社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－CVIM－143 (10)
2004／3／5

局所領域内のchain code変換法を用いた高精度な直線検出法

近藤 早苗* 成瀬 正*

画像の局所領域にHough変換を適用して線素を検出し，それを延伸することで線分を検出する手法が提案されている[1]．この手法は，従来のHough変換に比べて高速に直線を検出でき，延伸処理を行うことによりHough変換の際に生じる量子化誤差をある程度吸収できる特徴を持つ．しかし，Hough変換を利用しているため，量子化誤差の影響は避けられない．一方，chain code変換法[2]は，原理的に投票空間の量子化誤差の影響を受けない手法である．しかし，完全なchain code変換法は$O(N^4)$のコストがかかり，入力画像に直接適用するのは現実的ではない．そこで，簡略chain code変換法を局所領域に適用することで，Hough変換法と同程度以下の処理時間で，高精度に直線を検出する手法を検討した．本論文では，その手法を提案するとともに，直線検出精度，処理時間について実験的検討結果について報告する．

Line detection method using chain code transform in local area

Sanae Kondo* Tadashi Naruse*

It has been proposed a method that detects line segments in each local area of the given image, and gets the lines by extending the line segments over the whole image [1]. The features of the method are; 1) it can detect lines at high speed compared with basic Hough transform, 2) it is robust for the quantization errors. On the other hand, the chain code transform(CCT) gives a quantization error free method for line detection [2]. A reduced CCT with $O(N^3)$ computation time is proposed while a complete CCT takes $O(N^4)$ computation time for size $N \times N$ image. In this paper, we proposed a new line detection method which applies the reduced CCT to the local area instead of Hough transform. This method detects precise lines in a reasonable time compared with local area Hough transform method given by [1]. We show the effectiveness of the method by the empirical study.

1 はじめに

2値画像から直線を検出する手法として，Hough変換がある．Hough変換は，ノイズや隠蔽に強く，安定して直線を検出できるという特徴がある．しかしながら，以下のような問題点[3]がある．

1. 投票空間を表すための大きなメモリ空間が必要である．
2. 画像中の各点ごとに1本のHough曲線を描くため，計算量が高くなる．
3. 投票度数の多寡を線分らしさの評価基準としているため，短い直線検出は苦手である．
4. 投票空間中のセルと画像空間中のディジタル直線とは，1対1には対応していないため，ディジタル直線を構成する画素数分の投票度数が対応するセルに正確に蓄積されない．

*著者所属 愛知県立大学
*Affiliation Aichi Prefectural University

問題点1，2などの計算量およびメモリの効率化に関しては，確率的アルゴリズムの導入，並列化，局所的な特徴を用いる方法，階層的な投票を行う方法，投票空間の分解，軌跡の高速描画[4]などの手法が提案されている．

さらに，問題点1，2に加えて，問題点3に対する解決は，村上らの局所領域にHough変換を用いた直線検出法[1]によって試みられている．この手法は，画像を小領域に分割し，それぞれの小領域に対してHough変換を適用して，線分候補（以下，線素と呼ぶ．）を検出する．そして，線素が局所領域外に伸びている可能性がある場合は延伸処理を行って，線分を探索する手法である．したがって，小さな画像領域にHough変換を適用するため，計算量，メモリ量ともに標準的なHough変換の数分の1から10分の1程度までに抑えることができる．また，小さな画像領域を対象としているため，短い線分も検出することが出来る．しかし，Hough変換を使用

しているため，問題点4は依然解決されていない．

問題点4に対しては，成瀬らによるchain codeを用いた直線検出法[2]などにより，投票度数に偏りを生じずに，高精度に直線を検出する手法が提案されている．しかし，この手法は，ディジタル平面上のすべてのディジタル直線を数え上げる手法であるため，計算量，メモリ量ともに従来のHough変換よりコストが高く，直接画像全体に適用するのは現実的ではない．

そこで，本論文では，画像を小領域に分割することにより，高速に直線を検出する手法にchain code変換法を適用することで，高速性と短い線分も検出できるという特徴を維持したまま，更に高精度に直線を検出する手法を提案する．そして，投票実験，実画像を用いた直線検出実験，および局所領域にHough変換を用いる直線検出法との比較実験を行い，良好な結果を得た．本論文では，その結果を報告する．

2 局所領域にHough変換を適用する直線検出法

ここでは，局所領域にHough変換を適用する直線検出法[1]の基本原理，アルゴリズム，特徴と問題点を簡単に述べる．以降，この手法を**LHT**と呼ぶことにする．

2.1 基本原理

画像を小領域に分割し，任意の一つを選択して，この領域にHough変換を適用して線分候補を検出する．線素が小領域外に伸びている可能性がある場合は，延伸処理により延長する．この処理を各小領域に適用して，画像全体から直線検出を行う．

2.2 アルゴリズム

以下に基本アルゴリズムを示す．

[Step1 画像の分割] 画像全体を小領域(局所領域)に分割し，与えられた戦略に従って関心のある位置に局所領域を設定する．

[Step2 局所的な線分（線素）の検出] Step1で設定した局所領域内の画素(特徴点)に対して，通常のHough変換を適用して直線を検出する．ここで検出された直線を線素と呼ぶ．

[Step3 線分候補存在領域の検出] 線素及びそれを延伸した線分の含まれ得る領域(線分候補存在領域と呼ぶ．)を画像全体の中から検出する．

[Step4 線分の検出（検証処理）] 定められた戦略に従って線分候補存在領域内で線分の位置をより詳細に検証し，線分を検出する．(検出された線分の点をすべてラベル付けすることにより，他の局所領域において投票対象点から外す．このラベル付け操作により，処理の効率化を図ることが出来る．)

[Step5 終了判定] 与えられた条件を満足，あるいは，画像全体を処理したら終了する．

[Step6 局所領域の移動] 与えられた戦略によって局所領域を動かし，Step2～Step5の処理を繰り返す．

2.3 特徴と問題点

LHTには，以下のような特徴と問題点がある．

[特徴]

- 計算量，メモリ量を，標準的なHough変換の数分の1から10分の1程度に抑えられる．
- 短い線分も検出可能である．
- 線分候補存在領域で検証処理を行うことによりHough変換の際に生じる量子化誤差をある程度吸収できる．
- ラベル付け操作を行うことにより，投票対象点を削減できる．

[問題点]

局所領域に基づく直線検出法では，注目している小領域内で線素を検出し，小領域内で発生する誤差分を考慮しながら（誤差を考慮した範囲が線分候補存在領域である．），他の領域まで線素を延伸する．したがって，小領域内で検出された線素の誤差が小さい必要がある．しかし，Hough変換を利用しているため，直線検出精度には限界がある．延伸処理により入力画像中の直線と線素のずれが拡大して，画像全体から線分を検出できない場合がある．

3 chain code変換法

ここでは，chain code変換法の基本原理，アルゴリズム，特徴と問題点を簡単に述べる．この手法をCCTと呼ぶことにする．

3.1 基本原理

文献 [2] によれば，大きさ $N \times N$ の画像における傾きが $0 \leq \tan\theta < 1$ のディジタル直線は

$$y = \frac{m}{n}x + p + \frac{q}{n}, \quad where \quad 0 \leq \frac{m}{n} < 1, \quad 0 \leq q < n \qquad (1)$$

の形で表現される．ただし (1) 式において，m, n, p, q は整数であり，整数点 x における y の値は，$\lfloor y \rfloor$ (y のフロアー) として表わされる．また，m,n は互いに素であり，$n < N$ である．p と $\frac{q}{n}$ は，切片の整数部と小数部を表す．[1]

このとき，分離可能な傾きの数は N に属する Farey 数列の項数で与えられる．Farey 数列の項数は，$O(N^2)$ で与えられる．したがって，(1) 式で表される直線は，$O(N^2 \times N \times N) = O(N^4)$ の異なるものが存在することになる．

これを投票の基づく手法で分離する場合，投票空間は傾き（F と表す），整数切片（p と表す），小数切片（q と表す）の軸を持つ空間となる．これを $F-p-q$ 空間と書くことにする．この空間において，(1) 式の直線は完全に分離される（完全 CCT）．この時，投票アルゴリズムは次節のようになる．

3.2 CCT アルゴリズム

まず，次の事実に注意する．

$\frac{m}{n} \leq \frac{a}{b} < \frac{m+1}{n}$ なる傾き $\frac{a}{b}$ を持つ基準直線のディジタル直線は，画像平面上の点 (n, m) を通る．

[完全 CCT アルゴリズム]

入力画像サイズを $N \times N$ とし，傾き $\frac{m}{n}$ が $0 \leq \frac{m}{n} < 1$ の範囲のとき，以下の手順で処理を行う．それ以外の傾きに対しては，座標軸を変換して同様の手順を適用する（投票空間は 4 つに分けられる）．

[Step1] 画像空間の点 (n, m) が投票対象点ならば，$\frac{m-p}{n} \leq \frac{a}{b} < \frac{m+1-p}{n}$，$m-n < p \leq m$，$q = 0$ なる投票空間上の各点に投票する．

[Step2] Step1 で投票した投票空間の各点を $(\frac{a}{b}, p, 0)$ で表すと，$1 \leq q < b$ の範囲の各整数 q に対して，

$\frac{an+q}{b} \geq m-p+1 = m-(p-1)$ なら点 $(\frac{a}{b}, p-1, q)$ に投票する．

$\frac{an+q}{b} < m-p+1 = m-(p-1)$ なら点 $(\frac{a}{b}, p, q)$ に投票する．

[Step3] 画像空間上の各投票対象点に対して Step1, 2 を繰り返す．

しかし，完全 CCT アルゴリズムは計算量，メモリ量ともに $O(N^4)$ かかる．そこで，簡略アルゴリズムを考える．小数切片は式 (1) からわかるように，$\frac{q}{n}$ の平行移動を表しているから，直線を分離する上でその寄与は最も小さい．したがって，小数切片 q 軸を削減し第一に考え，$F-p$ 空間への投票を行う（以降，簡略 CCT と呼ぶ．）．簡略 CCT は，計算量，メモリ量ともに $O(N^3)$ に減らすことができる．(他の削減方法は，[2] を参照．) 次章では，この簡略 CCT を用いる．

簡略 CCT を用いても，$\frac{q}{n}$ は平行移動の項であるから，画像の傾きに関する検出精度には影響を与えない．したがって，十分な精度で直線を検出でき，計算量とメモリ量が削減できる．

3.3 簡略 CCT アルゴリズムに対する座標軸の変換

傾き $\frac{m}{n}$ が $0 \leq \frac{m}{n} < 1$ の範囲以外のときの座標軸の変換方法を以下に示す (図 1)．投票空間を添字 0 ～3 を付けて表す．

- $0 \leq \frac{m}{n} < 1$ の時（投票空間 0）
  画像空間の点 (n, m) が投票対象点ならば，$\frac{m-p}{n} \leq \frac{a}{b} < \frac{m+1-p}{n}$，$m-n < p \leq m$ なる投票空間 0 上の各点に投票する．

- $1 \leq \frac{m}{n} < \infty$ の時（投票空間 1）
  直線の傾きが正になるように座標軸を置き換えれば，$0 \leq \frac{a}{b} < 1$ の Farey 数列 $\frac{a}{b}$ を使うことができる．そのために座標を $(n, m) = (m, n)$ に置換し，$\frac{n-p}{m} \leq \frac{a}{b} < \frac{n+1-p}{m}$，$n-m \leq p < n$ なる投票空間 1 上の各点に投票する．

- $-\infty \leq \frac{m}{n} < -1$ の時（投票空間 2）
  同様に，座標を $(n, m) = (y\_size-1-m, n)$ に置換し，$\frac{n-p}{y\_size-1-m} \leq \frac{a}{b} < \frac{n+1-p}{y\_size-1-m}$，$m+n+1-y\_size < p \leq n$ なる投票空間 2 上の各点に投票する．

- $-1 \leq \frac{m}{n} < 0$ の時（投票空間 3）
  同様に，座標を $(n, m) = (n, y\_size -$

[1] 文献 [2] では，傾きを chain code で表現した．したがって，本論文ではこの手法に基づく直線検出法を CCT と呼ぶことにする．

$1-m$) に置換し，$\frac{y\_size-1-m-p}{n} \leq \frac{a}{b} < \frac{y\_size-1-m+1-p}{n}$，$y\_size-1-m-n \leq p < y\_size-1-m$ なる投票空間 3 上の各点に投票する．

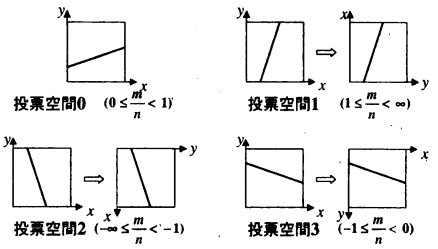

図 1: 座標軸の置換方法

3.4 特徴と問題点

LCCT には，以下のような特徴と問題点がある．

[特徴]

- 投票空間に偏りを生じない．
- 完全 CCT を用いると，すべてのディジタル直線を完全に分離でき，高精度に直線を検出できる．
- 4 つの投票空間に投票を行うので，**投票**と**ピーク検出**部分では並列化が可能である．
- 簡略アルゴリズムを用いることで，十分な検出精度を維持しながら，計算量とメモリ量を削減できる．

[問題点]

完全 CCT を用いると，計算量，メモリ量ともに $O(N^4)$ となり，画像全体に適用するには現実的でない．また，簡略 CCT アルゴリズムを用いても，計算量とメモリ量ともに $O(N^3)$ であり，N が大きくなると，標準的な Hough 変換よりもコストが高くなる．

4 局所領域に chain code 変換法を適用する直線検出法

局所領域には，第 3 節で述べた簡略 CCT を用いる．第 2 章の**[Step2　局所的な線分（線素）の検出]** での直線検出法が，簡略 CCT であることと，**[Step3　線分候補存在領域の検出]** の求め方が異なる以外は，LHT のアルゴリズムと基本的には同じである．以降，この手法を**LCCT** と呼ぶことにする．

4.1 線分候補存在領域の検出

線素は F - p 投票空間でのピーク値を与えるパラメータから逆変換によって生成されるが，簡略 CCT を用いているため，そこには小数切片削減に伴う誤差が付随する．この誤差分を考慮して，線素及びそれを延長した線分が存在し得る画像領域を特徴点の連続性を考慮して求める．

[Step1] F - p 投票空間でのピーク値を与えるパラメータから逆変換によって得られた線素の方程式から，局所領域内の [線素の始点]，[線素の終点] を求める．

[Step2] 得られた線素の方程式に従い，始点と終点から他の領域に線素を延ばしていく．ここで，小数切片 q における誤差は，直線を y 軸に平行移動した分の誤差であるので，高々 ± 1 である．したがって，延伸処理時の注目画素の ± 1 分の画素に線素の存在し得る．つまり，幅 3 画素の範囲の画像領域が線素の存在し得る領域となる．

[Step3] 入力画像の直線の特徴点が，拡大した領域内に存在する限り，拡大操作を繰り返す．gap 長で指定された回数の範囲内に特徴点と一致する点が見つからなければ，拡大操作を終了する．このときの直線の端点が線分候補存在領域の境界線となる．つまり，[始点-1]，[終点-1]，[始点+1]，[終点+1] を結ぶ平行四辺形の領域が線分候補存在領域となる．

5 実験

本手法の有効性を確認するため，LHT と LCCT を用いて直線検出精度，処理時間に関する二つの性能比較実験を行った．5.1 節では，局所領域内で適用する Hough 変換と簡略 CCT に対して，全画素からの投票を行った．そして，投票結果から投票空間の構造と直線分離能力に対して考察を行った．

また，5.2 節では，LHT と LCCT に対して実際に実画像を用いて直線検出実験を行い，比較検討を行った．

5.1 投票実験

全画素が特徴点で，画像サイズが 16×16 の2値画像から，Hough 変換と簡略 CCT に対して投票を行った．ここで，両手法ともに原点を画像の左上とした．簡略 CCT による投票結果を図 2，3 に示す．CCT は，3 章で示した 4 つの投票空間に分けられるが，図 2 は，投票空間 0 と投票空間 3 を合わせて表示し，図 3 は，投票空間 1 と投票空間 2 を合わせて表示している．

また，図 4 に Hough 変換に対して投票を行った結果を示す．投票空間の分割数は，後藤らの論文 [5] に基づき定めた．

両手法ともに投票数の最大値は，理論上の最大値である 16 である．簡略 CCT は，投票空間の上面が高さ 16 で，ひし形の平坦な面になる．これは，投票空間にひずみが現れず，正確に投票されていることを示している．一方，Hough 変換は投票数に偏りが生じ，投票空間にひずみが出来る．これらの図からわかるように，Hough 変換に比べて簡略 CCT が，直線分離能力が高いことがわかる．

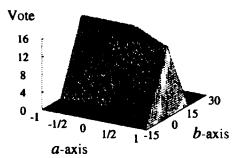

図 2: CCT($y = ax + b$)

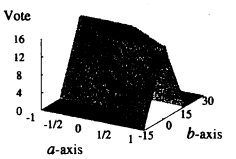

図 3: CCT($x = ay + b$)

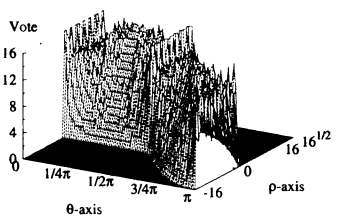

図 4: Hough 変換

5.2 直線検出実験

提案手法（LCCT）の有効性を確認するため，実画像を用いて 2 章の手法（LHT）との比較実験を行った．両手法を直線検出精度，処理時間の観点から比較検討する．

5.2.1 実験方法

実験には，雑音を含まない，異なる直線 11 本が描かれている入力画像 1(図 5 : 64×64) と，雑音を多く含み直線の長さ，傾きともに異なる直線が描かれている入力画像 2(図 6 : 256×256) を用いる．入力画像 2 は検出したい直線の情報が一部欠落している．実験では，注目している特徴点の 8 近傍に 1 点も特徴点が存在しない（孤立点）場合，その点を雑音とみなし，前処理により除去する．

投票空間の分割数は，LHT については文献 [5] に基づき，LCCT については文献 [2] に基づき定めた．アルゴリズムの制御パラメータには，局所領域の大きさの他に，投票数のピーク検出の閾値，線素の長さの閾値，線分の長さの閾値などがある．局所領域の大きさ以外のパラメータについては，出力画像が目視で最も良くなるように設定した．得られた出力画像を図 7〜14 に示す．

これらの出力画像に対して，比較評価を行った．評価方法は被験者 17 人による目視評価である．また，入力画像 1 については次式による一致度の評価も行った．

$$\text{一致度} = \frac{\text{一致した点の数}}{\text{入力画像の特徴点の総数}} \times 100\ (\%)$$

結果を表 1，2 に示す．目視評価は，被験者が 10 段階で評価の評価を行い，それを平均したものである．数字が高いほど評価値がよい．

5.2.2 結果と考察

入力画像 1 に対する直線検出処理では，局所領域の画像サイズを両手法とも 16×16 とした．LCCT の出力画像（図 7）は，おおむね入力画像とほぼ同じ場所に直線を検出できている．しかし，LHT（図 8）は，11 本中 10 本は精度よく直線を検出できているが，残り 1 本の直線が線分の途中までしか検出できていない．また，表 1，2 より，一致度，目視評価，処理時間すべて LCCT の方が良い結果となった．

LCCT の一致度が低下する主な要因は，線分候補存在領域内での検証処理による誤差である．検証処理は，以下のような処理である．

[Step1] 線分候補存在領域の端点境界上の始点候補と終点候補を 1 点ずつ選ぶ．

表 1: 局所領域内の chain code 変換法 (LCCT)

入力画像	出力画像	一致度 (%)	目視評価	検出本数	実測時間 [sec]
図 5	図 7(N_L=16)	87.09	7.88	11	0.0060
図 6	図 9(N_L=16)	-	9.25	50	0.0868
	図 11(N_L=32)	-	8.18	38	0.3670
	図 13(N_L=64)	-	7.06	44	0.8380

表 2: 局所領域内の Hough 変換法 (LHT)

入力画像	出力画像	一致度 (%)	目視評価	検出本数	実測時間 [sec]
図 5	図 8(N_L=16)	83.53	6.53	11	0.0091
図 6	図 10(N_L=16)	-	7.31	51	0.1368
	図 12(N_L=32)	-	6.00	64	0.2363
	図 14(N_L=64)	-	4.94	69	0.5810

CPU : Celeron1.2GHz

[Step2] 選ばれた 2 点を通る線分（2 点を通る直線の方程式を適用）を考え，その線分上に存在する点の数を数える（一致度と呼ぶ）.

[Step3] 始点候補と終点候補の組み合わせをかえ，最も一致度の高くなる線分を，最終的な線分として検出する.

この処理の**[Step2]** で 2 点を通る直線の方程式を用いることが，一致度を低下させる要因である．ディジタル平面において，異なる 2 点を通るディジタル直線は，ただ一つではなく複数本存在する．そのため，検出された線分は，線分上の点の並びがわずかにずれることがある．これが，LCCT の一致度を低下させる主な要因である．一方，LHT の一致度が低下する原因は，LCCT と同様の要因の他に次のような要因がある．すなわち，LHT は Hough 変換の投票空間の量子化誤差に起因する小領域内の線素の誤差が CCT に比べて大きい．そのため，線素を他の領域に延伸する処理において，入力画像中の直線と線素のずれが拡大し，一致度が低下する.

また，入力画像 2 に対して，局所領域のサイズを 16×16 , 32×32 , 64×64 と変えて実験を行った．結果を表 1,2，図 9〜14 に示す．図 9〜14 に示す通り，局所領域のサイズを同じサイズに設定したときの両手法の出力結果は，LCCT の出力画像は精度よく直線を検出できているが，LHT は直線の途切れが多くみられる．これも，入力画像 1 と同じ理由

[入力画像]

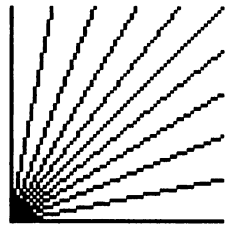

図 5: 入力画像 1(N_G=64)

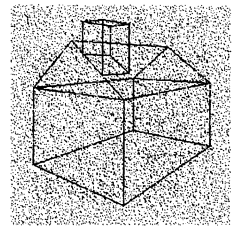

図 6: 入力画像 2(N_G=256)

[出力画像]

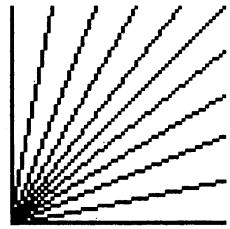

図 7: LCCT(N_L=16)

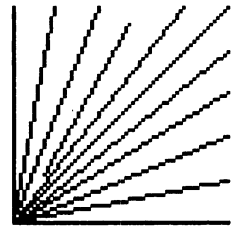

図 8: LHT(N_L=16)

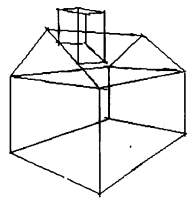

図 9: LCCT(N_L=16)

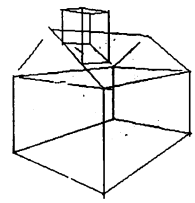

図 10: LHT(N_L=16)

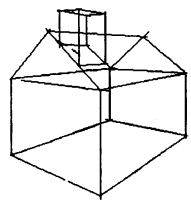

図 11: LCCT((N_L=32)

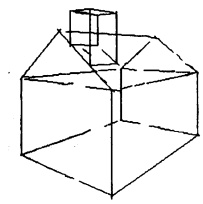

図 12: LHT(N_L=32)

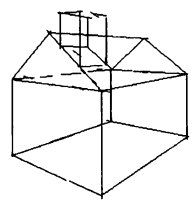

図 13: LCCT(N_L=64)

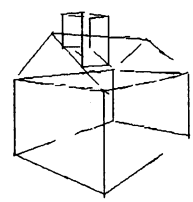

図 14: LHT(N_L=64)

で，局所領域内で検出された線素の誤差が大きいため，直線の途中までしか検出できないからである．表 1,2 の目視評価でも LCCT の方が評価が高い結果となった．

両手法の処理時間を詳しく分析するため，処理を [投票]，[度数ピークの検出]，[直線の検出] の 3 つの部分に分けて検討する．入力画像 2 に対する出力画像（図 7〜14）の各処理の処理時間を計測した．結果を表 3,4 に示す．

局所領域の画像サイズ (N_L とする) が 16×16 のとき，[投票] の実測時間を比較すると，LCCT の方が投票空間が大きいにも関わらず，処理時間は LCCT の方が高速となった．Hough 変換は，各特徴点に対して，Hough 変換式（投票する 2 次元配列の番号）を θ の分割数（$2\pi N_L$）回計算し，2 次元配列に θ の分割数（$2\pi N_L$）回投票する．(ただし，Hough 変換（$\rho = x\cos\theta + y\sin\theta$）の三角関数はあらかじめ table に格納している．したがって，毎回三角関数の計算はしていない．) 一方，簡略 CCT は各特徴点 $(x_i, y_i)(i = 0, ..., \text{特徴点の総数} - 1)$ に対して，CCT 変換式（投票する 2 次元配列の番号）を $(2x_i + N_L)$ 回計算し，2 次元配列に Farey 数列の項数（$(\frac{3}{\pi^2}N_L^2 + O(N_L \log N_L)) \times 4$）回投票する．したがって，明らかに，

$2\pi N_L > 2x_i + N_L$

であるため，画像サイズに関わらず Hough 変換式を計算する回数は，CCT 変換式を計算する回数より多い．逆に，投票を行う回数は，

$2\pi N_L < (\frac{3}{\pi^2}N_L^2 + O(N_L \log N_L)) \times 4$

であるため，簡略 CCT の方が Hough 変換よりも計算回数が多い．(ただし，局所領域の画像サイズを 4×4 などのように極端に小さくするのは禁止的である [1].)

また，Hough 変換式や CCT 変換式を計算する単位演算コストは，投票する（2 次元配列を +1 する）単位演算コストよりも明らかに高い．したがって，局所領域の画像サイズが 16 や 32 などの小さい場合は，θ の分割数と Farey 数列の項数の差は小さい．そのため，一票入れるコストの総和の差よりも，変換式の計算する総和の差が大きくなり，LCCT の方が高速に [投票] 処理を行うことが出来る．

逆に，画像サイズが 64 以上になると，θ の分割数と Farey 数列の項数の差が大きくなる．その結果，変換式を計算する総和の差よりも，一票入れるコストの総和の差が大きくなり，LHT の方が高速に [投票] 処理を行うことが出来る．

次に，両手法の [度数ピークの検出] の実測時間を検討する．5.1 節の実験結果からわかるように，LHT は LCCT に比べ直線分離能力が低い．したがって，LCCT と同程度の出力結果を得るためには，LHT のピーク検出に用いる閾値を下げる必要がある．その結果，表 3,4 が示すように，LHT の検出されるピーク数は LCCT に比べて増大する（理由①）．これが，LHT の計算時間を増す要因である．一方，LCCT は投票空間の大きさが計算時間を増す要因である．その結果，局所領域の画像サイズが 16×16 などの小さい場合，両手法の投票空間の大きさの差は小さい．したがって，LCCT の投票空間の大きさに起因する処理時間よりも，LHT のピーク数の多さに起因する処理時間が増し，LCCT の方が高速にピーク検出することが出来る．

しかし，画像サイズが 32, 64 では投票空間のサイズが，LCCT が LHT に比べ約 3.56 倍，7.13 倍と大きくなる．そのため，LCCT が投票空間全体を走査するのに時間がかかり，LHT の方が高速にピーク検出できる．

[直線の検出] 処理にかかる時間は，線分候補存在領域のサイズと線分候補として検出される直線の本数に大きく依存する．線分候補存在領域のサイズは，LCCT の方が大きい．しかし，線分候補として検出される直線の本数は，表 3,4 から分かるように，LCCT の方が少ない．また，線分の長さの閾値などにも依存する．したがって，実験の範囲ではどちらの手法が高速に直線を検出できるかは言いがたい．

[投票]，[ピーク検出]，[直線の検出] を足し合わせた直線検出処理全体の実測時間としては，局所領域のサイズが 16 などの小さいとき，LCCT の方が高速に直線検出できる．32 や 64 などに大きくなると LHT の方が高速に直線検出できるという実験結果を得た．

また，表 3,4 より，局所領域内で線分候補（線素）として検出された直線のうち，実際に直線として検出される本数の割合は，LCCT は約 41.90〜55.07% である．それに対して，LHT は約 7.74〜21.16% と非常に低い結果となった．これは，LHT が理由①により，ピーク値が多数出現し，線分候補が多数検出されるのに対して，LCCT ではそれが抑えられることによる．これらの実験的検討から分かるように，LCCT は LHT よりも高精度に直線を検出でき，また，局所領域が小さい場合には LHT よりも高速に

表 3: 入力画像 2 に対する LCCT の実験結果 (実測時間の単位 : sec)

local size 投票空間のサイズ	実測 時間	投票			度数ピークの検出		直線の検出		
		時間	投票対象点数	F_L	時間	ピーク数	時間	線分候補	検出本数
$16(73 \times 4, 32)$	0.0868	0.0104	4425	73×4	0.0578	140	0.0126	110	50
$32(309 \times 4, 64)$	0.3670	0.0483	4502	309×4	0.1738	237	0.1019	69	38
$64(1229 \times 4, 128)$	1.8380	0.5370	5006	1229×4	0.8698	2019	0.2020	105	44

表 4: 入力画像 2 に対する LHT の実験結果 (実測時間の単位 : sec)

local size 投票空間のサイズ	実測 時間	投票			度数ピークの検出		直線の検出		
		時間	投票対象点数	K_L	時間	ピーク数	時間	線分候補	検出本数
$16(101, 55)$	0.1368	0.0336	4418	101	0.0870	971	0.0141	241	51
$32(202, 110)$	0.2363	0.0801	4843	202	0.1085	3903	0.0470	335	64
$64(403, 219)$	0.5810	0.1735	5002	403	0.2353	7852	0.1662	891	69

※ CPU : Celeron1.2GHz
入力画像 2 に，ノイズ除去をした後の特徴点の総数は，5481 である．
ラベル付け操作により，投票対象点が削減される．
F_L を局所領域での Farey 数列の項数とし，K_L を局所領域での θ の分割数とする．

直線を検出できる手法であると言える．

6 おわりに

本論文では，局所領域に簡略 chain code 変換法を適用する手法を提案した．そして，簡略 chain code 変換法と Hough 変換の投票空間の構造を検討し，両手法の特徴を明らかにした．さらに，直線を高精度に検出できることを投票実験と直線検出実験によって，実験的に示した．また，局所領域の画像サイズ，入力画像の画像サイズによっては，LHT より高速に直線を検出できることを実験的に示した．

今後は，chain code 変換法と Hough 変換の投票空間の微細構造を更に詳しく検討し，直線検出に用いるパラメータ値の最適な設定法を検討する．

謝辞

本研究を進めるにあたり，御指導と御助言を賜った中京大学情報科学部輿水大和教授，愛知県立大学情報科学部村上和人助教授に感謝する．

参考文献

[1] 村上和人，成瀬正：「局所領域内の Hough 変換を用いた高速な直線検出法」，信学論 (D-II),vol.j83-D-II,no.3,pp.918-926,march.2000.

[2] 成瀬正，村上和人，高橋友一，小林幸雄：「ディジタル直線を記述するためのチェインコードが持つ性質とディジタル直線の構造について」，情報処理学会 CVIM 研究会資料 (2001.11)

[3] 松山隆司，輿水大和：「Hough 変換とパターンマッチング」，情報処理，Vol.30，No.9，pp.1035-1036 (1989.9)

[4] 松山隆司，久野義徳，井宮淳：「コンピュータビジョン　技術評論と将来展望」，新技術コミュニケーションズ，pp.149-150 (1997)

[5] 後藤英昭，阿曽弘具：「ハフ変換におけるパラメータの効率的なサンプリング間隔」，信学論 (D-II),vol.J81-D-II,no.44,pp.697-705,April 1998』