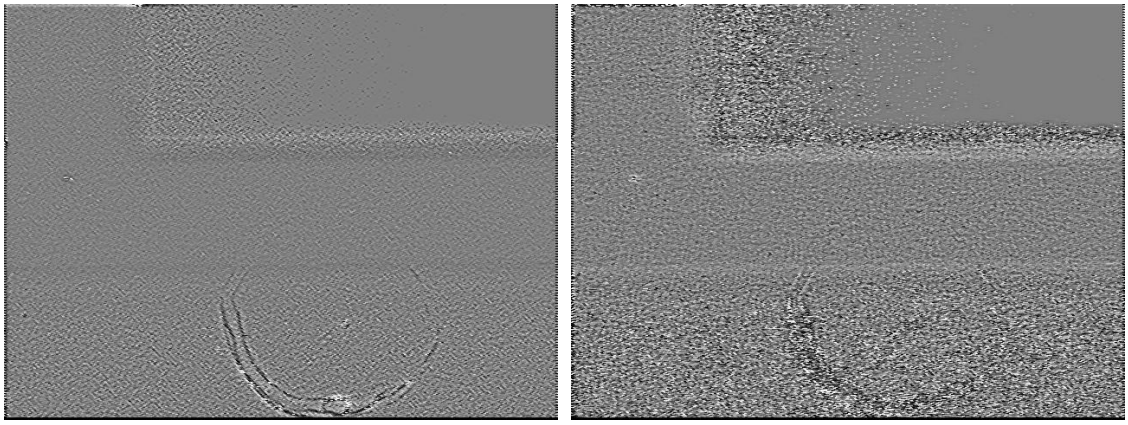


(a)ノイズなし　　(b)ノイズあり

図 14．ラプラシアンフィルタ実行時

このように、ラプラシアンフィルタをかけることで画像のエッジ（輪郭）を抽出することができる。しかしノイズが含まれている右の図では境目があいまいになっており、うまくエッジを抽出することができない。エッジ抽出は画像の先鋭化や、物体検出などに応用することができる。

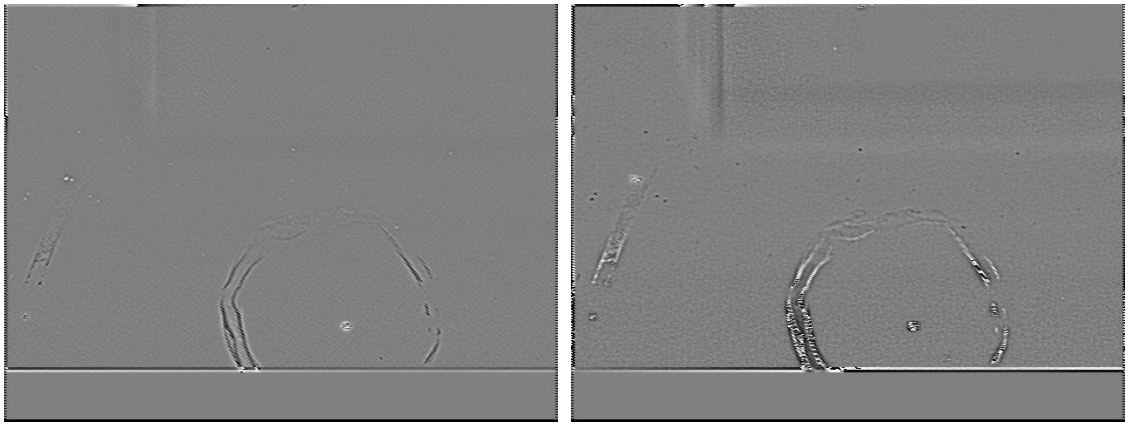
本研究で用いる画像にラプラシアンフィルタをかけると図 15 のようになる。ノイズが多く含まれているため、うまくエッジが抽出できず全体的に滑らかでない。



(a) 4 近傍　　(b)8 近傍

図 15　元画像にラプラシアンフィルタをかけた画像

次に TDI 処理を 60 回行った後にラプラシアンフィルタをかけた画像を図 16 に示す。ノイズの影響が減ったため、元画像に比べると滑らかになった。



(a) 4 近傍　　(b)8 近傍

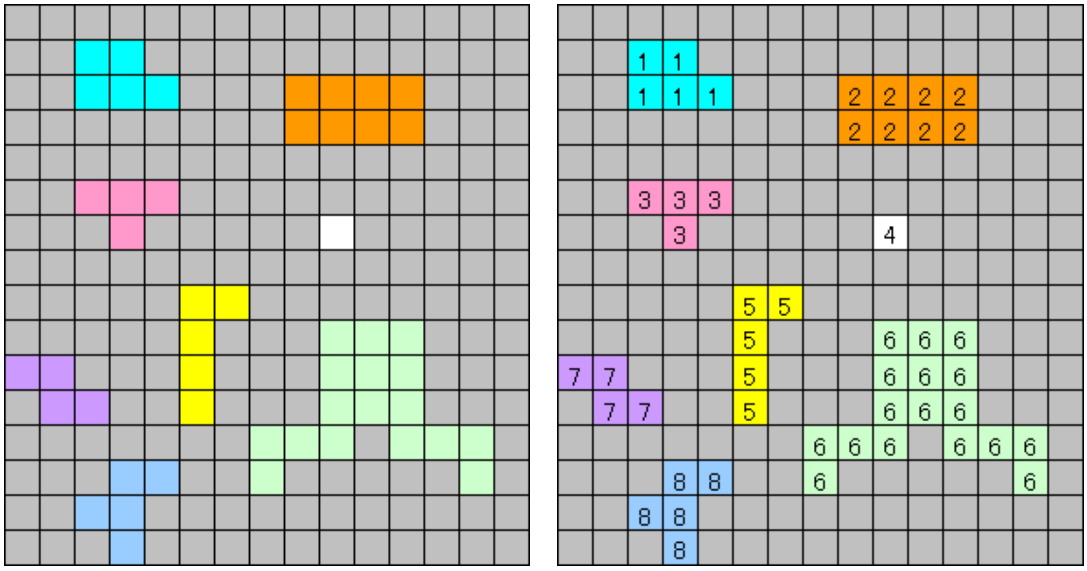
図 16. TDI 後にラプラシアンフィルタをかけた画像

図 15、16 から見てもわかるように、4 近傍よりも 8 近傍フィルタをかけた画像の方が荒い。これは、より明確にエッジを抽出していることを示している。処理時間（計算回数）も同じなため、以降の実験は全て性能の高い 8 近傍のラプラシアンフィルタを用いることとする。

3　ガラス欠損検出画像におけるノイズ検出

3．1　ラベリング処理のアルゴリズム

つながっている全ての画素（連結成分）に同じラベル（番号）を突け、異なった連結成分には異なった番号をつける処理をラベリングという。特徴量抽出などのための前処理に当たる部分である。例えば図 17 のような一つの画像中に連結された物体が 8 個存在する場合、それぞれの物体に対してラベル（番号）をつけていくことになる。



(a)ラベリング前　　(b)ラベリング後

図 17．ラベリングイメージ

ここで、面積が 5 である物体だけを抽出したい。その場合は図 17(b)のラベルが 1，5，8 の画像だけを残し、その他の部分を消してしまうことで図 18 のように実現できる。本研究ではこのラベリングをノイズ検出に用いる。

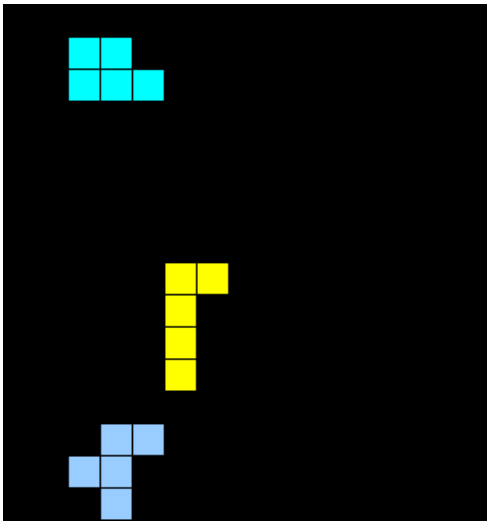
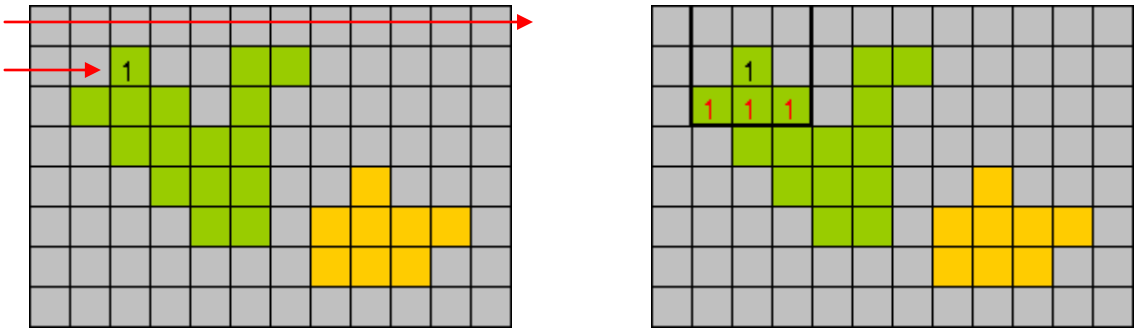


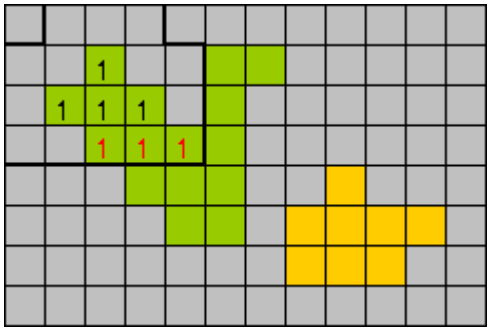
図 18．ラベリングを用いた抽出

次に、例として図 19 の画像を用いてラベリングのアルゴリズムについて説明する。この図の画素値を左上から順に検索していくと、背景（灰色）と違う緑の画素にあたる。これが何らかの物体だと判断し、この画素にラベル「１」をつける（手順 1)。次にラベル「1」をつけた画素の周囲 8 画素（図中の太枠）を検索する。この中にラベルをつけた箇所と同じ画素があれば同じラベルをつけ、それ以外の画素であれば何もしない。図の赤字の部分が新しくつけられたラベルである（手順 2)。以降、連結した画素全てにラベルがつけられるまで同様の処理を繰り返す（手順 3)。全ての画素にラベルを付け終わると、次の連結成分の検索を始める（手順 4)。既にラベルをつけられている画素値を無視して検索を続けていくと、またラベルのつけられていない新たな画素値を発見する（手順 5)。先ほどの緑とは違う値なので、ラベル「2」をつけ、ラベリング処理を行う。画像の最後まで検索を終えるとラベリングの作業は終了となる（手順 6)。

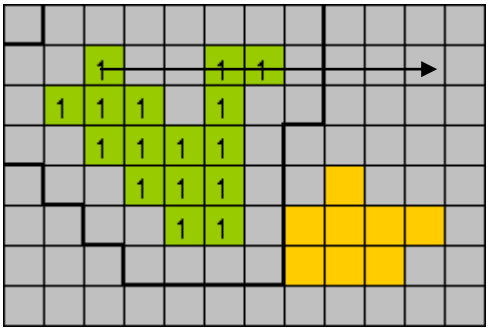


(a)手順 1

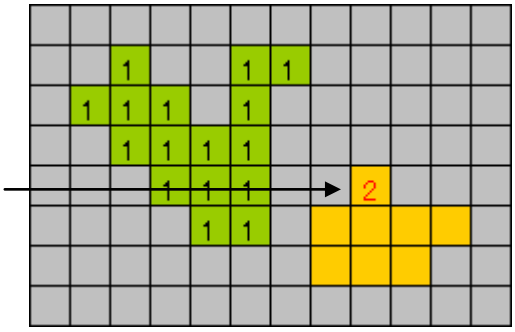
(b)手順 2



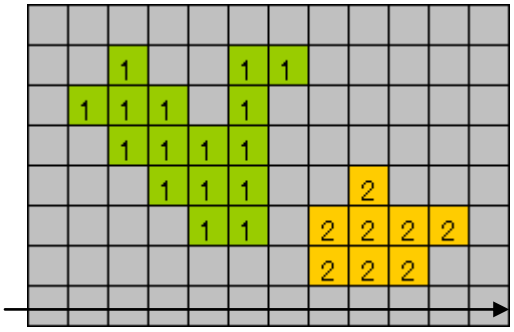
(c)手順 3



(d)手順 4



(e)手順 5



(f)手順 6

図 19 ラベリング処理

3．2 ラベリング処理の実験結果

液晶ガラスの画像にラプラシアンフィルタをかけ、2 値化した画像である。あまりにも多量のノイズが含まれているため、うまくエッジを検出することができない。ラベリングを行っても適切な結果は得られないと予想されるため、TDI 処理によってノイズ除去を行ってから検証を行うこととする。

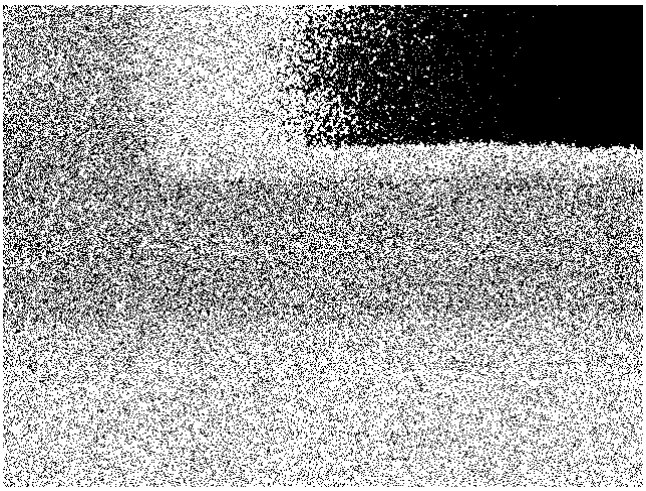
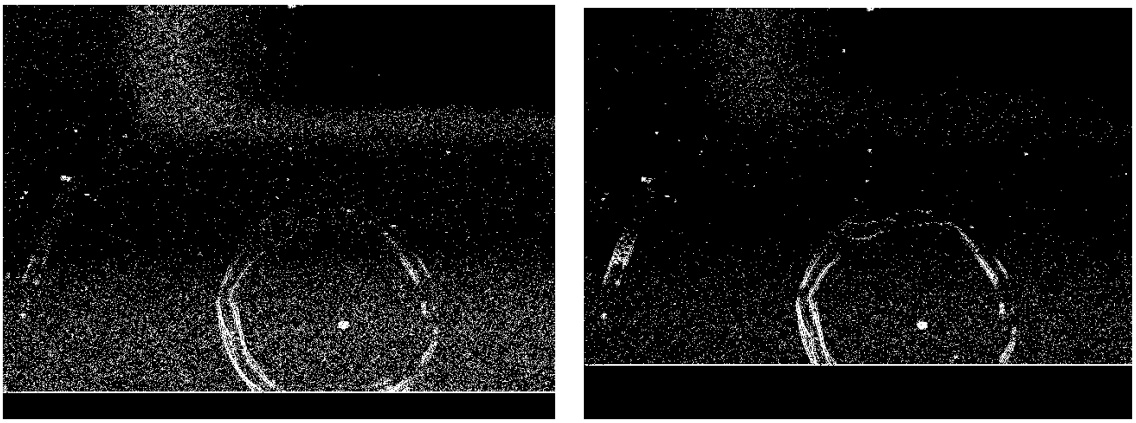


図 20．元画像にラプラシアンフィルタをかけ 2 値化

TDI 処理を 32 回行った画像と 64 回行った画像にラプラシアンフィルタをかけ 2 値化した画像を図 21 に示す。ノイズの影響がかなり少なくなったため、元画像と比較できるくらいになった。TDI だけではわかりにくかったが、ラプラシアンフィルタをかけて 2 値化した後では、処理回数の増加に伴ってノイズが減少していることがはっきりとわかる。



(a)TDI32 回　　(b)TDI64 回

図 21．TDI 後にラプラシアンフィルタをかけ 2 値化

3．3 考察

図 9～11 において TDI による画像のノイズ除去の効果は確認できたが、TDI32 回、64 回、128 回の間では違いが確認できなかったため、TDI は 32 回で十分ではないかということを述べた。しかし、ラプラシアンフィルタをかけてエッジを抽出し、それを 2 値化することによって TDI32 回と 64 回の間での違いを確認できた。つまり、TDI32 回の段階では目で見えないほどのノイズがまだ残っていたということが言えるため、十分な回数とは言えない。

そこで、TDI の各回数におけるノイズの量をまとめたグラフが図 22 である。2 値化した画像にラベリングを行い、そのラベルの数をノイズの量とした。TDI 回数を増やせば増やすほどノイズ（図における白い点）の量は減っていくことがわかる。TDI 回数の少ない段階の方がラベルの数は少ないという結果が出ているが、元々ラベリングとは連結成分にラベルを振り分ける処理である。TDI 処理を行っていない図 20 を見ればわかるように、多量のノイズ同士が連結しているため、ラベルの数がノイズの量に直結しているとは言えない。

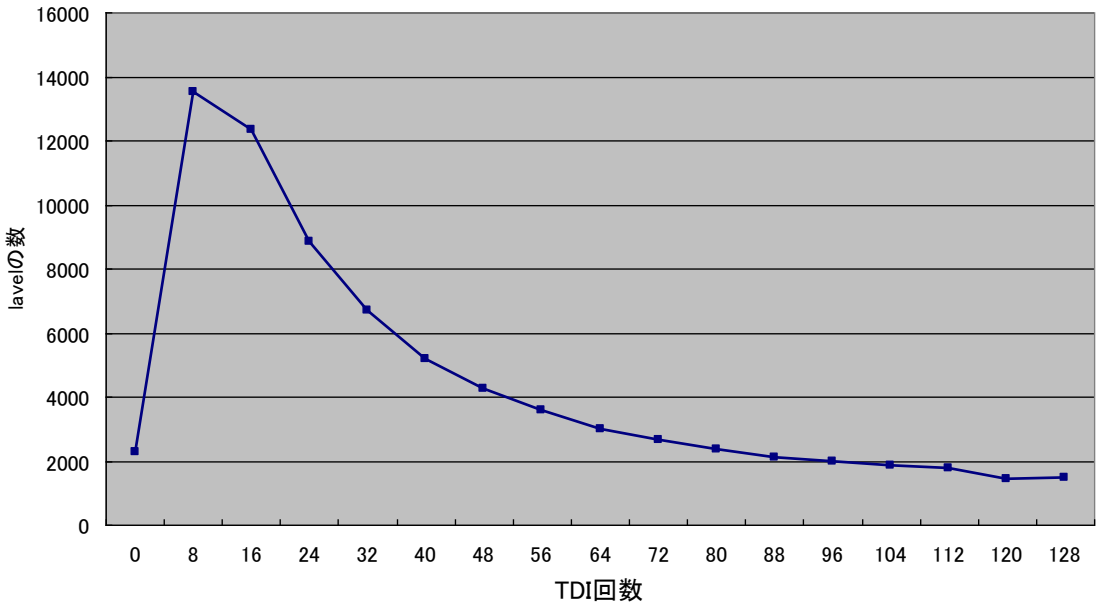


図 22．TDI 回数とラベルの数の関係

4.ハードウェア記述

4. 1 FPGA(Field-Programmable Gate Array)とは

FPGA ボードは製造後に購入者や設計者が回路を記述し、用途に合わせて動作させることができるボードである。また、回路の書き換えも容易に行うことができるため、アップデートが頻繁に行われるものを中心に幅広い分野で用いられている。FPGA は通常、アプリケーションの仕様に合わせて高速に実行するように設計されているため、効率よく、また、専用回路は並列動作が可能なので、非常に高速に処理することが可能である。

本研究ではこれまでに示した TDI、ラプラシアンフィルタ、ラベリングの 3 つのプログラムをシミュレーションによってコンピュータ上で動作の確認を行った後、FPGA ボードへの実装によるプログラムの高速化を検証していく。回路記述には Verilog-HDL という言語を用いる。本研究で作成する欠損検出画像処理回路の全体構成を図 23 に示す。

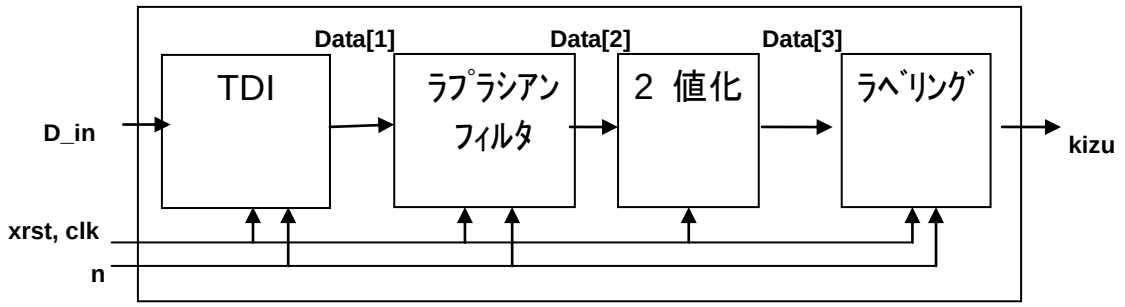


図 23：画像処理回路ブロック図

まずは画像データを入力し、TDI 処理を行い画像のノイズを除去していく。次にラプラシアンフィルタをかけエッジを抽出し、2 値化によってラベリングをしやすくする。最後にラベリングを行い、欠損の有無を判定し、結果を出力する。図 21 を見てもわかるように、ノイズは 1 画素単位で混入するため、面積は 1 ピクセルである。しかし、ガラスの傷や気泡などの欠損部分はそれなりの大きさを持っている。そのため、過去のデータから予想される欠損の面積の情報を予め与えておき、その面積に一致するラベルがあるかどうかを判定することで、欠損の有無を検出することができると考えられる。また、この実験では 64*64 の画像を用いる。

4．2 各画像処理の回路設計

4.2.1　TDI モジュール

TDI モジュール内部処理のアルゴリズムを図 24 に示す。まず、D_in から入力された画像データをメモリに格納する。次に 2 枚目の画像データを入力し、画像の下部 1 ピクセル分、すなわち 64 番地分を飛ばして各番地のデータを加算していく。この処理を n から入力されたデータの回数だけ繰り返す。また、図 9～11 と同じように、（64*n）の番地に 0 を代入する。最後に各番地のデータをそれぞれ n で割り、data1 として出力する。

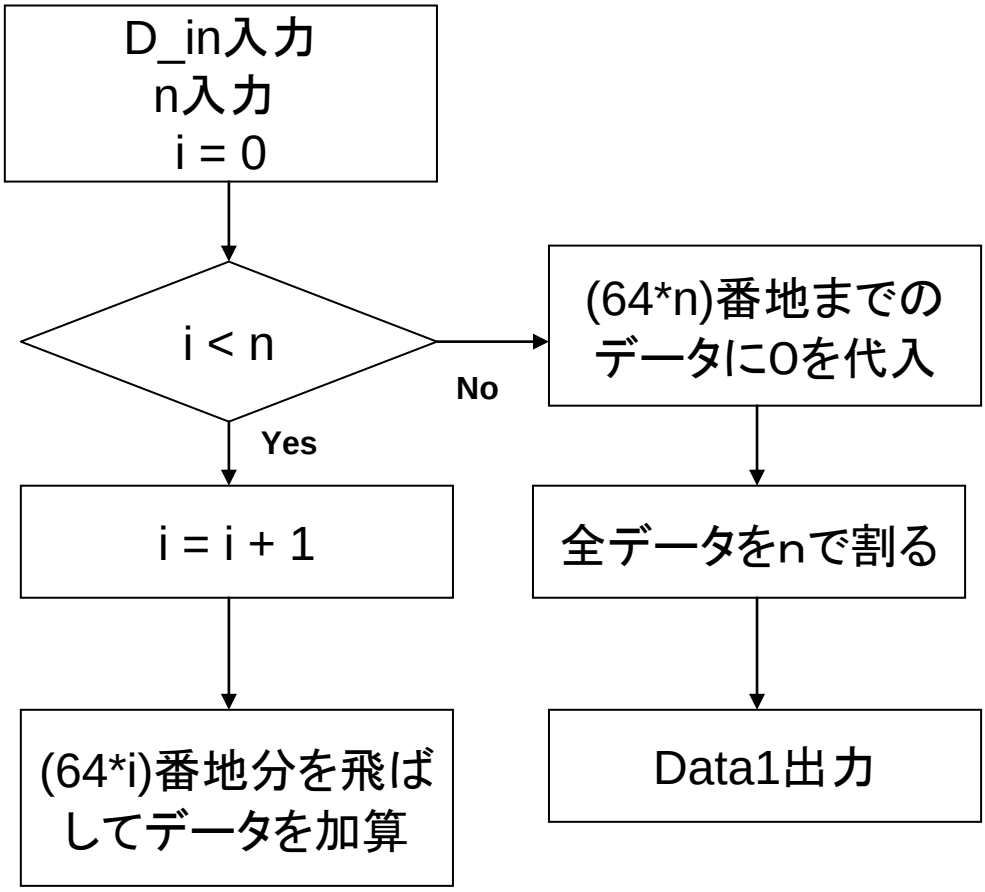


図 24. TDI モジュールのアルゴリズム

4.2.2　ラプラシアンフィルタモジュール

TDI モジュールから出力されたデータを入力として受け取り、その画像データにラプラシアンフィルタ演算を行うモジュールである。フィルタの性質上、画像の端は黒(0)が代入されるため、最初と最後の 64 画素には 0 を代入してある。モジュールのアルゴリズムを図 25 に示す。

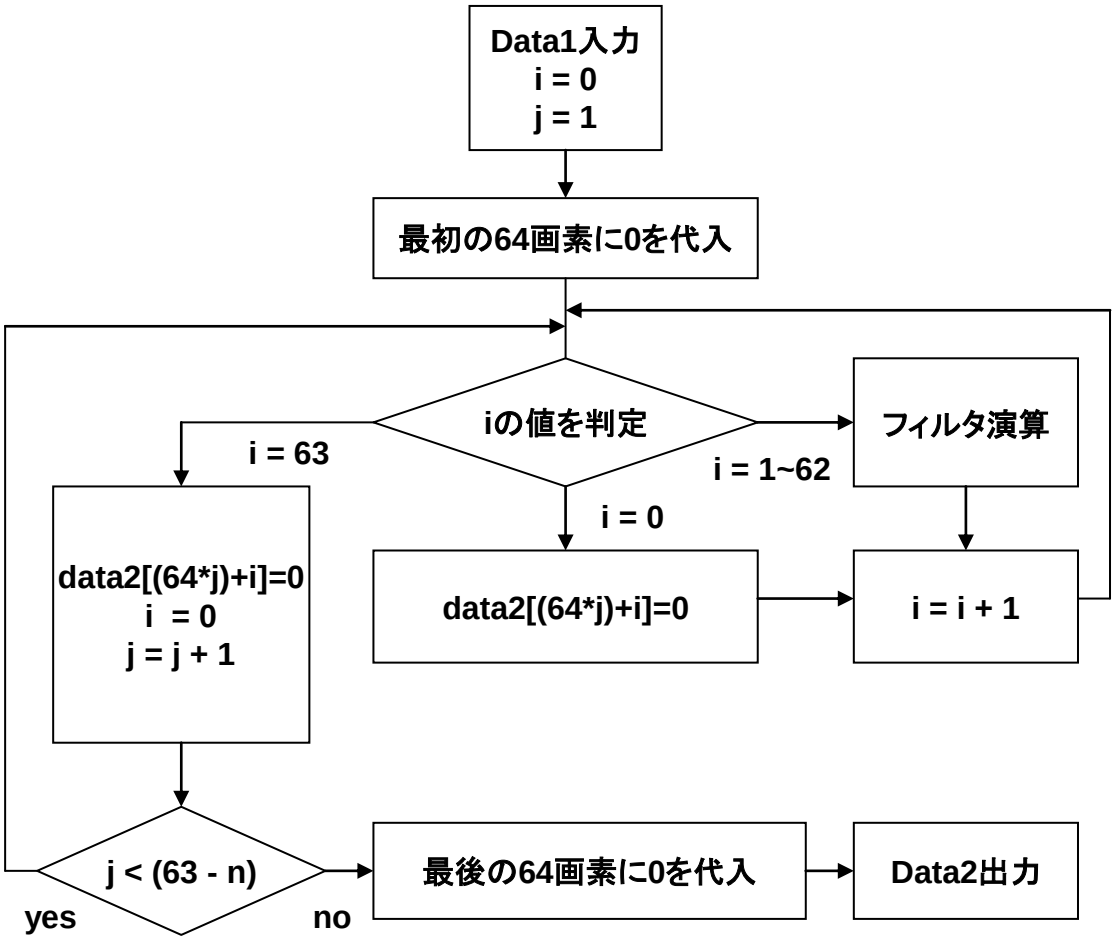


図 25. ラプラシアンフィルタモジュールのアルゴリズム

4.3.3. 2 値化モジュール

ラプラシアンフィルタモジュールから出力された Data2 を入力として取り込み、画像を 2 値化するモジュールである。画素値が 127 以下であれば 0 を、128 以上であれば 255 を Data[3]として出力する。図 26 に 2 値化モジュールのアルゴリズムを示す。

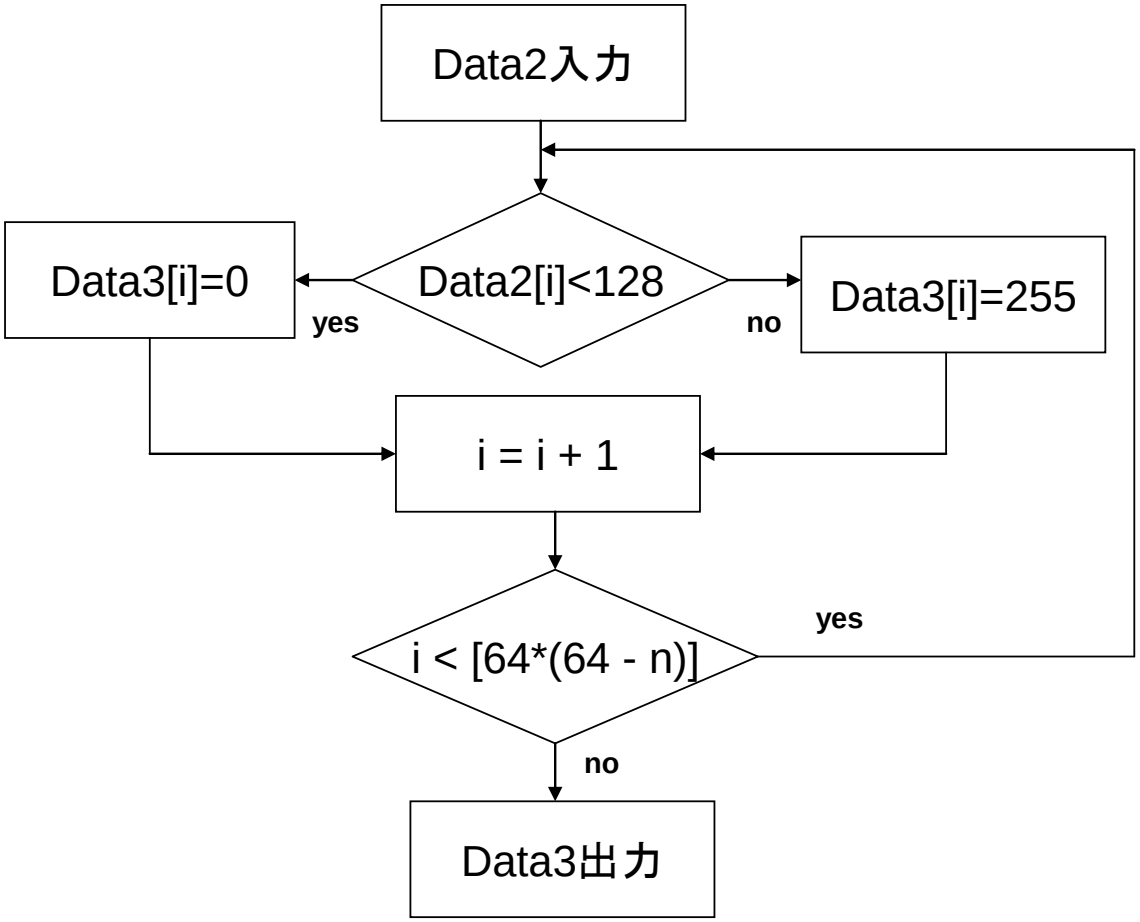


図 26. 2 値化モジュールのアルゴリズム

4.4.4　ラベリングモジュール

データは 2 値化されているため、値が 255（白）である画素にラベリングをしていくモジュールである。ラベルをつけられた画素の周囲 8 方向を検索し、隣接している画素を見つけてラベルをつける、拡張領域法という手法を用いる。図 27 にラベリングモジュールのアルゴリズムを示す。

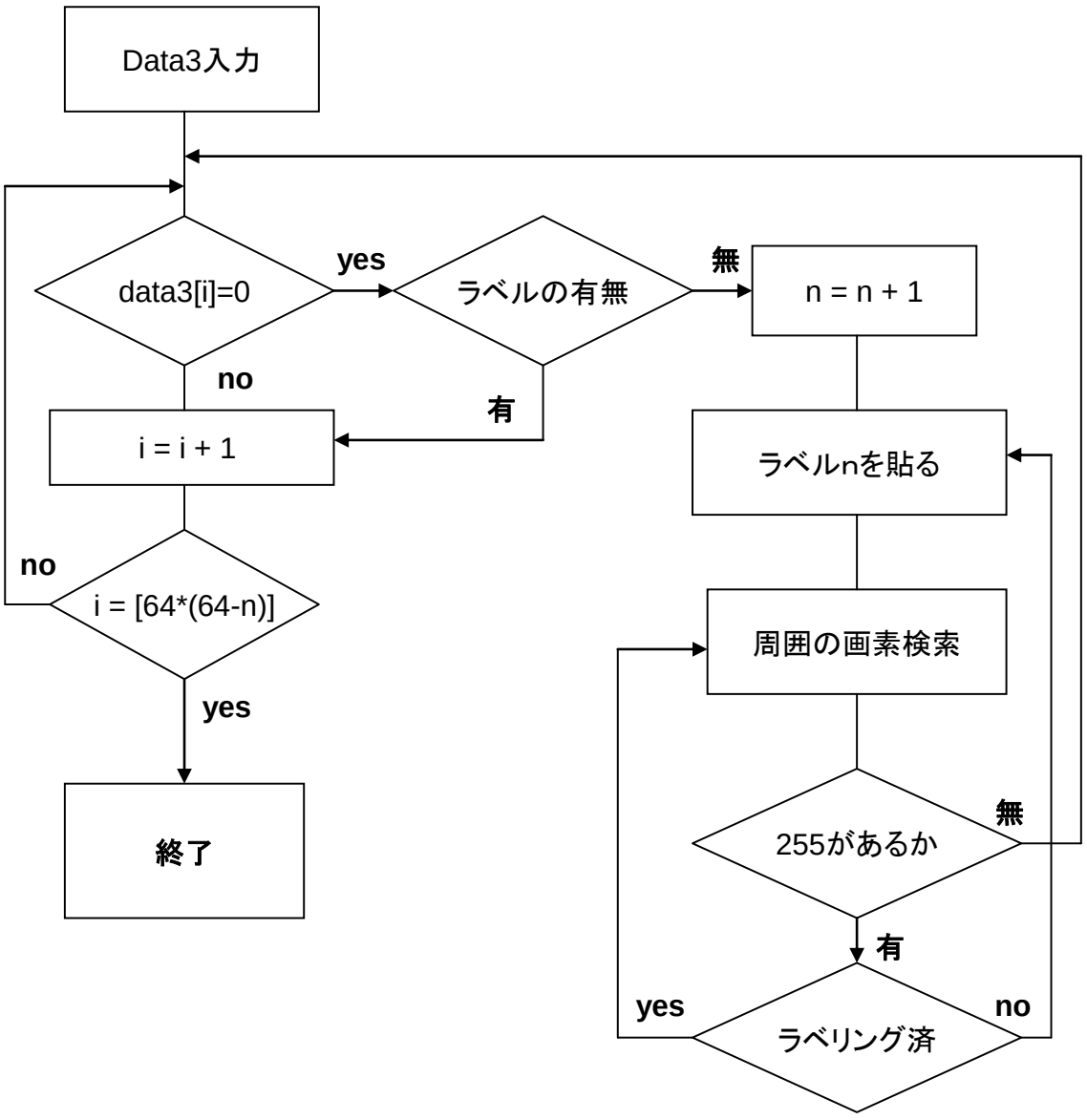


図 27. ラベリングモジュールのアルゴリズム

おわりに

本論文では TDI による画像データのノイズ除去、ラプラシアンフィルタとラベリングによるノイズ除去効果の検証、及びそれらの画像処理プログラムの FPGA ボード上への実装による高速化の検討を行った。TDI は回数を重ねるほどにノイズ除去の効果は大きくなる反面、画像そのものが小さくなってしまうという面もある。しかし、少しの回数でも大幅な画質改善が見込めるため、画像のノイズ除去としては非常に有効な手段であると言える。

今後の課題としては、これらのプログラムを Verilog-HDL 記述による FPGA ボードへの実装・検証が挙げられる。特にラベリングに関しては処理時間が大きいため、FPGA ボード実装によって処理時間の大幅な短縮が期待できる。

謝辞

本研究の機会を与えてくださり、ご指導を頂きました山﨑勝弘教授に深く感謝いたします。また、本研究に関して様々な相談に乗って頂き、貴重な助言を頂いた PISHVA JOHN CYRUS P 氏をはじめ、様々な面でご協力を頂いた高性能計算研究室の皆様に心より感謝いたします。

参考文献

[1]井上誠喜、八木伸行、林正樹、中須英輔、三谷公二、奥井誠人：C 言語で学ぶ実践画像処理、オーム社、2010 年

[2]小林優：改訂　入門 VerilogHDL 記述、CQ 出版、2009 年

[3]松崎 裕樹:マハラノビス距離を用いた画像判別とラベリングの高速化の実現
立命館大学理工学部卒業論文 2006 年

[4] 乾 圭佑:画像処理アルゴリズムのハード/ソフト最適分割の検討
立命館大学理工学部卒業論文 2009 年

[5] 大山 佳宣:OpenMP を用いた画像処理プログラムの並列化
立命館大学理工学部卒業論文 2010 年