卒業論文

ＧＰＵ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：岡崎大輔
学籍番号：2260070020-4
指導教員：山崎勝弘教授
提出日：平成 23 年 2 月 18 日

立命館大学　理工学部　電子情報デザイン学科

内容梗概

近年、液晶用ガラスの需要が高まっているが、液晶用ガラスには製造時に稀に欠損箇所が存在する。画像処理プログラムを用いて欠損検出を行う場合、肉眼で確認できない欠損箇所も検出することが可能である。本研究では、液晶用ガラスの欠損検出を行う画像処理プログラムに対し、GPU を用いて高速化を図る。高速化の対象となる画像処理アルゴリズムはラプラシアンフィルタ、ラベリングである。ラプラシアンフィルタを GPU 上で実装し出力したが、CPU の場合と出力結果が少し異なった。

目次

図目次

表目次

１．はじめに

近年、液晶用ガラスの需要が増え、スマートフォンの液晶など、多くの分野で使用されているが、液晶用ガラスが製造された段階では稀に欠損箇所がある。画像処理プログラムを用いて欠損検出を行えば、肉眼では確認できない小さな傷を確認することが可能である。画像処理プログラムを用いて欠損箇所の検出を行う際、高い処理精度、GPU を用いた高速な処理が必要とされる。

本研究では液晶用ガラスの欠損検出プログラムの GPU による高速化の有効性を実験で検証することを目的する。

並列計算とは大量のデータを高速に処理するための手法の１つである。スレッドやマルチプログラミングを除いて、現在のプログラミングの多くは、逐次処理されるプログラムである。並列計算では、複数の計算スレッドを同時に利用し、データの処理の高速化を行う。画像処理、映像処理のような大量のデータを扱う問題に対して非常に有効な方法である。

並列手法として、GPU を用いて画像処理プログラムの並列化を行う。GPU とは、Graphics ProcessingUnit であり、ストリーミングマルチプロセッサの中にスカラプロセッサが入っているプロセッサである。スカラプロセッサは算術演算など非常に限られた性能しかもたないが、ストリーミングマルチプロセッサには命令デコード機能などが載っている。これらのプロセッサをスレッドに分割し、並列に動作させる事により、プログラムの高速化を図る。

本研究で用いる画像処理手法は TDI、ラプラシアンフィルタ、２値化、ラベリングであり、ラプラシアンフィルタ、ラベリングの並列化を行う。これらの画像処理プログラムを並列化し、GPU 上で実行するため、CUDA プログラミングを行う。傷検出プログラムのアルゴリズムとして、ラプラシアンフィルタ処理を行う事により、液晶用ガラスの欠損箇所のエッジ抽出を行う。次に、２値化処理によりエッジ抽出部分、非抽出部分の画素値を 0、255 のそれぞれに２値化する。次に、ラベリングする事により、２値化された画像のラベル付けを行う。ラベル付けされる欠損箇所が隣接する画素である場合、同じラベル番号を付ける。

２．画像処理アルゴリズム

２．１　Time Delay Integration

Time Delay Integration とは、同じ画像の画素をずらし複数回撮影し、各画素を積算して平均化を行う。(a)~(d)の各画像の積算処理により S/N 比を高めることが可能である。赤い数字がノイズの含まれる画素である。各画素をずらし撮影する画像を図１に示し、積算処理によって得られる画像を図２に示し、ノイズのない理想データを図３に示す。図２、図３より、理想のデータに対し平均化を行う事により、よりノイズの少ないデータを得る事ができる。

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

(a)被積算画像1

2	2	2	2
3	3	3	3
4	3	4	4
5	4	5	5
6	6	6	6
7	7	7	7
8	8	8	8

(b)被積算画像2

1	1	1	1
2	2	2	2
3	3	3	3
4	4	5	4
5	5	5	5
6	6	6	5
7	7	7	7

(c)被積算画像3

0	0	0	0
1	1	1	1
2	2	2	2
3	3	4	3
4	4	4	5
6	5	5	5
6	6	6	6

(d)被積算画像4

図 1. TDI 処理前

3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6

図 2. 理想画像

2.8	3	3.3	3
4	3.8	4.3	4.3
5.3	4.8	5	5
6	6	6	5.8

図 3. TDI 処理後

２．２　ラプラシアンフィルタ

ラプラシアンフィルタとは、２次微分フィルタリングを行い、画像中のエッジ部分を抽出するフィルタである。２次微分ではエッジの中心、輪郭線に相当する部分に正負の山ができる。この正から負へ変化する部分を求める事により、直線輪郭線を求める。

ラプラシアンは偏微分を用いて次式のように表す。

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

またグラディアントを再度微分する事で次式のように表す。

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

またラプラシアンの計算に用いる微分オペレータとして、４近傍、８近傍など数種類存在する。

４近傍の微分オペレータは $X = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

８近傍の微分オペレータは $X = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$,である。

４近傍、８近傍の特徴として、４近傍では上下左右４方向の差分を計算する事に対し、８近傍では上下左右の差分に加え斜め方向を含む８方向の差分を求める。これにより、双方のラプラシアンフィルタを比較すれば、エッジ抽出の精度は後者の方が上高い。
本研究では、検証した結果、８近傍でフィルタリングを行う。

２．３　ラベリング

ラベリングとは、画素の連結しているすべて画素（連結成分）に同じラベル（番号）を付け、異なった連結成分には異なった番号を付ける処理である。この処理により個々の連結成分に分離することができ、各連結成分の特徴を調べることが可能である。２値化した画像をラベリングする手法として、ルックアップテーブル法を用いる。手順として、

① 注目画素に対し(y-1,x),(y,x-1)(y-1,x-1),(y-1,x+1)の画素値を調べ、注目画素と対象の画素値が同じなら同じラベルをつけ配列を image1 に格納する。

(y−1,x−1)	(y−1,x)	(y−1,x+1)
(y,x−1)	注目画素	

② ラベルづけを行った image1 の配列を読み込み、(y,x+1),(y+1,x),(y+1,x-1),(y+1,x+1)と注目画素の画素値を調べる。注目画素と(y,x+1),(y+1,x),(y+1,x-1),(y+1,x+1)の成分を比較し、①でラベルづけされていて、且つ注目画素と成分のラベルが違う場合、注目画素と同じラベルにし、image2 に格納する。

	注目画素	(y,x+1)
(y+1,x−1)	(y+1,x)	(y+1,x+1)

③ この処理を画像サイズである 1280x960 全て行い、以降、②の処理を繰り返す。image2=②の処理の結果となった場合、条件文を break し処理を終了する。

３．Ｃ言語での画像処理実験結果

３．１　実験環境

C 言語での実験は OPTIPLEX755 を用いる。

CPU(Hz)：Core2Duo(3.16G)

メモリ：4.0GB

HDD：250GB

入力画像は 8bit のビットマップファイルであり、液晶用ガラスの欠損画像である。画像の大きさは 1280x960 である。TDI の回数は 0 回、32 回、64 回、128 回である。矢印で示す点が欠損箇所である。

３．２　TDI

TDI を 32 回、64 回、128 回行い、出力画像した画像を図 4 に示し、比較する。原画像と 32 回、64 回、128 回、TDI を行った画像を比較すると、32 回、64 回、128 回と徐々にノイズが軽減されている。

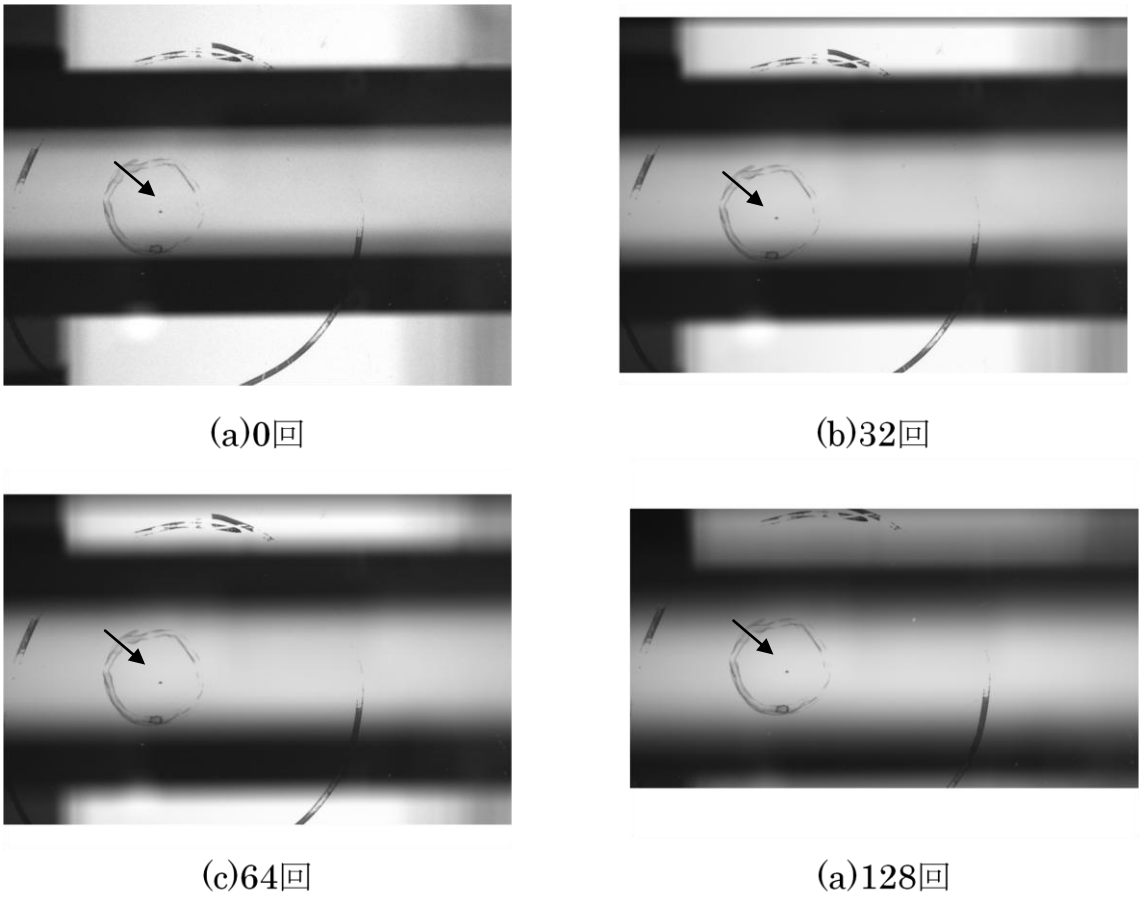
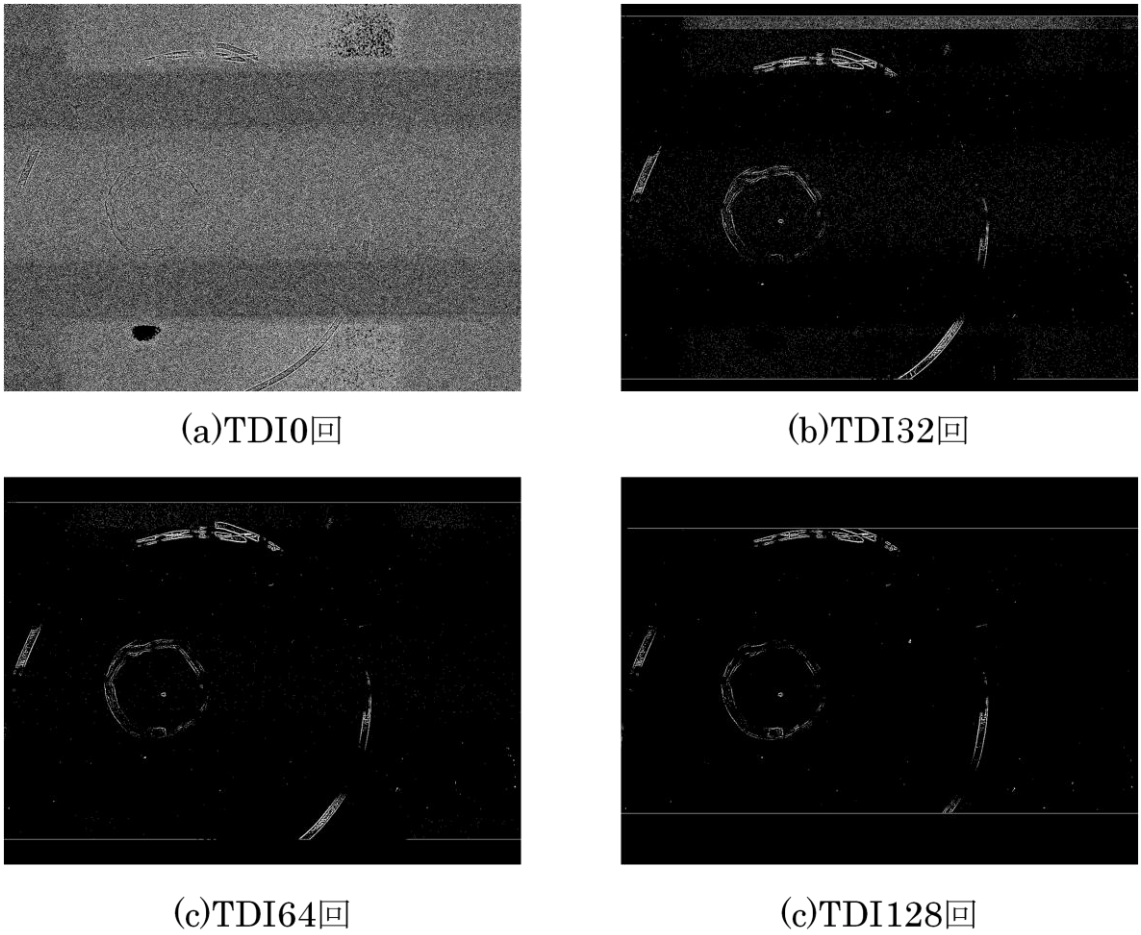


図 4. TDI 画像

３．２　ラプラシアンフィルタ

TDI を 32 回、64 回、128 回行い、ラプラシアンフィルタの処理後、２値化を行った画像を図５に示す。その処理時間、出力画像の比較を行う。処理速度の変化は(a)、(b)、(c)、(d)全て 0.17 秒という結果が得られた。処理精度に関して、(a)と(b)の画像を比較すると、大きくノイズを軽減する事ができた。また(b)の画像から、(c)、(d)と比較すると、大きくはないがノイズが軽減されている。



(a)TDI0回　　(b)TDI32回

(c)TDI64回　　(c)TDI128回

図 5. ラプラシアンフィルタ

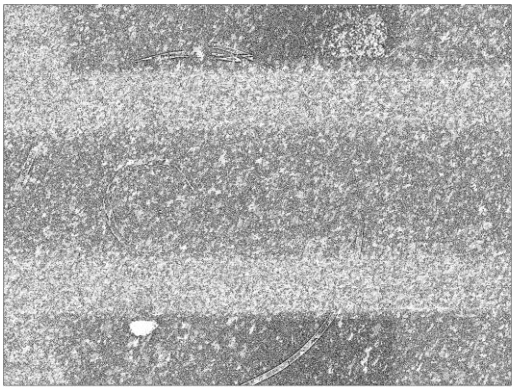
表 1. ラプラシアンフィルタ処理の実行時間

単位：秒

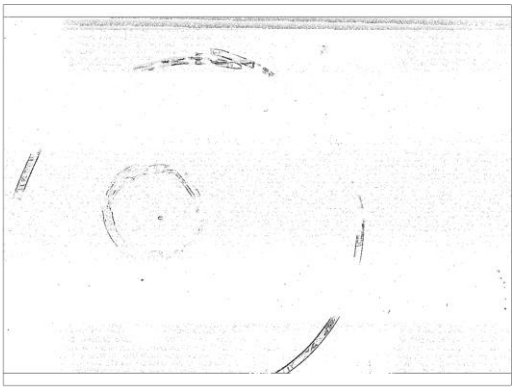
	TDI0回	TDI32回	TDI64回	TDI128回
実行時間	0.155	0.155	0.156	0.155

３．４　ラベリング

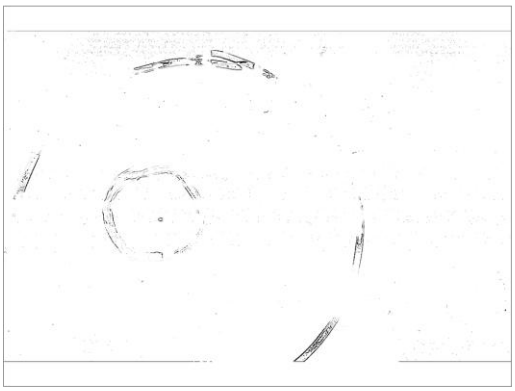
TDI、ラプラシアンフィルタ、２値化、ラベリング処理を行い、出力した画像を図６に示す。(a)の画像ではノイズの影響でフィルタリング、２値化が行えていないため、ラベリングが行えていない。(b)、(c)、(d)ではラベリングが行えている。(b)、(c)、(d)の画像を比較すると TDI の回数を多くするほどノイズの軽減が得られている。また実行時間を表２に示し、速度向上比を図３に示す。



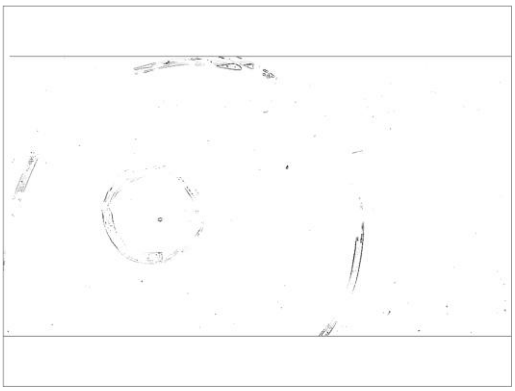
(a)TDI0回



(a)TDI32回



(a)TDI64回



(a)TDI128回

図 6. ラベリング

表 2. ラベリング処理の実行時間

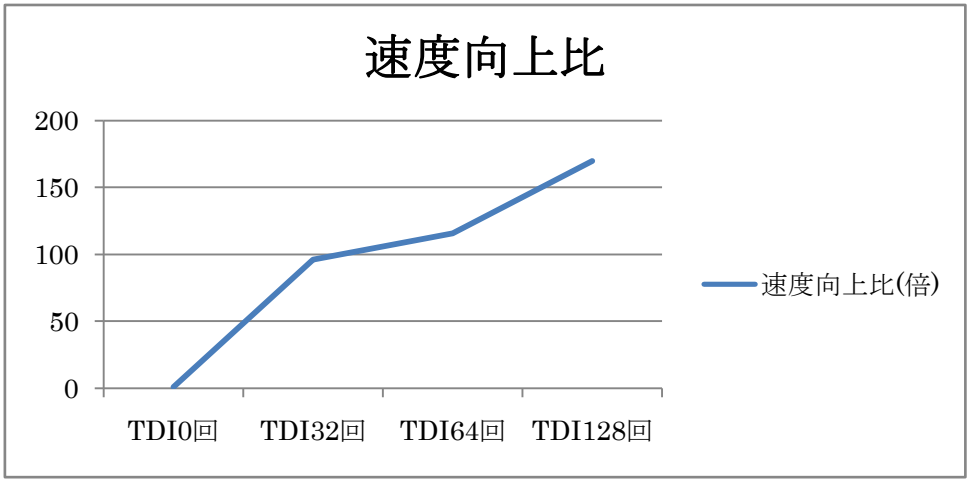
単位 : 秒

	TDI0回	TDI32回	TDI64回	TDI128回
実行時間	307.766	3.202	2.656	1.844

表 3. 速度向上比

	TDI0回	TDI32回	TDI64回	TDI128回
速度向上比	1	96.1	115.8	170.1

図 7. 速度向上比



４．GPU による画像処理高速化の実装と検討

４．１　実験環境

CUDA での実験は UNI-D2G+T を用いる。

CPU(Hz)：Core2Quad(2.66G)

メモリ：4.0GB

HDD：1000GB

GPU：TeslaC1060

CUDA プロセッサコア：240

４．２　動作概要

CUDA はコンピュータのマザーボード上にあるビデオカード内で動作を行う。このとき、マザーボードには CPU があり、メモリも装着されている。また、ビデオカードには GPU が搭載されている。マザーボードおよび CPU 側がホストであり、GPU 側がデバイスである。デバイス上でプログラムを実行する際のホスト、デバイスの動きを図 7 に示す。

GPU 上のプロセッサの構成として、CUDA のハードウェアの中で一番小さい単位がスカラプロセッサ(SP)である。スカラプロセッサの上位にストリーミングマルチプロセッサがあり、スカラプロセッサを複数個持つ。

並列処理の実行単位としてブロック、グリッドがある。１ブロック内の最大スレッド数は 512 であり、１グリッド内の最大ブロック数は 65535 である。これらを同時に動かすことにより並列効果を得ることが可能である。

本研究で用いているラプラシアンフィルタの動作概要を図８に示す。逐次処理では一次元配列である画素値を順番に積算するのに対し、GPU を用いた並列処理では画素をブロック数で分割し、そのブロック数をさらにスレッド数で分割し、積算する。

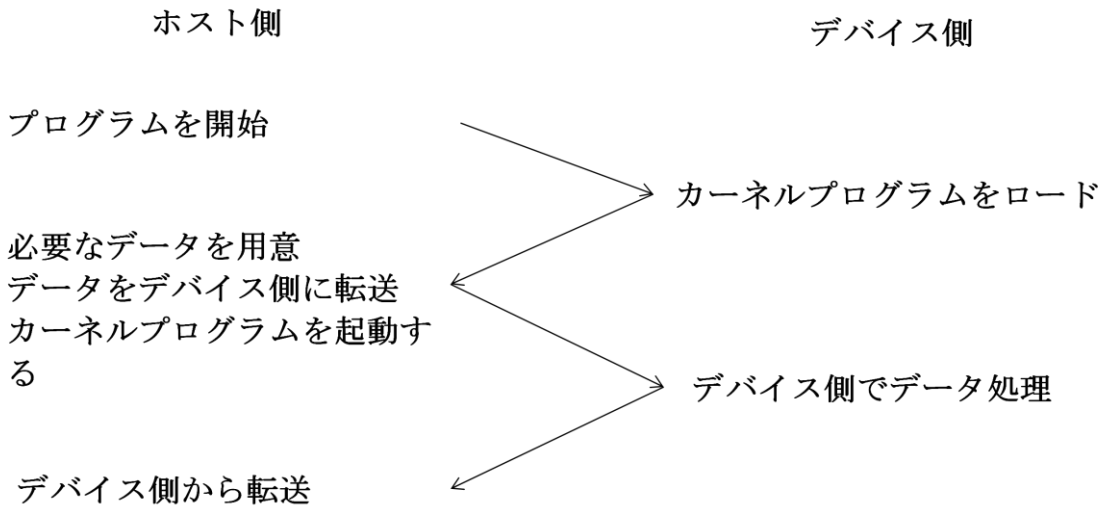


図 8. ホスト側、デバイス側のプログラムの流れ

図 9. GPU 上でのラプラシアンフィルタ動作概要

４．３　ラプラシアンフィルタ

1280x960 の画像に対し、ブロック数、スレッド数を変化させ並列処理を行い、速度比較を行う。TDI の回数を 0 回、32 回、64 回、128 回行い、CUDA 上でラプラシアンフィルタの処理後、２値化を行った画像を図９に示す。CPU と GPU での処理速度の比較を表４に示し、ブロック数、スレッド数を変化させ実行したものを表５に示す。CPU と GPU で出力画像が同じになるはずだが、GPU での処理では少しノイズの影響を受けているので正確な速度比較はできない。CPU の実行時間 0.155 秒に対し GPU での実行時間は 0.008 秒であり、約 20 倍の速度向上が得られた。ブロック数、スレッド数を変化させた速度変化では、32 スレッドを 1 ワープとし、実行しているものは、最も高速であったが、ブロック数が多い、スレッド数が 16 の倍数でないものは GPU 上の実験では高速でなかった。引き続き、出力結果が CPU、GPU で同じものが得られるまで検討していく必要がある。

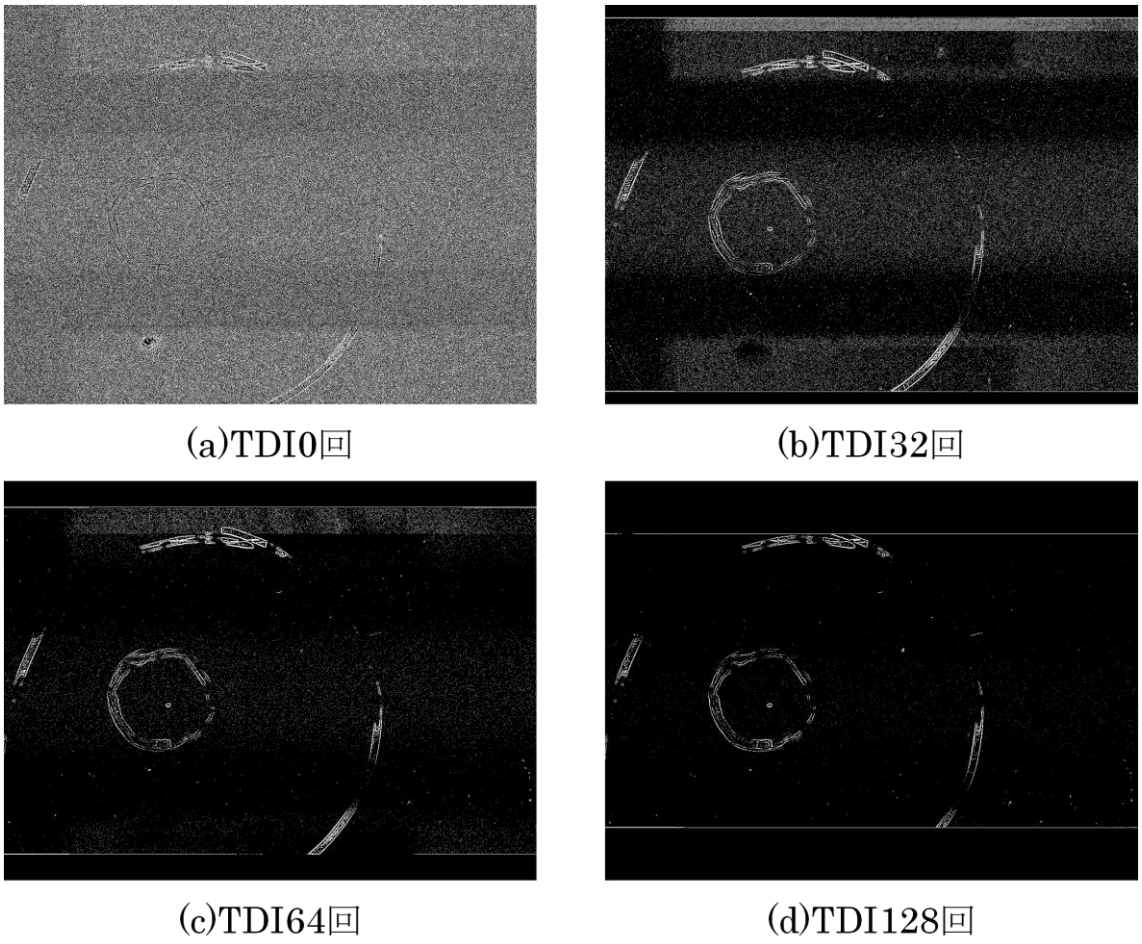


図 10. CUDA 上でのラプラシアンフィルタ

表 4. CPU、GPU での速度比較

単位：秒

	TDI0回	TDI32回	TDI64回	TDI128回
CPU	0.155	0.155	0.156	0.155
GPU	0.008	0.008	0.009	0.009

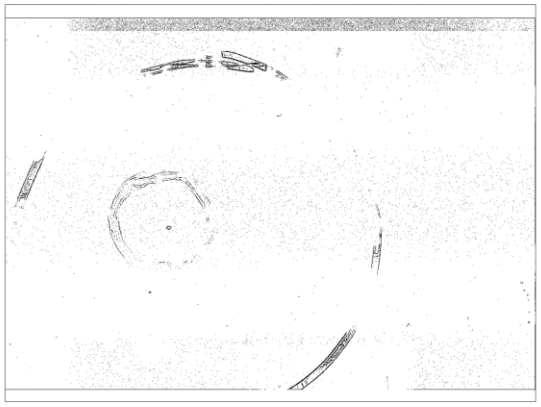
表 5. 各ブロック数、スレッド数での速度比較

単位：m秒

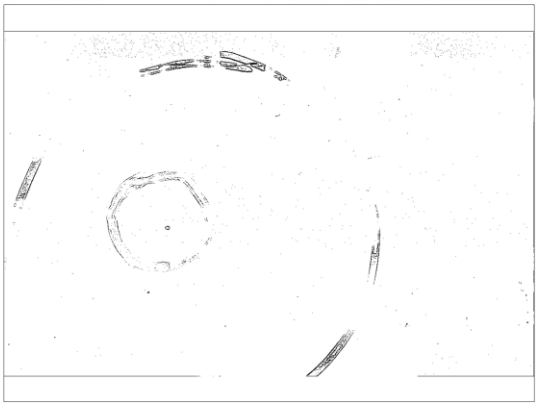
<gd,bd>	<80,16>	<40,32>	<64,20>	<3,512>
実行時間	8.721	8.446	8.798	10.11

４．４　ラベリング

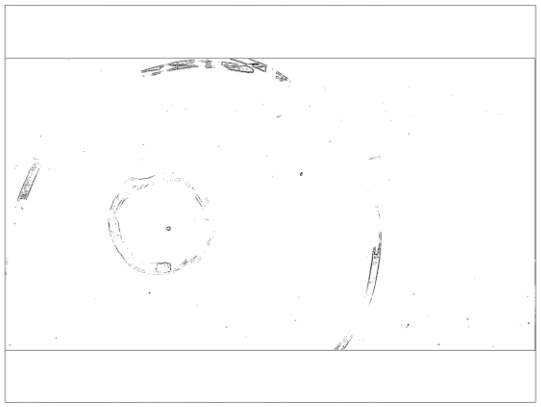
1280x960 の画像に対し、ブロック数、スレッド数を変化させ並列処理を行い、速度比較を行う。TDI の回数を 0 回、32 回、64 回、128 回行い、CUDA 上にラベリングを行った画像を図１１１に示す。CPU、GPU で速度比較しているのだが、CPU の時に比べ正しくラベル付けができていない箇所がある。TDI が回のものに対してはエラーがかえってきた。また、速度比較を表６に示す。



(a)TDI32回



(b)TDI64回



(c)TDI128回

図 11. CUDA 上でのラベリング

表 6. CPU、GPU の速度比較

単位：m秒

TDI回数	なし	32	64	128
CPU	307.766	3.202	2.656	1.844
GPU	6135	105.6	58.91	41.79
CPU/GPU	51	30	45	44

表 7. ブロック数、スレッド数を変化させた速度比較

単位：m秒

〈ブロック数、スレッド数〉	〈10,128〉	〈20,64〉	〈40,32〉	〈80,16〉	〈160,8〉
処理時間(ms)	34.46	44.27	40.69	41.15	41.78

５．　考察

ラプラシアンフィルタでは TDI の回数を増やすことにより 0 回のときはエッジを抽出することができなかったが、TDI の回数を増やすことによりエッジを抽出することができた。それぞれ TDI の回数を変化させラプラシアンフィルタ、２値化を行った出力画像にラベリングを行うと、エッジの抽出できている画像に対しては、アルゴリズムと同様のラベリングを行うことができた。

GPU での画像処理の高速化について、ラプラシアンフィルタを CPU、GPU 上で実行し高速化を図った。CPU 上での実行では 0.155 秒だったのに対し、GPU 上での実行では 0.008 秒であった。これは約 20 倍の速度向上が得られたが出力画像が少し違うので正確に速度比較することができない。ブロック数、スレッド数を変化させ実行した場合、1 ワープ 32 スレッドで実行した場合が最適であった。またラベリングも同様、約３０～５０倍程度速度向上を得る事ができたが、少しアルゴリズムと違う処理をしている。

６．　おわりに

本研究では、画像処理プログラムの GPU を用いた高速化の有用性を検証することを目的とし、ラプラシアンフィルタの高速化を図った。ラプラシアンフィルタでは出力画像が少し違い正確に比較する事ができなかったが、CPU 上での実行時間の平均が 0.155 秒に対し、GPU 上での実行時間の平均は 0.008 秒であり、約 20 倍の速度向上を得る事ができた。大量のデータを持つ画像処理に対し、GPU を用いた高速化を実現することができた。更なる高速化を実現するために、コアレッシングなど意識したプログラミングにより高速化を実現することが可能だと考えられる。今後、ラプラシアンフィルタの出力が CPU、GPU で同じものを得るために検討を行う。

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崎教授に深く感謝いたします。また、共同研究において、機会とともに実験データを与えて下さった株式会社ケーデーイーの三宅氏、西田氏に感謝いたします。様々な面で貴重な助言や励ましを下さった研究室の皆様に深く感謝いたします。

参考文献

[1]井上誠喜　他：C 言語で学ぶ実践画像処理,オーム社,2010
[2]岡田賢治：CUDA 高速 GPU プログラミング入門,秀和システム,2010
[3]NVIDIA：CUDA テクニカルトレーニング vol1：CUDA プログラム入門,
http://www.nvidia.co.jp/docs/IO/59373/VolumeI.pdf
[4]大山佳宣：OpenMP を用いた画像処理プログラムの高速化,
立命館大学電子情報デザイン学科卒業論文,2010
[5]手越一雄：はじめての C 言語,技術評論社,2000
[6]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
大学電子情報デザイン学科卒業論文,2006
[7]松崎裕樹、山崎勝弘：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,
平成 18 年度情報処理学会関西支部　支部大会講演論文集
C-09, p.187-190, 2006