2015 年度　卒業研究論文

画像と図形のエッジ検出について

岡山理科大学

総合情報学部　情報科学科

Ｉ１２Ｉ０４０　小名川　薫儀

Ｉ１２Ｉ１０１　渡邊貴裕

目次

1. はじめに

近年，スマートフォンやタブレットのアプリケーションや，自動車の自動運転技術の分野で歩行者や対向車を識別するのに，図形認識の技術が使用されている．そのようなアプリケーションではフローチャートやダイアグラム作成の補助機能を，利用し手書きした図形から正確な図形を識別している．自動運転技術では前方の自動車や歩行者，白線を瞬時に識別し，車間距離を保つための処理などに使用されている．このように，図形や物体を識別することの重要性が増していると考えられる．それらの処理がどのように行われているのかに注目し，処理されるのかに関心を持ち，画像内から図形認識を行い，どのように認識されるのか調べる事にした．

現在は, RGB 各８ビットの情報を有する画像によるカメラが一般的に使用されているが, 近年は RBG 各１６ビットの画像や, RGB 各チャンルが８ビット以上の画像を撮影可能なカメラが存在している．そして，各チャンネルのビット数が増加するほど，より高精度で細部まで描画を行う事が可能になると考えられる．各チャンネルのビット数の異なる画像を用意し，ビット数の多い画像を用いて図形認識を行えばより精度の高い図形検出が行えると考え，ビット数を変えることにより図形検出に　にどう影響するのかに興味をひかれた．

本研究では，円形の図形検出に焦点を当て，画像内にそのような対象物が存在する場合，どのように認識されるのかについて，調べている．また, 通常使用されているＲＧＢ各８ビットの画像（以降８ビット画像と呼ぶ）以外にもさらにビット数の多いＲＧＢ各１６ビット画像（以降１６ビット画像と呼ぶ）で同様の処理を行う場合, どのような違いがあるのかを実験により調べている．

2. デジタル画像のビット数

デジタル画像において，画像の各ピクセル（画素）で表現可能な色の階調数はビット数により表わされている．ビット数が多ければ多いほど使用可能な階調数が増加し，より鮮明で詳細な表現が行える．画像のビット数をｎとすると階調は2^nレベルであり，ビット数が８ならば2^8＝２５６階調，１６ビットの場合は６５５３６階調まで表現可能である

3. 図形認識に用いた画像と方法

比較に用いた画像は本研究室内で撮影したものを利用する．この画像はカメラの仕様上ＲＧＢ各１４ビットのＲＡＷデータ（画像）である．このＲＡＷデータはデジタルカメラで撮影した色の情報そのままのデータであるため，互換性の高い，　８/１６ビットＴＩＦＦ画像に変換して用いることにする（図1）．

3.1. ハフ変換

ハフ変換とは，画像内から直線または円を検出する方法の一つである．基本原理は 1962 年に Paul Hough により提唱されている．画像の高さと幅をそれぞれ直角座標(x, y)に変換「，直行座標上に存在する点(x, y)を円の中心点（CenterX，CenterY）と半径(Radius)の三次元空間に変換し，直行座標上に存在する点(x, y)を全て検索することで円形の対象物を探している．

直行座標上に円が存在する場合，円は点(x, y)，円の中心点（CenterX，CenterY）と半径（Radius）によって表される（図 1)．点(x, y)を，CenterX, CenterY, Radius の 3 変数による組み合わせに変換し，以下の式 3.1 を用いて Radius の値を求める．

$$\mathrm{Radius}^2 = (\mathrm{x\text{-}CenterX})^2 + (\mathrm{y\text{-}CenterY})^2 \qquad (3.1)$$

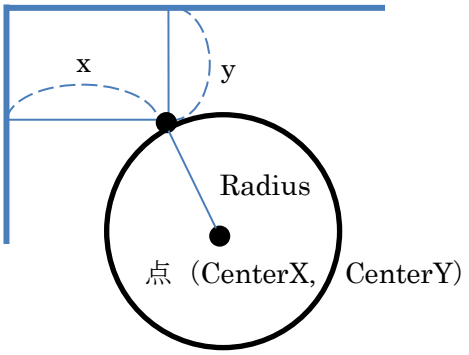

図 1　直角座標上に存在する円の構成

式 3.1 を用いて，直角座標上に存在する点（x, y）を新しい 3 次元空間（CenterX, CenterY, Radius）に変換すると，直角座標上に存在する点は，3 次元の一枚の面に対応する．直角座標上に多数の点が存在すると，それらを線で連結させることにより，一つの円として並ぶことになる［2］.

円検出のハフ変換は，画像上にある全ての点(x, y)を検索し，加えて式__を用いて中心点（CenterX, CenterX）を検索する為円形の検出に非常に処理時間が掛かる．よって処理を高速化する為にあらかじめ隣接する中心点同士の距離，半径の値を定めて検出を行う［3］.

4. 検出方法と画像について

用意した８/１６ビットＴＩＦＦ画像を比較の容易なグレースケール画像に変換する.撮影に使用したデジタルカメラはRBG各１６ビットRAW画像データで保存される．しかし，RAW画像データはイメージセンサーで記録した情報の為，JPEGやTIFF形式の様にR,G,B各チャンネルの配列で保存されていない．一度RAW画像データを8/16ビットそれぞれで保存可能なTIFF画像に変換し，画像内から図形検出を行う.比較を行う際には，フィルタ処理を施さず全てグレースケール画像を検出した場合と，平滑化処理を施した場合，画像に先鋭化処理とを施して検出する場合の三通りについて，円形図形の検出を行い比較する事にした.使用環境は以下に示した通りである.

使用環境

- Microsoft VisualStudio2010(言語:C++)
- OpenCV ver.2.4.10 (BSD license)
- Image Data Converter Ver.4
- SONY α7R デジタルカメラ （14ビットRAW出力）
- ミラーレスデジタル一眼

開発環境で利用したOpenCV（Open Source Computer Vision Library）とは，Intel社が開発したオープンソースライブラリである．画像処理,画像解析および機械学習等といった処理が行える機能が多数実装されており，C/C++,java,Pythonといった言語と組み合わせて利用することができる.

4.1. TIFF画像

TIFF（Tagged Image File Format）画像形式とは，1986年にMicrosoftとAldus（現在はAdobe）によって開発された画像データのファイル形式のことである．記録形式の異なる様々なファイルを保存でき，拡張子として「.tif」あるいは「.tiff」が付加される.

TIFFファイルには，画像データのヘッダ（先頭）部分に記録形式についての属性情報が記録される．これに依拠してデータの処理方式が決定されるので，保存時には解像度や色数，符号化の方式などの形式によらず自由にデータを収めておくことでが可能となる.。

それと同時に，Windowsで用いられるTIFFファイルをMacintoshで読み込むことができるなど，アプリケーションソフトへの依存度も低減される．ただし，記録形式のバリエーションがあまりに多いために，互換性を持たない記録形式も含まれることもある．なおTIFFファイルは，圧縮を伴うLZW型と無圧縮型とに分かれる．また，詳細な画像情報を持つため再現性は高いが、ファイルサイズは大きくなることが難点である.

4.2. RAW画像データ

RAWデータとは，デジタルカメラなどの画像形式の一種で，カメラのイメージセンサーから得たデータをそのまま羅列したものである．デジタル一眼レフカメラやミラーレス一眼カメラなどで利用できる形式で，いわゆるコンパクトデジタルカメラにはRAWデータでの出力機能は無いことが多い.

RAWデータはカメラのイメージセンサーの構造に依存した形式のため，それ自体をコンピュータで表

示・加工・印刷などすることはできず JPEG, TIFF など汎用的な画像形式に変換する必要がある．この作業・変換処理のことをネガフィルムから印画紙への焼き付けになぞらえて「現像」ということがある．RAW データはメーカーや機種によって異なる独自形式であるため，現像処理はカメラに添付されたメーカー製の専用ソフトでなければできないことが多い．

5. フィルタ処理

円形の図形を検出しやすくする為に，画像にフィルタ処理を施し比較を行った．フィルタ処理とは，画像内に含まれる不要なものを取り除き，目的とする情報を取り出す処理の事である．フィルタ処理は，オペレータやカーネルと呼ばれる矩形の重み付けの為の行列を用いて，積和演算を行う線形フィルタ処理と，注目する画素の最大値と最小値を求める非線形フィルタの二種類がある．

線形フィルタ処理は，f(x,y)は処理を施す入力画像，g(x,y)は処理を施した出力画像，h(m,n)はオペレータとなる重み付けを行う行列とm,n番目の値，オペレータのサイズは(2w＋1)・(2w＋1)とし，式(5.1)として表す．例としてエッジ検出を行うオペレータの処理手順を示す[図 3]．

$$g(x,y) = \sum_{n=-w}^{w} \sum_{m=-w}^{w} f(x+m, y+m) \cdot \mathrm{h}(m,n) \qquad (5.1)$$

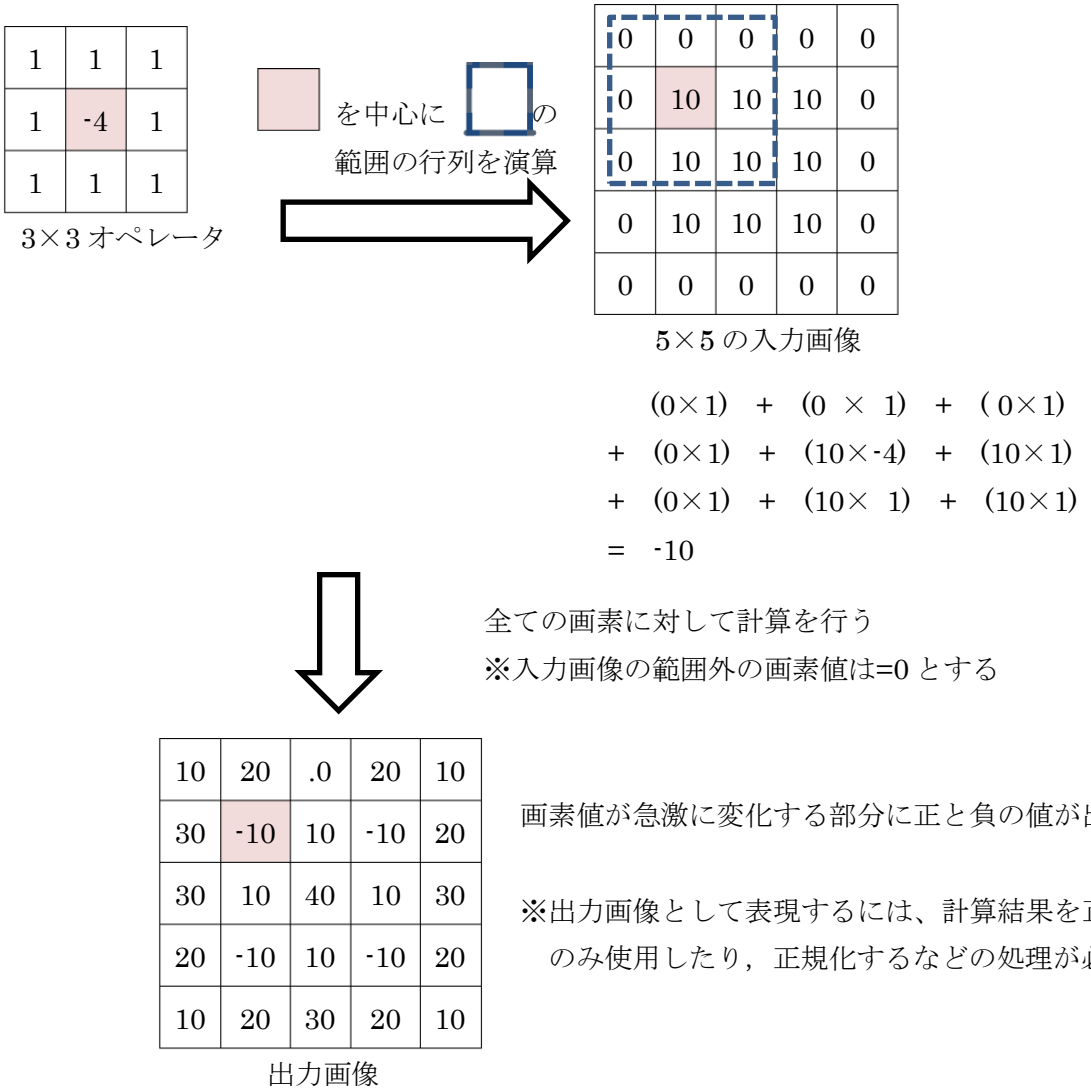

図 2　線形フィルタ処理の例

式（5.1）から分かるように，画像の各画素値 I(x,y)を，その画素周辺の値を活用して変換を行う処理の為，出力画像は入力画像と同じサイズとなる．

非線形フィルタ処理は，画素の最大値や平均値や最小値を求めるかによって，処理の結果が異なる．今回は平滑化と線形化の，二つのフィルタ処理を使用している．

5.1. 平滑化

平滑化とは，画像全体を滑らかにする処理である．注目する画素の値と周囲の画素値を用いて平均化処理を行い，画像内のノイズを減らし滑らかにする．平滑化では移動平均フィルタとガウシアン（加重平均）フィルタがよく利用されているが,今回はガウシアンフィルタを使用して平滑化を行っている．

ガウシアンフィルタとは，注目画素に近いほど大きな重みを付けた平滑化処理の事である．単純な平均化オペレータに比べ，平滑化を行いながらも入力画像に近い平滑化処理を行うことが出来る．ガウシアンフィルタのオペレータは図 2 に示す.

一般的な画像では，注目する画素に隣接する画素の輝度値は，注目する画素に近い場合が多いが，注目する画素から遠くなるほど，輝度値の差は大きくなる．ガウシアンフィルタはこの事を考慮し，注目する画素に近いほど，平均化を計算するときの重みを大きくし，注目する画素から遠くなるほど重みを小さくなるようにガウス分布の公式（5.1）を用いて計算を行っている．

$$f(x,y)=\frac{1}{2\pi\sigma^2}\cdot\exp\left(\frac{x^2+y^2}{\sigma^2}\right) \tag{5.2}$$

$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{9}$
$\frac{2}{16}$	$\frac{4}{16}$	$\frac{2}{16}$
$\frac{1}{16}$	$\frac{2}{16}$	$\frac{1}{16}$

3×3 ガウシアンフィルタ

$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{6}{256}$	$\frac{24}{256}$	$\frac{36}{256}$	$\frac{24}{256}$	$\frac{6}{256}$
$\frac{4}{256}$	$\frac{16}{256}$	$\frac{24}{256}$	$\frac{16}{256}$	$\frac{4}{256}$
$\frac{1}{256}$	$\frac{4}{256}$	$\frac{6}{256}$	$\frac{4}{256}$	$\frac{1}{256}$

5×5 ガウシアンフィルタ

図 3 ガウシアンフィルタのオペレータ

ここでσは標準偏差，（x､y）は注目画素からの距離とする.標準偏差σが小さい場合は平滑化の度合いが小さく，標準偏差σが大きくなるにつれて平滑化の度合いが増えていく．平均化フィルタとガウシアンフィルタの処理例は以下に示す通りとなる（図 3）．今回は現画像に近い平滑化を行う 3×3 のガウシアンフィルタを使用し，平滑化を行った.

(a) 入力画像　（b）３×3 ガウシアンフィルタによる平滑化　（c）５×５ガウシアンフィルタによる平滑化

図 4　ガウシアンフィルタを用いた平滑化

5.2. 先鋭化フィルタ

先鋭化フィルタとは，画像内の輝度値の変化を強調する処理のことである.輝度値が急激に変わる部分を強調することにより図形の輪郭を検出することが出来る．画像の中から特徴や図形を検出する為にこ

とが多い．ここでは一般的なソベルフィルタ,ラプラシアンフィルタ、キャニーフィルタを使用している．

5.2.1.ソベルフィルタ

ソベルフィルタは注目する画素と周囲併せて９個の画素に対し乗算を行い，画像の輪郭を検出するフィルタである．左右に隣接する画素値の差を求める一次微分と平滑化を組み合わせ，輪郭の検出を行う．それぞれ縦方向と横方向のオペレータがあり（図 5)，今回は横方向の使用している．ソベルフィルタを施した画像は以下に示す（図 7)．

-1	0	1
-2	0	2
-1	0	1

（ａ）横方向オペレータ

-1	-2	-1
0	0	0
-1	-2	-1

（ｂ）縦方向オペレータ

図 5　ソベルフィルタのオペレータ

5.2.2.ラプラシアンフィルタ

ラプラシアンフィルタは注目する画素と周囲併せて 9 個あるいは 25 個を用い，画像内の輝度値の変化量が極端に大きなっている部分を強調し，輪郭を検出するフィルタである．ソベルフィルタは異なり，上下左右に隣接する画素値の差を求める二次微分を利用し輪郭を抽出する．それぞれ上下左右の微分を行う４方向のオペレータと，上下左右に加え斜め方向の微分を行う８方向のオペレータがあり（図 6)，今回は４方向のオペレータを使用している．ラプラシアンフィルタを施した画像は以下に示す（図 8)．

0	1	0
1	-4	1
0	1	0

（ａ）４方向オペレータ

1	1	1
1	-8	1
1	1	1

（ｂ）８方向オペレータ

図 6　ラプラシアンフィルタのオペレータ

5.2.3. キャニーフィルタ

キャニーフィルタはガウシアンフィルタとソベルフィルタを組み合わせ，輪郭を検出するフィルタである．入力画像に対し，以下の４つの処理を施し輪郭を抽出する．

(1)　**ガウシアンフィルタを施し全体をぼかし，画像内に含まれるノイズを減らす.**

(2)　**ソベルフィルタを適用し輪郭を抽出する.**

(3)　**ガウシアンフィルタにより太くなった輪郭を細線化する.**

(4)　**細線化した輪郭の連結性を上げる為，高い値と低い値の二種類の閾値化を用いた二値化を行う**

ラプラシアンフィルタを施した画像は以下に示す（図 8)．

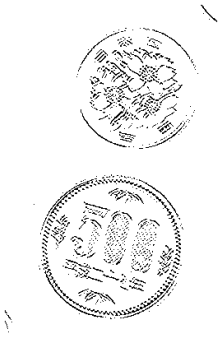

図 7　ソベルフィルタ

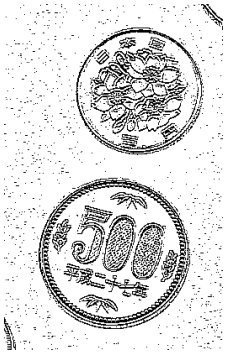

図 8　ラプラシアンフィルタ

図 9　キャニーフィルタ

6. 検出結果

検出結果を以下にまとめる．原画像（図 10）内で円形の図形は 21 個存在し（表 1）, 検出結果は処理順とビット数により異なるものとなった（表 2，3）. それぞれの検出結果と画像中の青丸と赤丸により示している（図 11，図 12，図 13，図 14, 図 15, 図 16, 図 17，図 18, 図 19, 図 20）.
青丸は画像内の円形と一致あるいは円を検出しているもの, 赤丸を誤検出したものを示している. なお, 先鋭化処理を施した画像には，処理結果を分かりやすくする為に白黒反転処理を施している.

表 1　画像内の円形

画像内の図形数	２１

表 2　8 ビットグレースケール画像

	検出数	誤検出数
８ビット平滑化無し	２３	2
８ビット平滑化処理	２１	0
８ビットソベル	0	0
８ビットラプラシアン	１９５	１７５
８ビットキャニー	7	0

表 3　16 ビットグレースケール画像

	検出数	誤検出数
１６ビット平滑化無し	２２	1
１６ビット平滑化処理	２１	0
１６ビットソベル	0	0
１６ビットラプラシアン	１９３	１７４
１６ビットキャニー	6	0

図 10　ハフ変換に使用した現画像

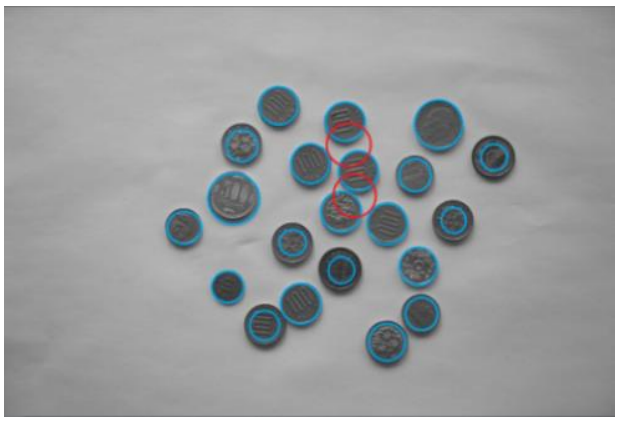

図 11　8 ビットグレースケール画像

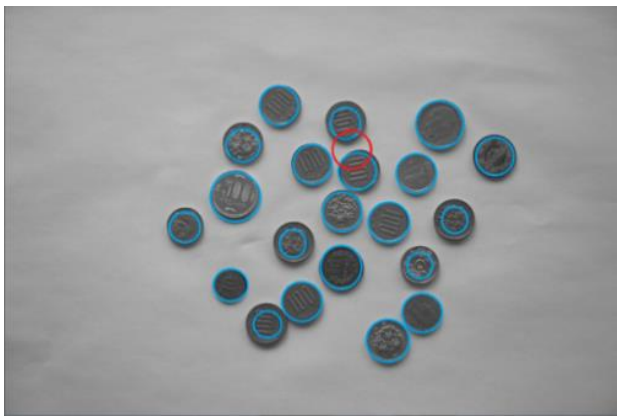

図 12　１６ビットグレール画像

図 13　８ビット平滑化画像

図 14　１６ビット平滑化画像

図 15　８ビット画像ソベルフィルタ処理

図 16　１６ビット画像ソベルフィルタ処理

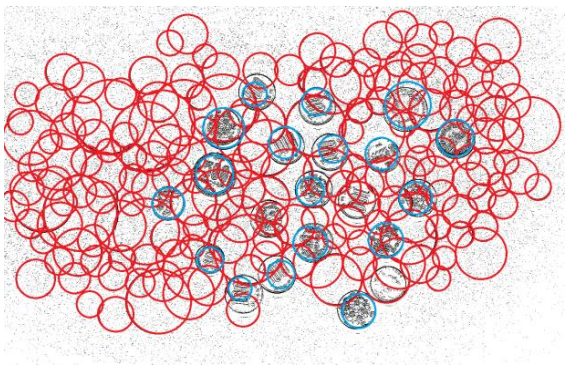

図 17　８ビット画像ラプラシアンフィルタ処理

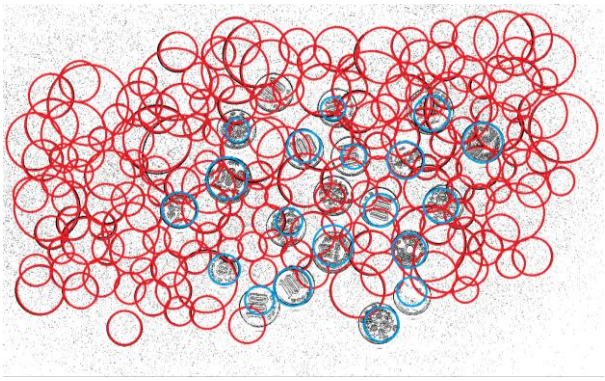

図 18　１６ビット画像ラプラシアンフィルタ処理

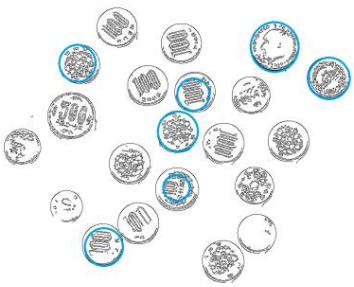

図 19　８ビット画像キャニーフィルタ処理

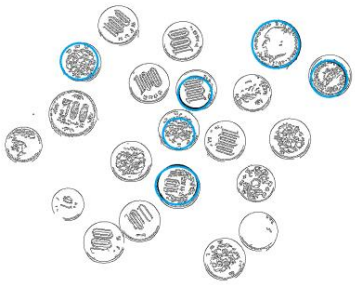

図 20　１６ビット画像キャニーフィルタ

ビット数の異なる８/１６ビットグレースケール画像をそれぞれ平滑化を行わず円検出をした場合，８ビット画像では検出数２３個で誤検出数は２個，１６ビット画像では検出数２２個で誤検出数は１個と，どちらも誤検出数は少なくほぼ正確に円形に検出している（図 11，12）（表２，３）．

平滑化を施した場合では，画像内からノイズが減った為，どちらも検出数は２１個で誤検出は無く，正確に円を検出している（図 13，14）（表２，３）．

ソベルフィルタを施した場合では，ソベルフィルタが画像内から輪郭を抽出しているが，縦方向の輪郭を抽出しておらず，正確に輪郭を描画できていない為，どちらも検出数が０個となった（図 15，16）（表２，３）．

ラプラシアンフィルタを施した場合では，ソベルフィルタと比べて 4 方向からの輪郭抽出をしている為，正確に輪郭を描画しているが，その分背景のノイズを描画している．その為，ノイズを円と認識し，８ビット画像では検出数が１９５個したがって誤検出数が１７５個，１６ビット画像では検出数が１９３個で誤検出数が１７４個となり，どちらも誤検出数が大幅に増加している（図 17，図 18（表２，３）．

キャニーフィルタの場合では，８ビット画像では検出数が７個，１６ビット画像では検出数が６個とソベルフィルタと比べて検出数は増加したものの，２１個中の１/３程しか検出できておらず，全体的に検出数は少ない結果となった（図 19，図 20）．

7. まとめ

実験の結果,図 11 と図 12 や図 13 と図 14 のように，平滑化処理してからもしくはそのままで検出を行った場合，8 ビットグレースケール画像よりも 16 ビットグレースケール画像の方が誤検出は少ないが,全体的に検出数が減少することが分かった.

先鋭化フィルタを施した場合では，図 15 と図 16 や図 19 と図 20 のように検出される円の個数が減少する場合や，図 15 と図 16 のように全体の誤検出数が増加する結果となり，ハフ変換を行う場合では先鋭化フィルタとその単純な組み合わせは実用的でない事も分かった.

今回，比較に使用した画像は，先鋭化処理が行いやすい背景が白の状態で撮影した為、そのまま検出した場合と平滑化処理を施した場合では誤検出が少なくなり，先鋭化を施した場合では検出数の減少や誤検出が増加し，検出誤差が減少しこのような結果になった．撮影時に背景の色を変更した場合や，撮影する角度を変更すれば，今回とは異なる検出結果になると考えている.

今後の課題として，グレースケール画像以外に，ＲＧＢカラー画像を用いた円検出の比較や,異なる背景や正面以外の方向で撮影した画像で，今回と同じ条件で図形検出を行った場合，どのように違いが出るのかを調べたい.

謝辞

本論文を執筆するにあたり多くの御指導，御助言を下さいました岡山理科大学大学教授総合情報学部情報科学科，澤見英男教授に心から御礼申し上げます.

参考文献

[1] OpenCV による画像処理入門， 小林正直，上田悦子，中村恭之， 講談社，2014

[2] OpenCV と Visual C++による画像処理と認識
http://homepage3.nifty.com/ishidate/opencv_9/opencv_9.html

[3] Hough 変換による画像からの直線や円の検出
http://codezine.jp/article/detail/153