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(I)

氏名：天井 康雄
学籍番号：2260070063-8
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 と共同研究の一環として Time Delay Integration(TDI)の効果を検証する。

株式会社 KDE から提供された英語論文の VGA(640×480)画像 687 枚を対象に、C 言語で作成した TDI、ラプラシアンフィルタ、2 値化、ラベリングのプログラムで実験を行った。パラメータ調整としては TDI で読み込ませる画像枚数 2 値化する際の背景幅の設定がある。

実験結果から、TDI の有効性が確認できた。

目次

図目次

表目次

1. はじめに

近年、スマートフォンの流行により、携帯電話や PC に用いられるフラットパネルディスプレイの需要が高まっている。フラットパネルディスプレイとは、筐体が板状で画面が平面になっているディスプレイ機器のことである。省電力など副次的なメリットを備える方式もあり、様々な方式が研究されている。製造の過程において、液晶用ガラスの傷や欠陥の検出は非常に大事であるということがいえる。傷や欠損の検出の速度・精度が向上すれば、液晶用ガラスの生産量が向上し、それらに関わる機器の生産量も向上する。

本研究では、液晶用ガラスの欠損検出画像処理の FPGA を用いた高速化を目的としている。FPGA とは、自分で論理回路を作成し、書き換え可能な集積回路である。また、作成した論理回路に合わせて高速に動作するようになっており、C プログラムを逐次で動作させるよりも処理時間が短くなると考えられる。FPGA は VHDL や Verilog-HDL といったハードウェア言語で設定しなければならなかったが、近年では C 言語といった高級言語による開発ツールも登場し、価格も安価になってきていることから、開発のハードルは低くなってきている。

ガラスの製造過程において生じる泡・気泡を判定する C プログラムを実装し、判定精度を向上させる。そこで画像中の背景のノイズを軽減させる TDI を実装した。TDI は、1 行分の画素をずらしながら複数枚の画像を読み込み、各画素の平均値を計算しノイズを小さくするというものである。その TDI を行った場合と行わなかった場合とを比較し、判定精度が向上するかを検証する。ラプラシアンフィルタで画像中の輪郭を強調する。判別するための指標である特徴量を抽出するためにラベリングも実装した。

判別の対象となる画像データは KDE より頂いた英語論文の画像で、687 枚で画像サイズは 640×480 ピクセルのグレースケール bmp ファイルである。実験手順として、まず TDI 処理を行った画像と行わなかった画像を用意する。そしてラプラシアンフィルタとラベリングを行う。ラベリングは 2 値化画像を作成し、それにラベリングしていく。2 値化画像を作成する際に物体領域と背景領域を手動で入力する。

本論文では、2 章で TDI、ラプラシアンフィルタ、2 値化、ラベリングの 4 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、3 章では C 言語で記述した画像処理プログラムの実行結果を表記する。4 章では Verilog-HDL プログラムでの画像処理全体と TDI、ラプラシアンフィルタ、ラベリングの 3 つのモジュールの動作処理内容について述べる。5 章では考察を述べる。

2. ガラス欠陥検出画像処理のためのアルゴリズム

2.1 TDI

TDI とは Time Delay Integration の略で、画素をずらして同じ物体を繰り返し撮影し、その複数枚の写真の平均画像を出力するものである。TDI を行うことによって、写真の背景のノイズが減を減らすことが出来る。TDI で用いる画像が多ければ多いほどノイズが減るというメリットがあるが、その分画像は小さくなるといったデメリットもある。

TDI を行うイメージ図を図 1 に示す。

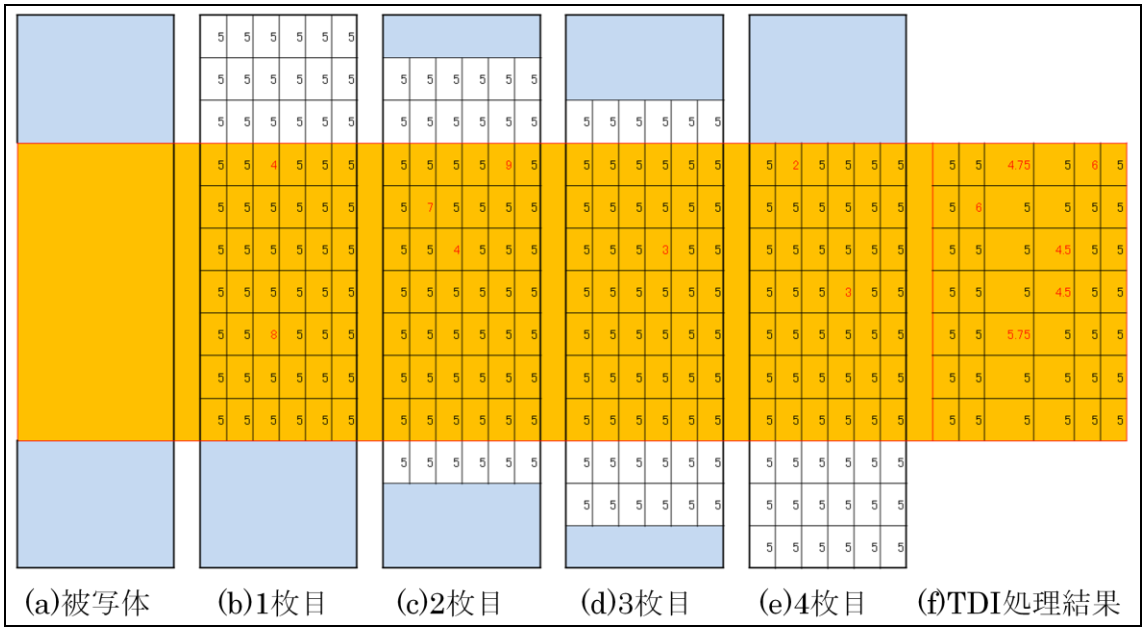

図 1：TDI 実行のイメージ

図 1(a)は被写体である。それを(b), (c), (d), (e)のように、1 行ずつずらして撮影していく。撮影したら、画像の値は全て 5 となるのが正常だとする。また、(b), (c), (d), (e)内の赤で表された数字は、写真内のノイズと考えてもらいたい。(f)は TDI 処理の結果である。(b), (c), (d), (e)で共通する黄色い部分が出力範囲となる。今回は(b), (c), (d), (e)の 4 枚の画像で平均を取った。赤字で表記されている個所が 4 枚の中にノイズがあった箇所となる。平均を取ったことによって(b), (c), (d), (e)のノイズと比べ、正常値である 5 に近くなっていることが分かる。

2.2 ラプラシアンフィルタ

ラプラシアンフィルタは画像中の輪郭(エッジ)を強調するフィルタである。ラプラシアン ∇^2は画像のような 2 次元のデータに 2 階微分を行う。

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

これを差分形式で表すと次のようになる。

$$\nabla^2 f(\mathrm{x},\mathrm{y}) = f(\mathrm{x}-1,\mathrm{y}) + 2f(\mathrm{x},\mathrm{y}) + f(\mathrm{x}+1,\mathrm{y}) + f(\mathrm{x},\mathrm{y}-1) - 2f(\mathrm{x},\mathrm{y}) + f(\mathrm{x},\mathrm{y}+1)$$
$$= f(\mathrm{x}-1,\mathrm{y}) + f(\mathrm{x}+1,\mathrm{y}) + f(\mathrm{x},\mathrm{y}-1) + f(\mathrm{x},\mathrm{y}+1) - 4f(\mathrm{x},\mathrm{y})$$

これらを係数で表すと

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

よなり、これにさらに 45℃方向を含める場合、

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

となる。これらの係数をマスクパターンとし、注目画素と周囲 8 画素を微分していく。

2.3 2 値化

2 値化は物体領域の抽出を目的に行う。また、ラベリングを実行する際に、輝度値が 0 か 255 かを判別するだけになるので、処理が簡単になることが期待できる。画像の輝度の平均値を調べ、背景幅の範囲を決定する。決定した背景幅の範囲を T とする。画像中の画素を 1 つずつ走査してゆき、画素の輝度値が T に収まらなければ白に、収まれば黒にしていく。2 値化のフローチャートを図 2 に示す。

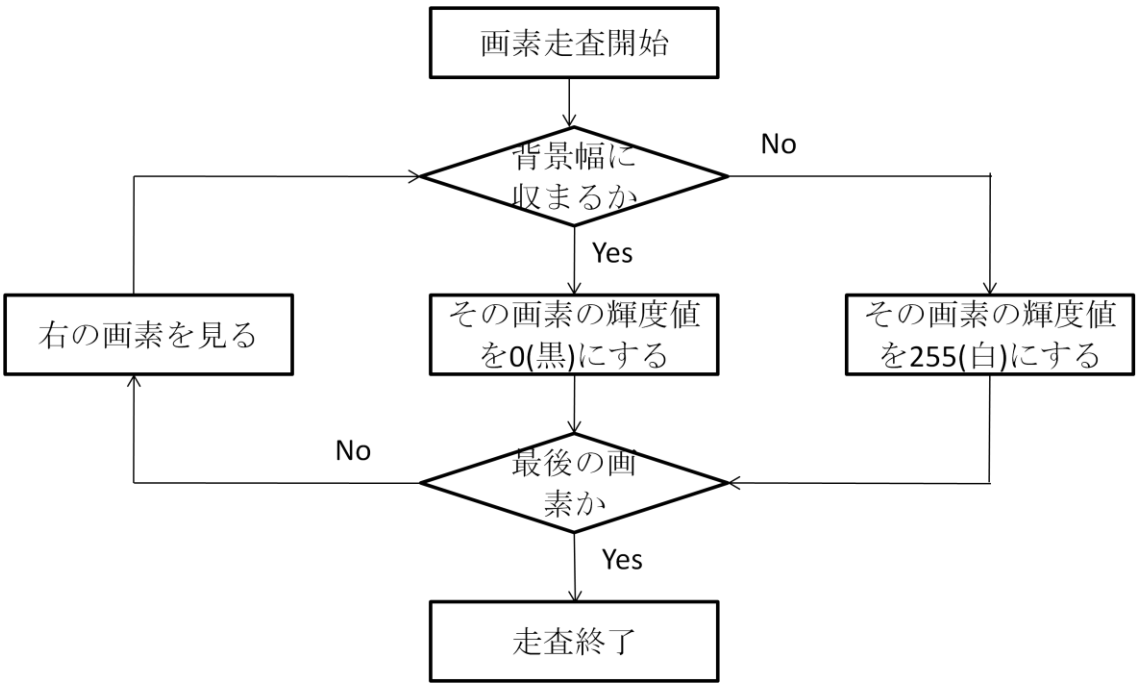

図 2： 2 値化のフローチャート

2.4 ラベリング

ラベリングとは、画像中の物体を検出する処理のことである。物体の条件(例えば画像中の輝度値の値など)と一致する画素に注目しラベルを貼る。そして周囲 8 画素のどれかと連結している場合は、注目した画素と同じラベルを貼る。連結している全ての画素にラベルを張り終わったら、新しいラベルを作り、また条件に合う画素を走査する。最後の画素まで走査したときに終了する。こうすることによって、画像中に物体がいくつあるかが分かるようになる。ラベリングを実行したときのイメージを図 3 に示す。

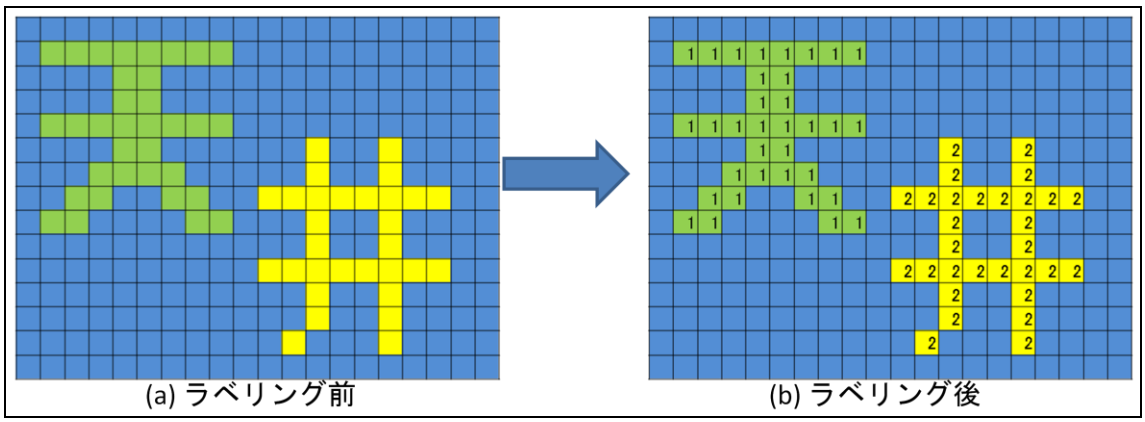

図 3：ラベリング処理のイメージ

今回使用する画像は、ラベリングの前に 2 値化しているので、背景の輝度値 0(黒)と文字の輝度値 255(白)を判別し、輝度値 255(白)にラベルを貼っていけば良い。
フローチャートを図 4 に示す。

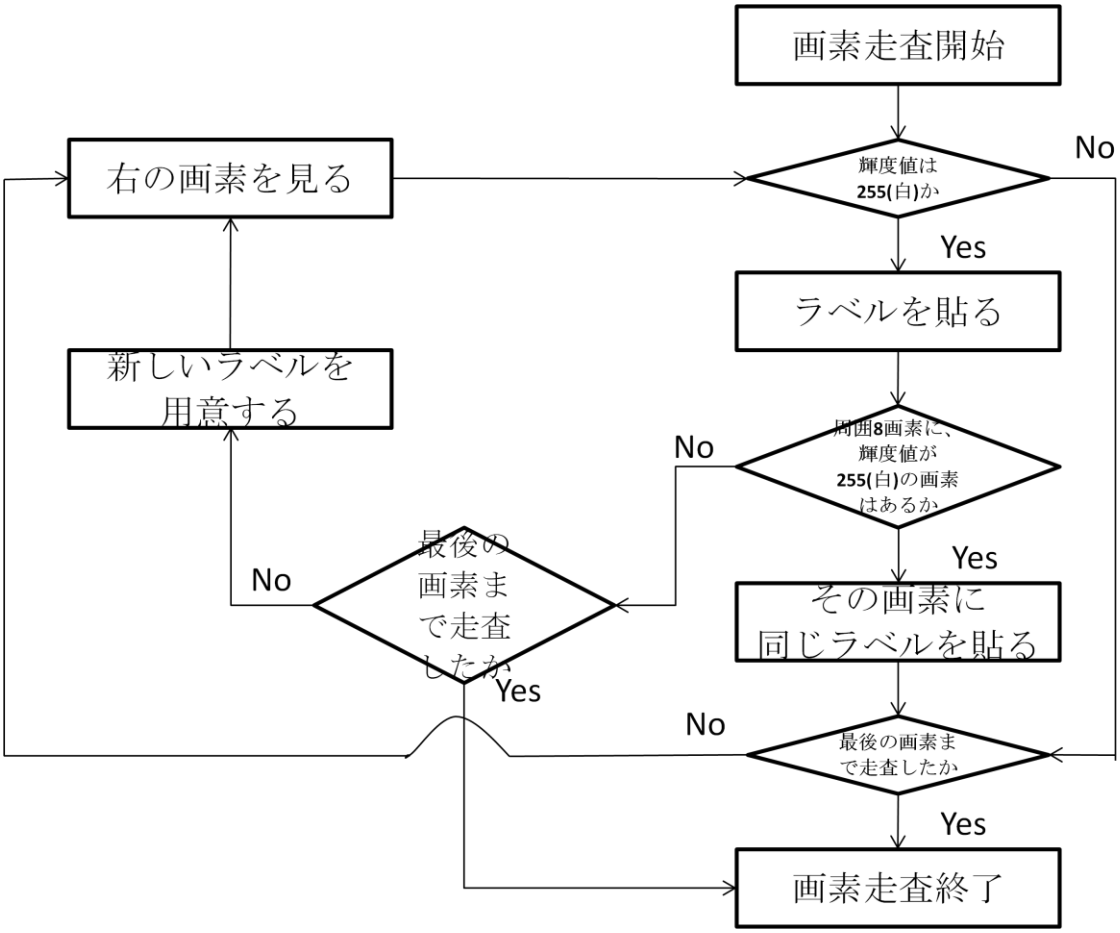

図 4：ラベリングのフローチャート

3. C 言語プログラムでのガラス欠陥検出画像処理の実行結果

3.1 TDI

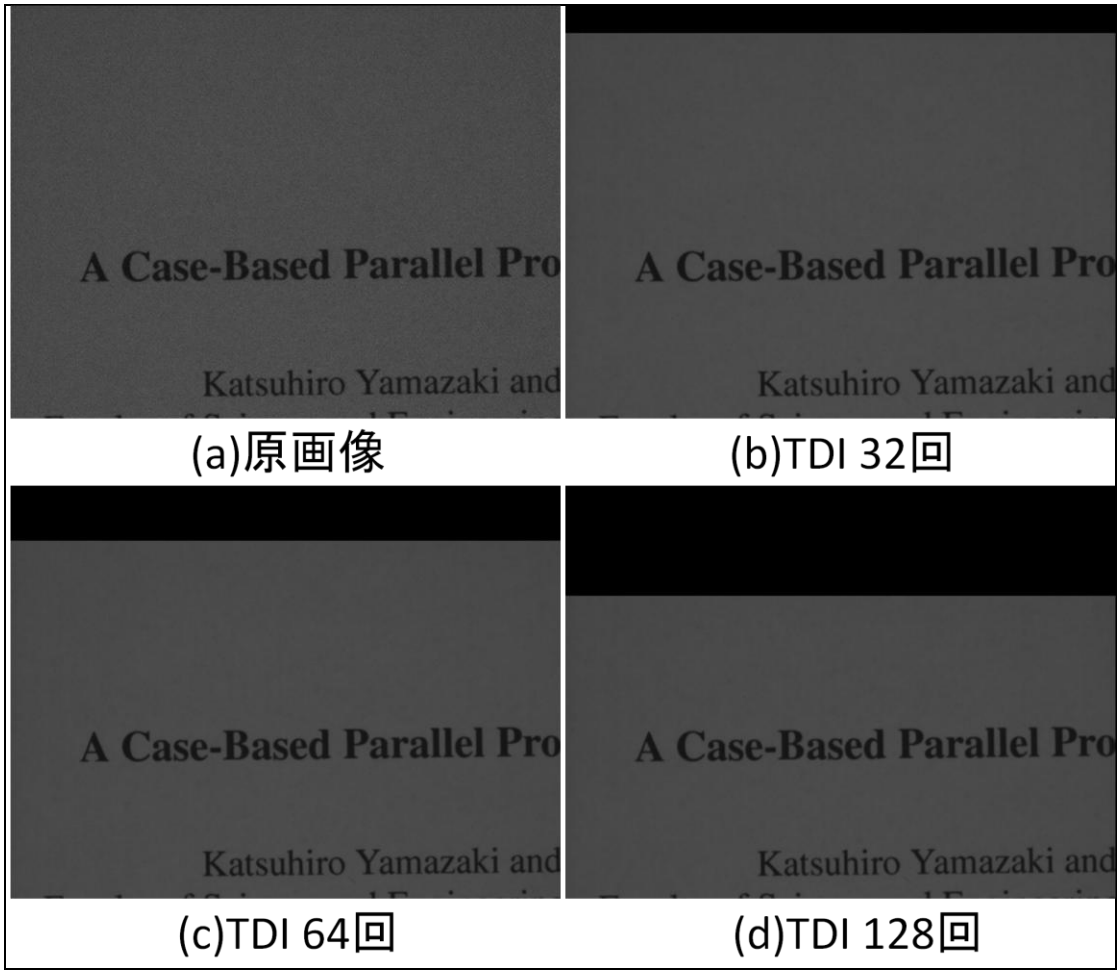

図 5：原画像と TDI 処理を行ったものの比較

図 5(a)は TDI 処理する前の、開始番号 000 の原画像である。000 から 031 の 32 枚、000 から 063 までの 64 枚、000 から 127 までの 128 枚で TDI 処理を行う。図 5(a)の原画像をみると、背景にノイズがあることが目に見えて分かる。

図 5(b)が 32 回、図 5(c)が 64 回、図 5(d)が 128 回 TDI を行ったものである。回数が多ければ多いほど、背景のノイズは減っているが、判別に使える画像の領域は小さくなる。

3.2 ラプラシアンフィルタ

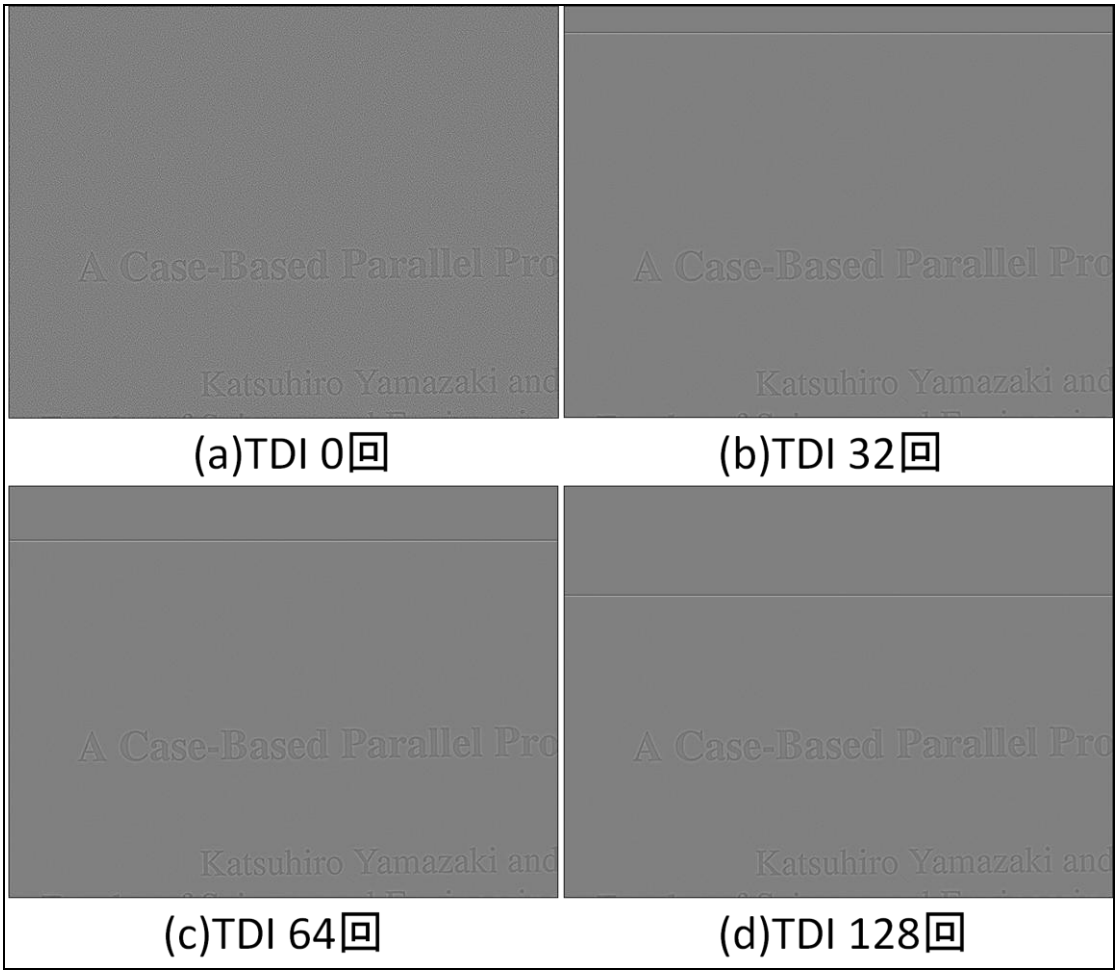

図 6：4 近傍ラプラシアンフィルタ処理を行ったもの

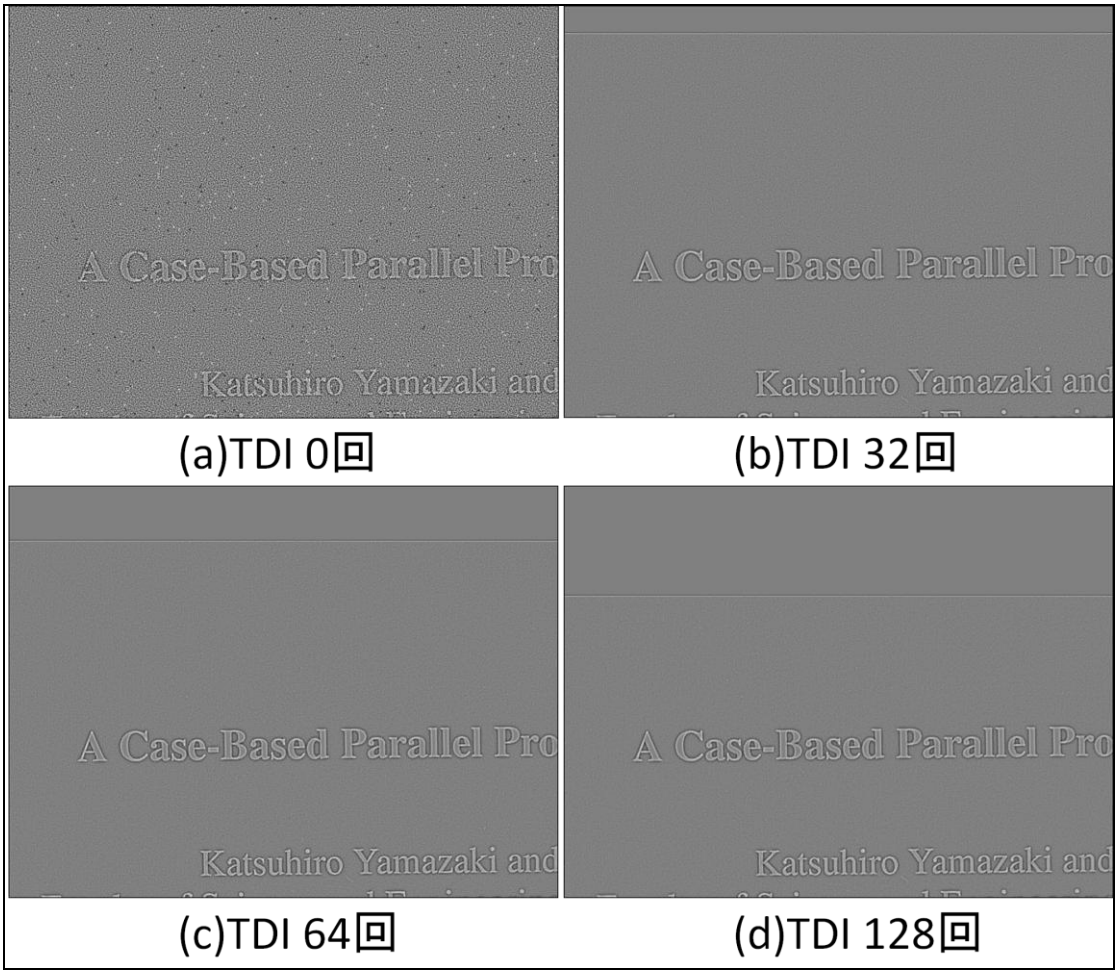

図 7：8 近傍ラプラシアンフィルタ処理を行ったもの

図 6 と図 7 を見た限りでは、TDI の回数は何回が良いかはわからない。しかし、TDI を行わなかったことによって、図 7 の(a)では画像の文字だけではなく、背景のノイズが強調されていることがわかる。図 6 の(a)では背景にノイズがあまり強調されていないが、それは 4 近傍のラプラシアンフィルタは横軸と縦軸の計算しか行わないため、強調される量も減るためだと考えられる。この結果から、TDI は少なからずとも効果があると考えられる。

3.3 ラベリング

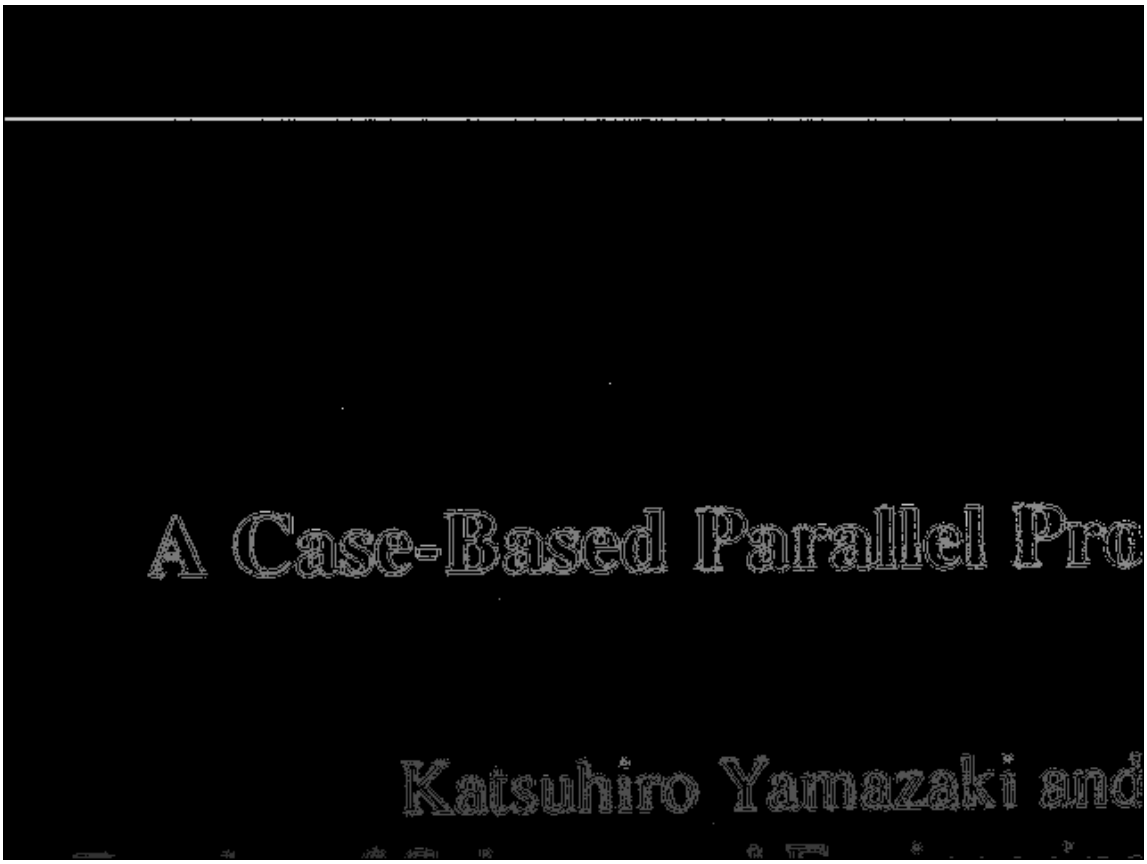

図 8 : TDI64 回、8 近傍ラプラシアンフィルタ処理しラベリングしたもの(グレースケール)

図 8 は、TDI の回数を 64 回、8 近傍フィルタで処理し、ラベリング処理を行った画像である。ラベリング前に行う 2 値化処理において、背景幅は 100~160 で行った。100~160 に設定した理由は、ラプラシアンフィルタで輝度値 128 を基準に処理するため、画像の平均値が 128(≒130)前後になり、背景は 100~160 の範囲内に収まると考えられるからである。図 8 のように、グレースケールではラベリングの効果が分かりづらいため、図 9、図 10 にラベリング処理をカラーで出力した画像を載せることにする。

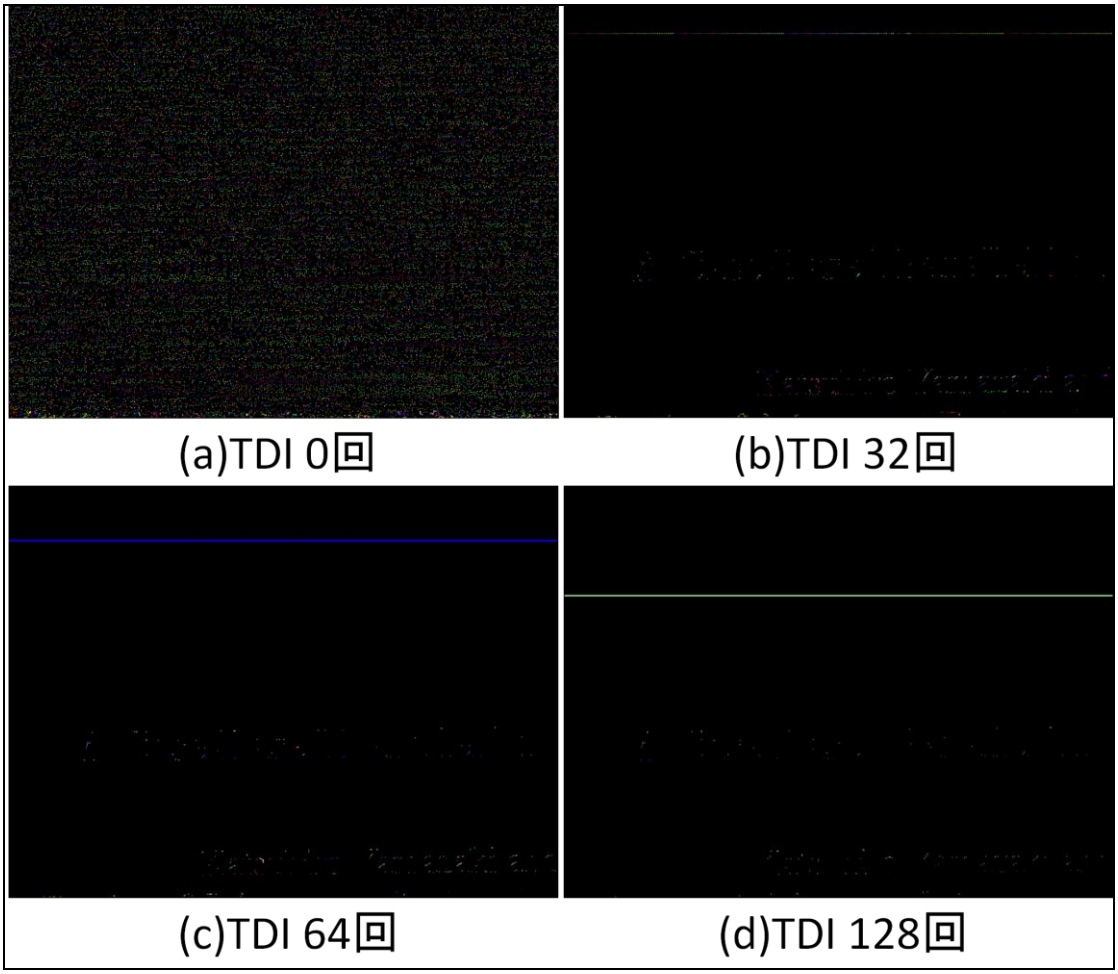

図 9：4 近傍ラプラシアンフィルタで処理しラベリングしたもの(カラー)

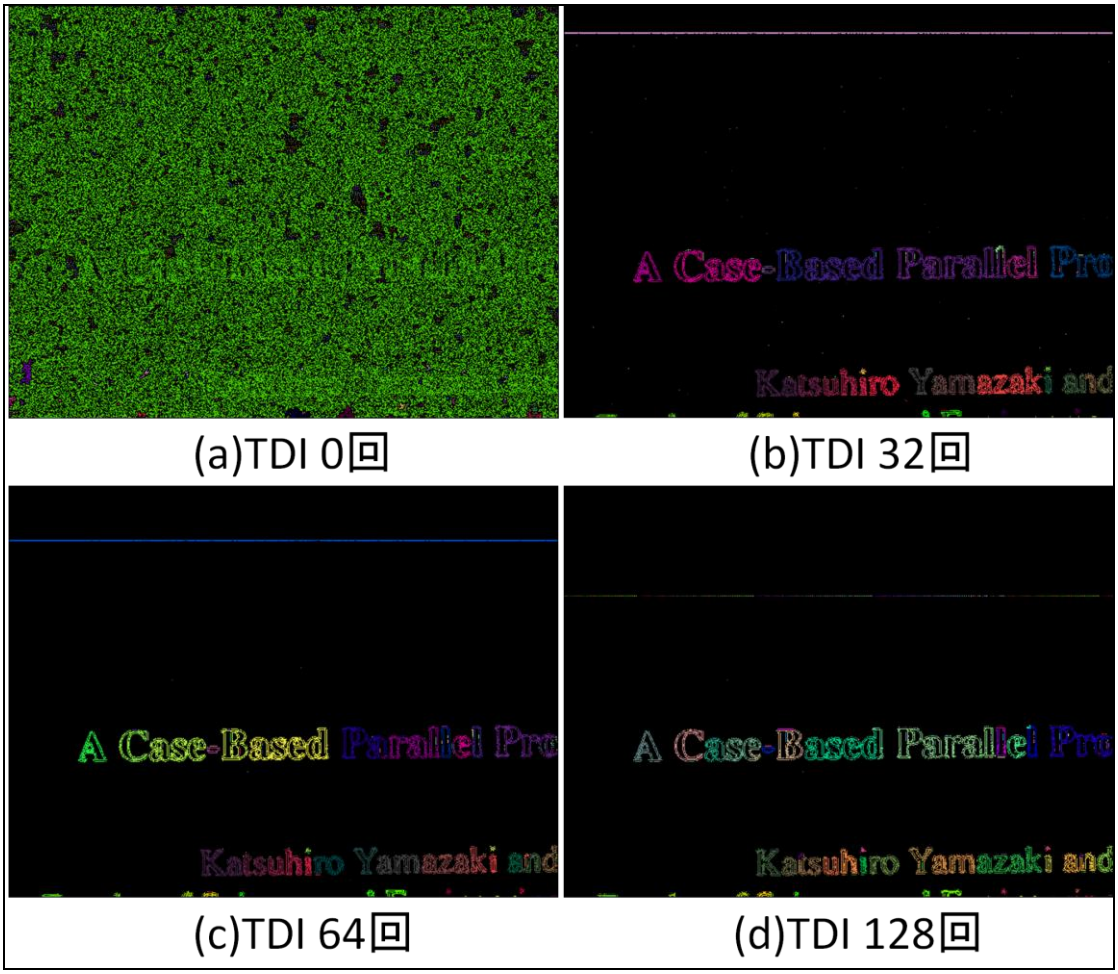

図 10：8 近傍ラプラシアンフィルタで処理しラベリングしたもの(カラー)

図 9、図 10 を見ると、ラプラシアンフィルタは 8 近傍で行った方が良いことが分かる。また、TDI の回数は図 10 の(b)を見ると、まだ背景にノイズが多く、図 10(c)、図 10(d)を見るとノイズは殆どないことが分かる。TDI の回数は多いほど背景のノイズが減るが、その分判別に使用できる画像の領域は狭まる。よって、TDI の適正回数を探さなければならない。そこで、TDI と 8 近傍ラプラシアンフィルタの処理済みの画像を用意し、貼られるラベルの枚数とラベリングにかかる処理時間を調べることにした。実験に用いた PC の CPU は Core2Duo3.16GHz、RAM は 4GB である。実験結果を表 1 に示す。

表 1：TDI の回数に伴うラベルの枚数と処理時間の変化

TDI(回数)	ラベル(個)	処理時間(sec)
0	3235	2.74
32	181	0.57
64	159	0.51

128	535	0.84

表 1 より、今回の実験で用いた画像では TDI 回数は 64 回前後が適切であると考えられる。また、TDI を行うことによってラベルの枚数も減ることも、肉眼ではなく数値で確認することができた。

3.4　ガラス画像

3.3 から、TDI の有効性が確認できた。そこで、ガラスの画像で欠陥を検出できるか実験する。図 11 にガラスの原画像と、検出したい欠陥を赤い円で囲み表示した。ガラス画像を対象に画像処理プログラムを実行した結果は図 12 となる。

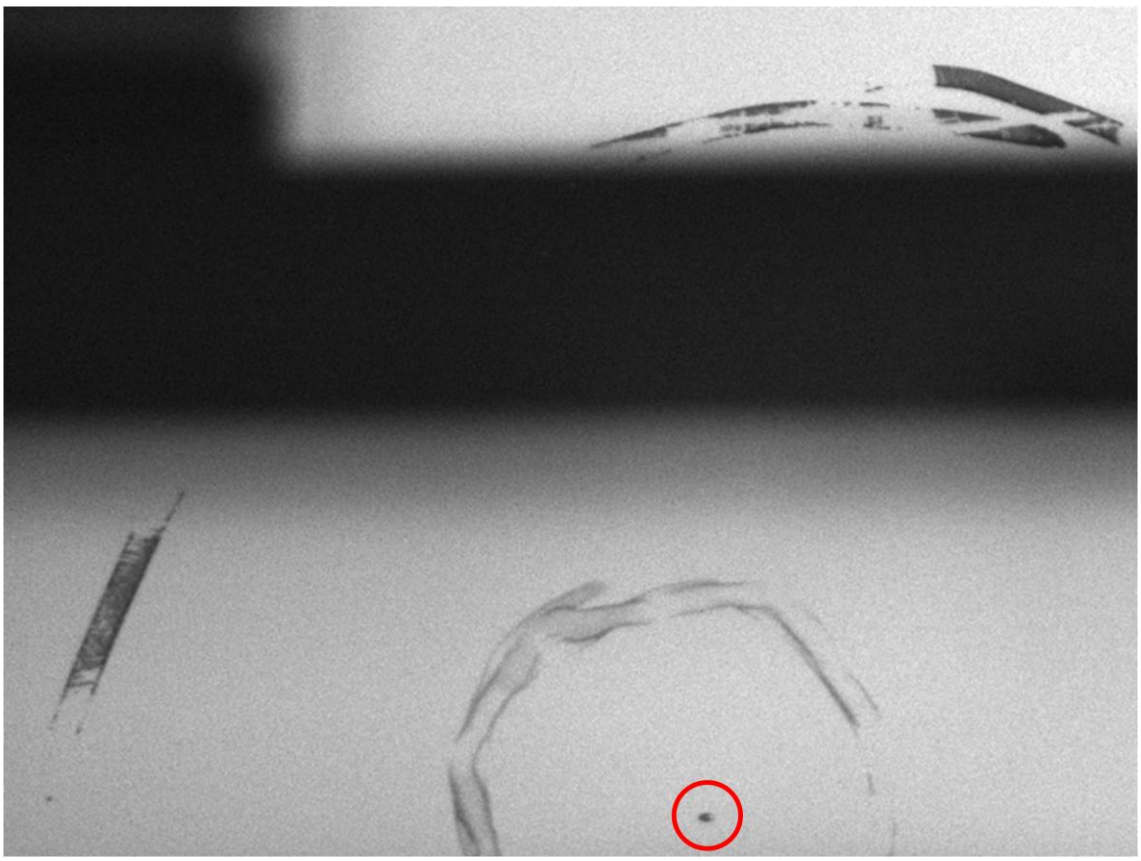

図 11：ガラスの原画像と検出したい欠陥

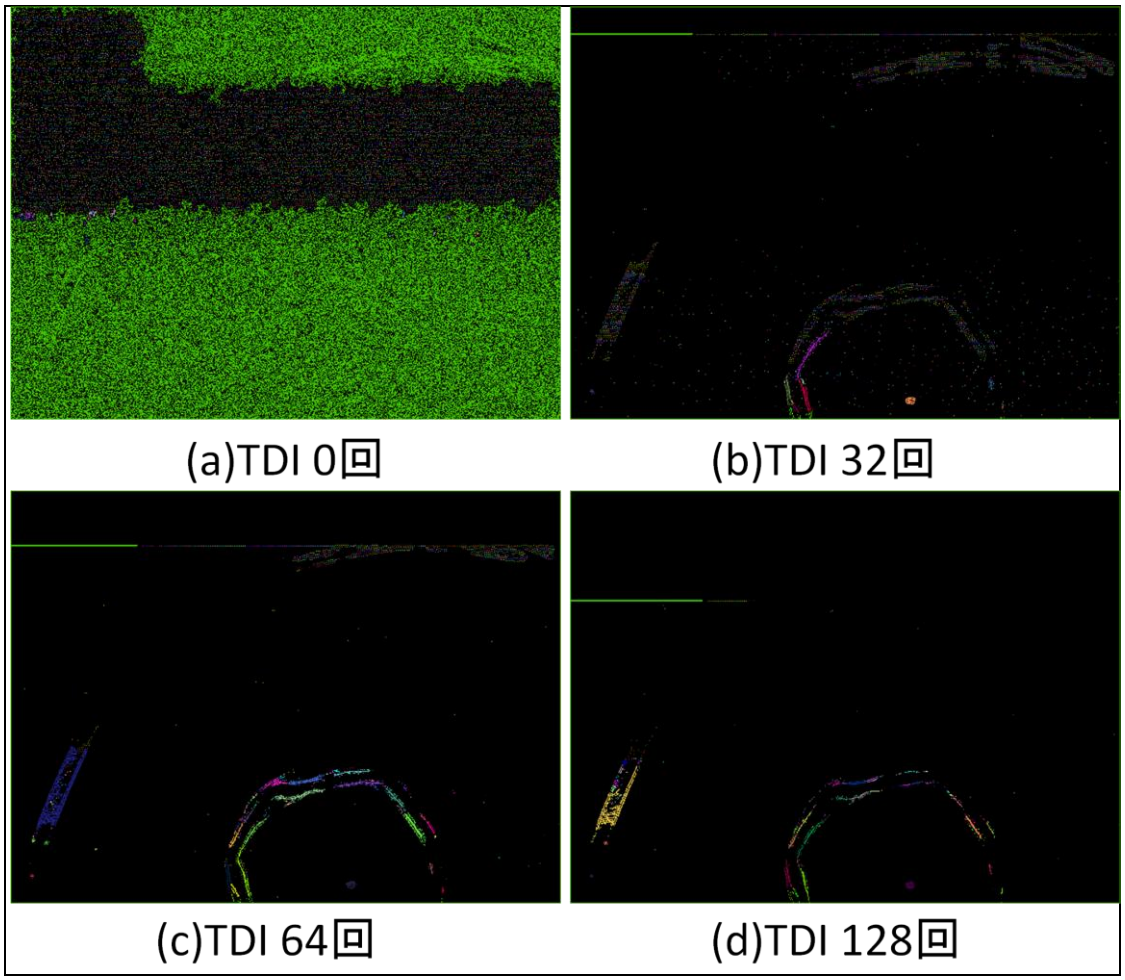

図 12：ガラス画像を対象に画像処理プログラム実行結果

図 12 をみると、TDI 処理を行うことによって欠陥を検出できたことが分かる。また、TDI の回数によってラベルの枚数と処理時間の変化を表 2 に示す。実験に用いた PC スペックは表 1 で検証したときと同じく、CPU は Core2Duo3.16GHz、RAM は 4GB である。

表 2：TDI の回数に伴うラベルの枚数と処理時間の変化(ガラス画像)

TDI(回数)	ラベル(個)	処理時間(sec)
0	12631	9.95
32	3080	2.39
64	1081	1.00
128	274	0.35

表 2 から、ガラスの画像では TDI を 128 回行ったときが、ラベルの枚数と処理時間が最も少なくなるということが分かった。

4. Verilog-HDL での画像処理モジュールの説明

4.1 画像処理全体

Verilog-HDL で記述する TDI、ラプラシアンフィルタ、ラベリングのプログラムの記述の考え方は C 言語で記述する際とさほど変わらない。TDI、ラプラシアンフィルタ、ラベリングを順に処理させていけば良い。図 13 に全体の画像処理プログラムのモジュールを示す。また、図 14 にデータの流れをブロック図として示す。

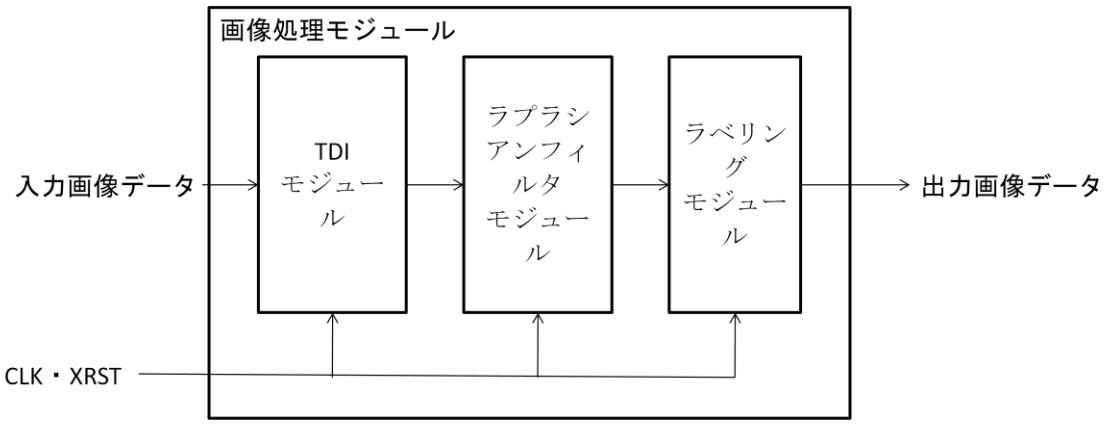

図 13：画像処理モジュールの全体図

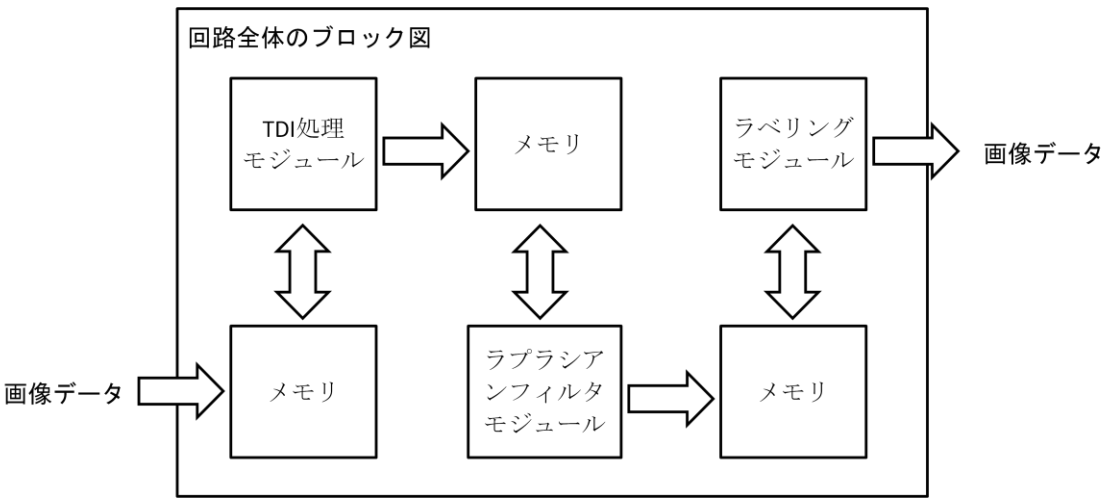

図 14：データの流れ

4.2 TDI

TDI は 2.1 で説明した通り、画像をずらして平均を出力するものである。図 15 に TDI モジュールを示す。2 枚目以降ずらしてデータを入力していくために、input モジュールを

作成する。Input モジュールから受け取ったデータを、総和モジュールで各画素ずつ合計する。そして除算モジュールで、各画素の合計値を入力枚数で割って、各画素の平均値を出力する。Verilog-HDL ではプログラミングの際、C のように除算を使うことができないので、擬似的な除算モジュールを作成する必要がある。output モジュールで処理結果画像を出力する。

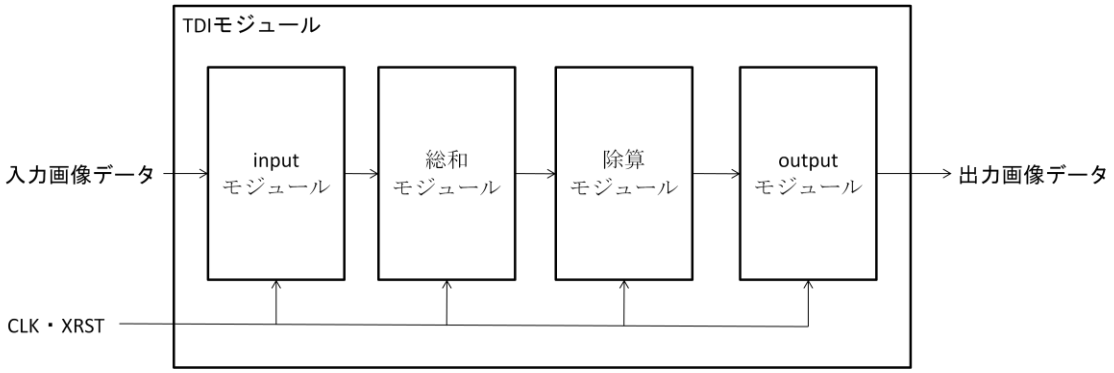

図 15：TDI モジュール

4.3 ラプラシアンフィルタ

input モジュールで、3 行 3 列の画像を読み込む。読み込み方は下の図 16 のように行う。

a	b	c
d	e	f
g	h	i

図 16：取り込み画像イメージ

取り込んだ画像の中央である e を基準に処理していく。処理内容としては、「e の周囲 8 画素の合計値が e を 8 倍した値(以降 8e と表記)以下」であれば 0 を出力する。そうでなければ、「周囲の合計値から 8e を引いた値」が 255 以上のとき 255 を出力する。そうでなければ、周囲の合計値から 8e を引いた値を出力する。図 17 にそのフローチャートを示す。

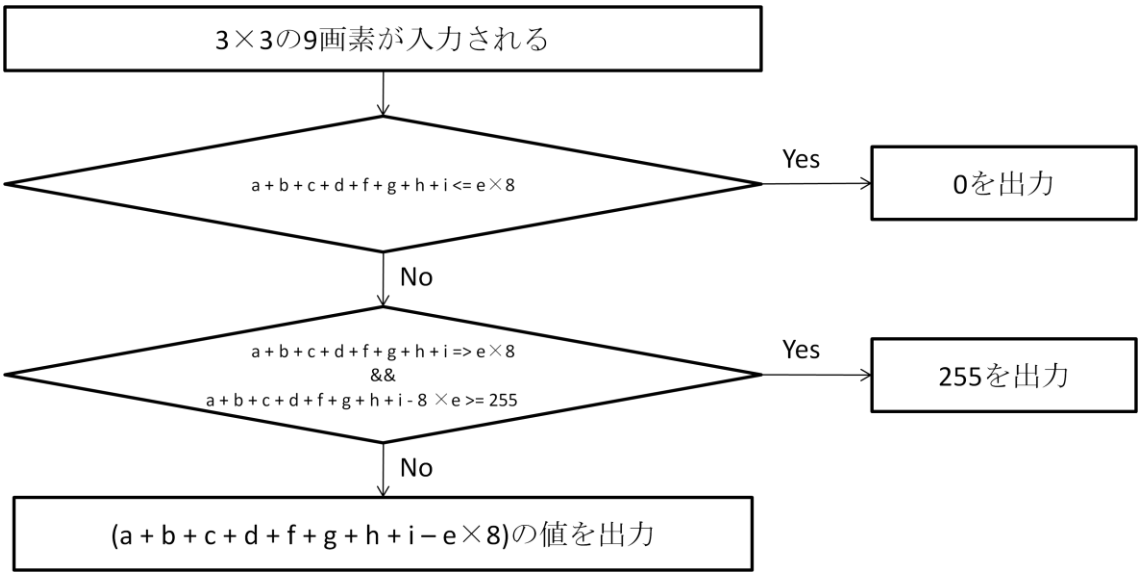

図 17：ラプラシアンフィルタのフローチャート(Verilog-HDL)

4.4 ラベリング

処理内容としては、おおまか図 4 のフローチャートと同様になる。今回 Verilog-HDL で記述するラプラシアンフィルタは、輝度値 255(白)と(a + b + c + d + f + g + h + i -8e)の値で描画される強調された輪郭と、背景が輝度値 0(黒)で出力されるため、ほぼ 2 値化になっている。ラベルは輝度値が 0(黒)以外にラベルを貼っていけば良いことになる。

5. 考察

図 5 の(a)の原画像のノイズが、TDI を行った(b), (c), (d)を肉眼で見て比較しても減っている(背景が滑らかになっている)ことが確認できる。このことによって、回数に限らず TDI を行うことによってノイズが減るということが考えられる。ラプラシアンフィルタ処理を行った画像である図 6 と図 7 を見ると、4 近傍と 8 近傍、どちらのラプラシアンフィルタが良いか肉眼では確認できないが、ラベリング処理を行った画像である図 9、図 10 を見ると、8 近傍の方が効果的だということが分かった。そして、TDI も何回行うと良いというのも分かった。

TDI を 32 回行った図 10 の(a)を見ると、背景にノイズは多いが 64 回行った図 10 の(b)と 128 回図 10 の(c)は殆どない。僅かに 128 回行った図 10 の(c)の方が、TDI を 64 回行ったものよりノイズが少ないということは肉眼で見て確認できる。しかし、表 1 を見てみると、ラベルの枚数が TDI を 128 回行った方が TDI 回数 64 回と比べ、多くなっている。これらのことから、TDI はノイズ除去に有効であり、本研究で用いた英語論文の実験用画像では処理回数は 64 回前後が一番有効だということが分かった。

しかし、理論上では TDI の回数が増えれば増えるほどノイズは減るはずである。そこで、背景ではなく、文字に注目することにする。TDI 処理を 64 回、128 回処理した画像中の文字「A Case-Based ~」の「A」を拡大したものを図 18 に示す。

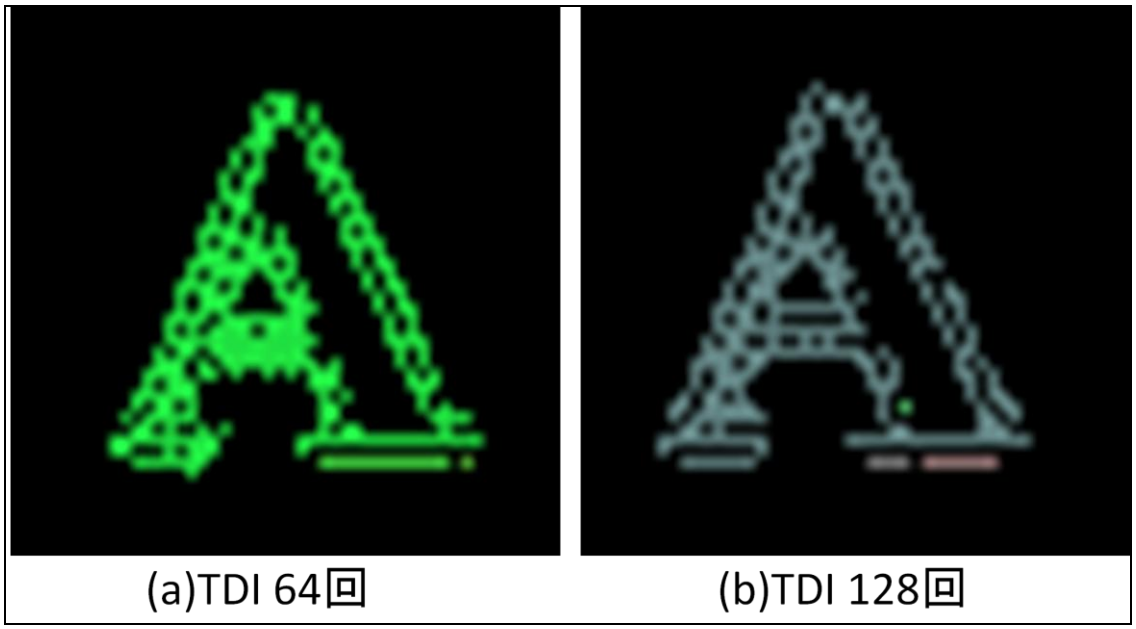

図 18 :「A」拡大図

図 17 の(a)と(b)を比較すると、(a)はほぼ同じ色であるのに対し、(b)は異なる色が混じっていることが分かる。ラベリングの前に行う画像の 2 値化で設定する背景幅は、範囲を狭めれば抽出する文字の範囲が大きくなるが、代わりに背景のノイズも拾いやすくなる。し

かし、今回は TDI 処理を行っているため、そのノイズを拾う可能性は小さくなる。このことから、2 値化を行う際の背景幅の設定を工夫すれば改善できるのではないかと考えた。図 10 では、背景幅の設定を 100~160 に設定していたが、今度は 100~160 を 102~158, 105~155 と、順に範囲を狭めて実行した。実験に用いた PC スペックは 3 章のときと同じく、CPU は Core2Duo3.16GHz、RAM は 4GB である。実行結果が表 3 になる。

表 3：背景幅の変更に伴うラベルの枚数と処理時間の変化

背景幅	TDI(回数)	ラベル(個)	処理時間(sec)
100~160	64	159	0.51
100~160	128	535	0.84
102~158	64	108	0.39
102~158	128	147	0.51
105~155	64	93	0.34
105~155	128	77	0.35
108~152	64	176	0.40
108~152	128	68	0.34
110~150	64	721	0.79
110~150	128	81	0.38
120~140	64	23724	18.07
120~140	128	13879	10.59

背景幅を 108~152 に設定して TDI を 128 回行い、ラベリング処理をさせたときが一番ラベルの枚数が少なく、処理時間も最も短い。そして、110~150, 120~140 とさらに背景幅を狭めていくとラベルの枚数及び処理時間も多くなるということが分かった。

実験をして判明した問題は、ラプラシアンフィルタを適応した画像は、各画素値の平均が 130 前後になるため、単純に平均値の±30 の背景幅で 2 値化、ラベリングをすれば最適な結果が得られるというわけではなく、手動で最適な背景幅を調べなければならないという点である。また、この実験では TDI を 128 回行ったものが最適な結果が得られることとなったが、検証する画像次第によっては、TDI の回数が 128 回よりも少なければならないときも考えられる。そのときそのときの最適な TDI の処理回数も手動で調べなければならない。これらのことから、この一連の画像処理プログラムを全自動化しても、出力される結果が最適な結果ではない可能性があるとうことである。

次に、Verilog-HDL で記述した加増処理プログラムと、C で記述した画像処理プログラムとを比較した場合の検討を行う。C と比べて一番速くなる画像処理プログラムは、ラベリング処理だと考えられる。その理由は図 4 のフローチャートが示すとおり、ラベリングが本研究で作成した画像処理プログラムの中で一番処理回数が多いためである。また、C と比

較し、最も差が出ない、もしくは遅くなると考えられるのは TDI である。TDI では画像の平均を計算しないといけないため、除算が必要となるが、Verilog-HDL では除算を行うことが出来ず、除算回路を作成し処理させなければならない。そのことから、図 23 のように 2 つのモジュールを使って処理させなければならなくなる。このことから遅延が少なからず発生すると考えられる。

6. おわりに

本研究では、TDI の有効性を実験で検証することを目的とし、C プログラムで TDI、ラプラシアンフィルタ、2 値化、ラベリングを行った。TDI を行わなかった場合と、TDI を 32 回、64 回、128 回行った場合の 4 種類の画像を用意し、それらに 4 近傍と 8 近傍のラプラシアンフィルタと 2 値化、ラベリングを行った。TDI を行わなかった場合は、ラプラシアンフィルタでかなりの量のノイズが強調されてしまい、ラベリングは有効に働かなかったが、TDI を行った画像では、TDI の処理回数を増やすたびにラプラシアンフィルタで強調されるノイズの量が減り、ラベリングも有効的に行えることが確認できた。これらの結果から、TDI の有効性が確認できた。

今後の課題として、Verilog-HDL で画像処理プログラムを記述し、シミュレーションを行い、FPGA ボードに実装し C プログラムとの速度比較を行うことが挙げられる。

謝辞

本研究の機会を与えてくださり、貴重な助言、ご指導をいただきました山崎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いただき、貴重な助言を頂いた John 氏、安部氏、Thuan 氏また共同研究者の前田氏、松山氏及び様々な面で励ましを下さった高性能研究室の皆様に心より深く感謝いたします。

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

[2] 小林優 : 改訂 入門 Verilog HDL 記述, CQ 出版, 2009

[3] 大山佳宜 : 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電子情報デザイン学科卒業論文, 2010

[4] 乾圭佑 : 画像処理アルゴリズムのハード/ソフトの最適分割の検討, 立命館大学理工学部電子情報デザイン学科卒業論文, 2009

[5] 安部厚志 : 画像処理プログラムの HDL 記述とハードウェア動作合成システムの検証(I), 立命館大学理工学部電子情報デザイン学科卒業論文, 2008

[6] 大岩広司 : 画像処理プログラムの HDL 記述とハードウェア動作合成システムの検証(II), 立命館大学理工学部電子情報デザイン学科卒業論文, 2008

[7] 千村亮介 : ラベリングとマハラノビス距離による画像判別の実現, 立命館大学理工学部情報学科卒業論文, 2006

[8] 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情報学科卒業論文, 2006

[9] 松崎裕樹 : マハラノビス距離を用いたガラス検査用画像判別の実現, 情報処理学会関西支部支部大会 VLSI システム研究会, 2006