A-03

FPGA ボードを用いた液晶用ガラスの欠損検出画像処理の高速化

Fast Flaw Detection of Liquid Crystal Glass with a FPGA Board

松山 圭輔†　　孟 林†　　山崎 勝弘†

Keisuke Matsuyama　Lin Meng　Katsuhiro Yamazaki

1.　はじめに

近年、液晶用ガラスはテレビやパソコンディスプレイ、スマートフォンなど、様々な分野で使用されている。液晶用ガラスの欠損を検出するためには、大量の画像データの処理が必要であり、その高速化が極めて重要である。これに対応するために、本研究では FPGA ボードを用いて高速化を図る。本研究では、まず画像のノイズ除去を行い、次にラプラシアンフィルタをかけてエッジ抽出を行い、それを 2 値化して最後にラベリング処理を行って液晶用ガラスの欠損を検出していく。本稿では、FPGA ボード上でのラプラシアンフィルタの実装と評価について述べる。

2.　画像処理アルゴリズム

2. 1　画像処理の手順

本研究で行った画像処理の手順を図 1 に示す。対象画像に対して、TDI (Time Delay Integration) と呼ばれるノイズを除去するアルゴリズムを用いて処理を行い、次にラプラシアンフィルタを用いて画像を処理して輪郭を検出して 2 値化を行う。最後にラベリングという画像内のオブジェクトを識別するための処理を行う。

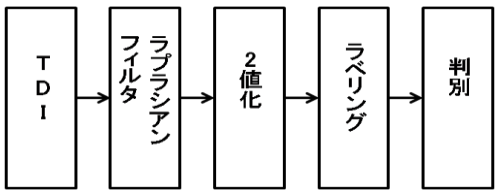
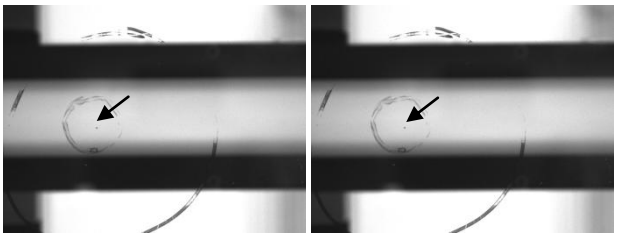


図 1.　画像処理の手順

本研究で使用したサンプル画像を図 2 に示す。図 2 に示す画像の画面中央の少し左にある点が傷である。図 2 (b) の画像は、図 2 (a) の画像を 1 画素ずらした物である。このように 1 画素ずつずらして撮影した画像をノイズ除去のために最大で 128 枚使用した。



(a) 1 枚目　　(b) 2 枚目

図 2.　サンプル画像

2. 2　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。画像の画素を数値化し、TDI 処理する過程を図 3 に示す。図 3 (a) が 1 枚目の画像、図 3 (b) が 2 枚目の画像である。このように、1 画素ずらして撮影したものを使用する。

0	0	0	0
1	0	1	2
1	2	2	2
3	2	3	2

2	1	1	1
2	2	3	2
3	3	3	4
3	4	4	4

(a) 1 枚目　　(b) 2 枚目

図 3.　画素の平均化

図 3 の 2 枚の画像を見ると、間の 3 行が共通部分であり、この数値のそれぞれの平均値を出す。図 4 (a) が理想の画

† 立命館大学大学院理工学研究科、Graduate School of Science and Engineering, Ritsumeikan University

像を数値化したもので、共通部分を平均化したものが図 4（b）である。比べると初期より理想の状態に近づいたと言える。画像の枚数が多ければ多いほど共通部分が少なくなり、画像が小さくなってしまうが、より滑らかな画像が手に入る。

1	1	1	1
2	2	2	2
3	3	3	3

1.5	0.5	1	1.5
1.5	2	2.5	2
3	2.5	3	3

（a）理想の画像　　（b）処理後

図 4．理想の画像と処理した画像

2．3　ラプラシアンフィルタ

ラプラシアンフィルタとは、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。画像に対して二次微分をするフィルタで、ニ変数関数 f（x、y）のラプラシアンは偏微分を使うと式（1）のように表せる。

$$L(x, y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y) \quad \cdots (1)$$

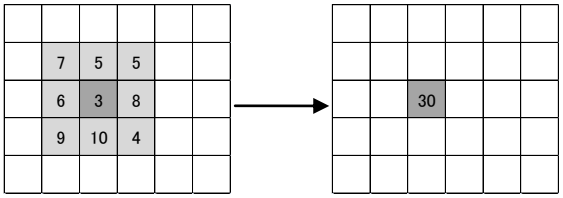
これを簡単な式で表すと、式（2）のようになる。

$$L(x, y) = 4f(x, y) - \{f(x, y\text{-}1) + f(x, y+1) + f(x\text{-}1, y) + f(x+1, y)\} \quad \cdots (2)$$

これを係数で表すと、$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ となり、

45° 方向も含めると、$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ という係数になる。

本研究では、後者の係数をマスクパターンとして使用する。例として、ラプラシアンフィルタ処理前の画像と処理後の画像を図 5 に示す。

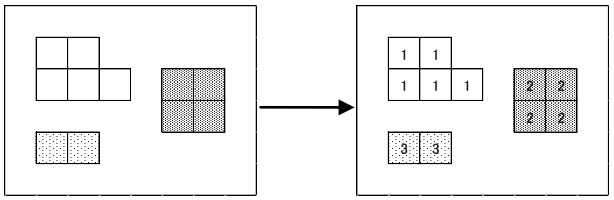


（a）処理前　　（b）処理後

図 5．ラプラシアンフィルタ処理

2．4　ラベリング

ラベリングとは、画像の繋がっている画素（連結成分）に同じ番号（ラベル）をつけ、異なった連結部分には異なった番号をつける処理である。ラプラシアンフィルタ処理と 2 値化処理を行った画像に対してラベリングを行うことにより、最後まで残ったノイズ（傷）を検出することができる。例として、ラベリング処理前の画像と処理後の画像を図 6 に示す。



（a）処理前　　（b）処理後

図 6．ラベリング処理

左上の画素から右に向かって走査を始め、右下の画素までラベル付けを行う。ラベルを付ける際に、周りの画素 4 つ（上、右上、左上、左）を調べる 4 近傍で行っている。

3．　FPGA ボード上でのラプラシアンフィルタの実装

ラプラシアンフィルタのブロック図を図 7 に示す。まず、入力された画像データを lap で処理する。メモリに、カウンタ兼アドレスを用いて格納していく。

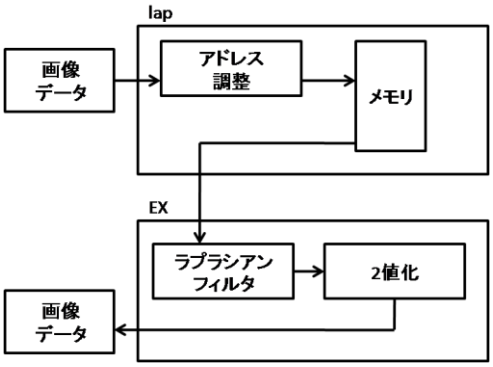


図 7．ラプラシアンフィルタのブロック図

メモリに格納する時に使用したカウンタ兼アドレスを用いて EX に演算に必要な 3×3 の画素データを渡し、ラプラシアンフィルタ処理を行う。ラベリング処理を行いやすいようにするために 2 値化判別もここで行う。ラプラシアンフ

ィルタ処理を行った画像データに対してすぐ閾値判別を行い、画像データが閾値より上なら画像データ 255（白）に、閾値より下なら 0（黒）を入れて出力する。今回閾値は 122 とした。2 値化を行う事で、画像が白と黒の 2 色となり、ラベリングを行う際の処理をスムーズに行えるようにしている。

画像データをメモリに格納し、演算処理部の EX に画素データを渡す時の処理を図 8 に示す。図 8 のようにメモリを作成し、画像データの左上から右に向かってメモリの上から下に向かって順次格納する。その上で、9 画素分の画素データを順次 EX に渡す。

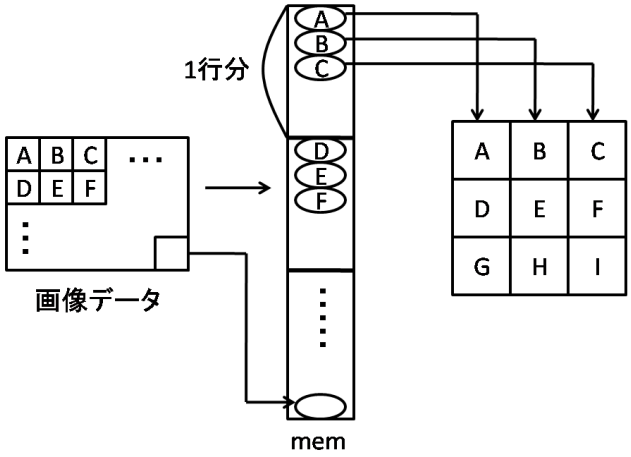


図 8. lap から EX へ渡すデータ

ラプラシアンフィルタのフローチャートを図 9 に示す。画像データを入力し、マスクパターンとかけ合わせたデータを D_temp に入れる。ラプラシアンフィルタでは出力が 0 を中心に正負の値を取るので、D_temp にオフセット(128)を足したものを D_temp1 に入れる。次に、閾値を用いて画像を 2 値化する。まず、先ほど計算した D_temp1 が 0 よりも小さい場合は 0（黒）を D_Out に出力し、255 よりも D_temp1 が大きい場合は 255 を D_Out に出力する。D_temp1 の値が 0～255 の間かつ 122 よりも上なら 255 を、122 よりも下なら 0 を D_Out に出力するようになっている。

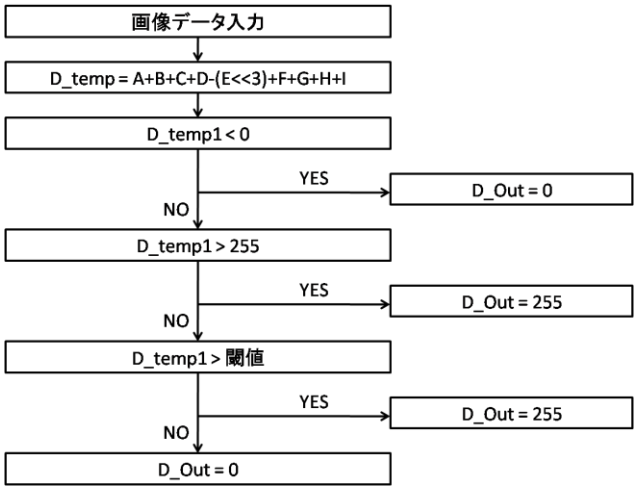


図 9. ラプラシアンフィルタのフローチャート

4. FPGA ボード上での実験

4. 1 実験条件

使用した画像は図 2 で示した画像を用い、サイズは 640×480 画素、入力データを 8bit、出力データを 8bit とし、TDI を 128 回行った画像を使用した。また、記述した回路との処理時間の比較をするために CPU（パソコン）でも処理を行い、その時間を計測した。計測に用いたパソコンのスペックは、プロセッサが Intel® Core™2 Quad CPU Q9400 2.67GHz、実装メモリが 4.00GB、OS は Windows 7 Ultimate、32bit オペレーティングシステムである。

本研究で使用した FPGA ボードは、Xilinx 社の FPGA ボード Spartan3A Starter Kit で、同社の設計ツールを用いて設計を行った。

4. 2 実験結果

記述したラプラシアンフィルタ処理モジュールの論理合成を行い、回路規模と遅延時間、最大動作周波数などを計測した。その結果を表 1 に示す。

表 1. ラプラシアンフィルタ処理部

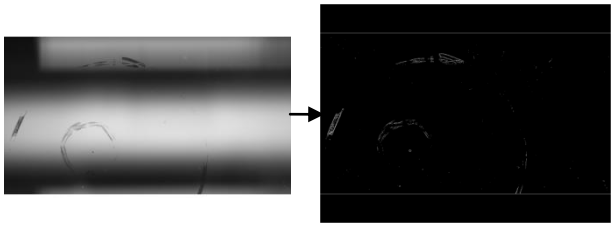
Number of Slices	314
Number of Slice Flip Flops	23
Number of 4 input LUTs	591
Delay	24.849(nsec)
Maximum Frequency	40.243(MHz)

次に、作成した回路での処理時間と、CPU での処理時間を表 2 に示す。今回使用した画像は、TDI を 128 回行ったものだが、TDI の回数を 0 回、32 回、64 回でも作成した回

路での処理時間はほぼ変わらなかった。しかし、CPU では TDI の回数が少なければ少ないほど処理に時間がかかる結果となった。

表 2. 処理時間

作成した回路	7.6（msec）
CPU（パソコン）	108（msec）



（a）処理前　　（b）処理後

図 10. ラプラシアンフィルタをかけた画像

また、ラプラシアンフィルタ処理を行った画像を図 10 に示す。実験には画像を 128 枚重ね合わせてノイズの除去したものを用いたので、128 枚の共通部分のみが処理されるため、図 10 (a) のように画像が小さくなっている。図 10 (b) の中央左下にある点がノイズ（傷）であり、正しく検出できているのが分かる。

4. 3　考察

表 2 に示すように、作成した回路での処理時間は、CPU（パソコン）での処理時間よりも約 14 倍速い、7.6msec という結果が得られた。ラプラシアンフィルタのハードウェア化により、ソフトウェアでの逐次処理よりも高速な処理が実現できた。この結果より、TDI、ラベリングについても同様に高速化できると考えられる。特に、ラベリングについては、CPU で処理させた時に一番時間のかかる処理部分なので、ハードウェア化によって一番速度が向上する部分だと予測される。

5.　おわりに

本論文では、液晶用ガラスの欠損検出のために、TDI、ラプラシアンフィルタ、ラベリングの 3 つのアルゴリズムを用いて画像処理を行い、そのうちラプラシアンフィルタのハードウェア化、及びその実験結果について述べた。CPU の処理と比べて、約 14 倍の速度向上が得られた。今後、他のアルゴリズム部分のハードウェア化を行い、それらも FPGA ボードに実装していきたい。さらには、これらの処理部分を繋げ、全体の欠損検出システムを FPGA ボード上で動作させ、システム全体の高速化を評価する予定である。

謝辞

本研究は、株式会社ケー・デー・イーとの共同研究であり、画像データを提供して頂いた。有益なコメントを頂いた三宅淳司氏と西田洋隆氏に感謝します。

参考文献

[1]　松崎、山崎、平岡、西田、三宅：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、FIT2006、I-031、2006.

[2]　松崎、山崎、小柳、平岡、西田、三宅：マハラノビス距離を用いたガラス検査用画像判別とラベリングのハードウェア化、情報処理学会関西支部　支部大会　VLSI システム研究会、C-09、2006.