

局所並列処理で構成された 画像処理アルゴリズムに関する研究

慶應義塾大学大学院 理工学研究科 開放環境科学専攻

平成 18 年度

岩片 智

目次

論文要旨	1
第1章 序論	2
1.1 はじめに	3
1.1.1 画像処理の分類	3
1.1.2 画像処理の歴史	3
1.2 文字抽出・看板抽出の諸研究	5
1.3 並列計算機とは	7
1.3.1 並列計算機の分類	7
1.3.2 アレイプロセッサ	8
1.4 並列画像処理ハードウェアアーキテクチャの諸研究	10
1.4.1 IMAP	10
1.4.2 人工網膜 LSI	11
1.4.3 超高速ビジョンチップ	11
1.4.4 指紋認証チップ	12
1.5 視覚装置ハードウェアについて	13
1.5.1 視覚装置の概要	13
1.5.2 視覚装置のハードウェアアーキテクチャ	14
1.5.3 視覚装置の有効性	16
1.6 視覚装置のための局所並列画像処理アルゴリズムに関する制約	20
1.6.1 視覚装置における PE の仕様	20
1.6.2 ブロック処理との相違点	21
1.6.3 PE の統合処理	21
1.6.4 局所並列画像処理アルゴリズムの目的	21
1.7 本研究の目的と位置付け	23
1.8 本論文の構成	24
第2章 局所並列処理のための文字抽出アルゴリズム	25
2.1 はじめに	26

2.2	文字抽出アルゴリズム	29
2.2.1	概要	29
2.2.2	近傍定義	29
2.2.3	輝度値変換	30
2.2.4	ベース画像生成過程	31
2.2.5	ノイズ抽出過程	36
2.2.6	文字候補抽出過程	40
2.2.7	文字抽出過程	43
2.2.8	ノイズ除去過程	47
2.3	計算機シミュレーション	49
2.3.1	情景画像からの文字抽出	49
2.3.2	パラメータ変更による抽出率の変化	54
2.3.3	難易度の高い画像からの文字抽出	55
2.3.4	PE ごとの計算量および通信量の測定	58
2.4	考察	60
2.5	むすび	62
第3章	局所並列処理のためのピクトグラム検出アルゴリズム	63
3.1	はじめに	64
3.2	駅構内画像からのピクトグラム検出アルゴリズム	66
3.2.1	概要	66
3.2.2	近傍定義	67
3.2.3	平滑化過程	68
3.2.4	多色分類過程	69
3.2.5	四角形抽出過程	71
3.2.6	輝度値変換過程	76
3.2.7	二値化過程	76
3.2.8	ピクトグラム分類過程	77
3.3	計算機シミュレーション	87
3.3.1	ピクトグラムの抽出	87
3.3.2	PE の計算量および通信量の測定	92
3.3.3	東急線駅構内のピクトグラム検出	93
3.4	考察	95
3.5	むすび	97

第4章 結論	98
謝辞	100
参考文献	101
付録	109
付録A章 近傍処理による情景画像からの文字抽出アルゴリズム	109
A.1 概要	110
A.2 詳細	111
付録B章 文字抽出シミュレーションの入出力画像	112
B.1 情景画像からの文字抽出	113
B.2 難易度の高い画像からの文字抽出	116
付録C章 駅構内ピクトグラム検出シミュレーションの入出力画像	119

本論文では、視覚障害者支援を目的とした並列画像処理ハードウェアのためのアルゴリズムに関する研究について述べる。

画像処理の分野では古くから処理アルゴリズムの研究が行われている。画像処理では、一度に大量の画素情報を扱わなければならないため、並列に処理するためのハードウェアやその並列性を活かしたアルゴリズムが多く研究されている。近年は半導体技術の進展によりハードウェアの集積度が高まり、画像処理用途の組み込みハードウェアの実現も可能になってきている。本論文は、そのような組み込みハードウェアに実装可能な画像処理アルゴリズムとして、局所並列処理のみで構成されたアルゴリズムについて述べたものである。

第1章では、本研究の背景および目的について述べる。

第2章では、情景画像から文字を抽出するアルゴリズムについて述べる。提案アルゴリズムは、画像から文字領域のみを抽出し、代読サービス等を通じた視覚障害者支援への応用を目的としたものである。アルゴリズムは局所並列処理のみで構成されている。近傍画素情報のみを用いた二値化により画像の照明条件が悪い場合でも抽出可能となっている。また、画素を重心方向へ移動させることにより文字領域の特徴量を求めるなど、単純な計算の繰り返しで情景画像から文字領域を抽出している。計算機シミュレーションを行い、抽出精度や特性、ハードウェア実装時の計算量に関して検討し、良好な結果を得ている。

第3章では、情景画像からピクトグラムを検出するアルゴリズムについて述べる。ピクトグラムとは「図記号」のことであり、ピクトグラムを抽出することにより、視覚障害者の歩行支援への応用を目的としている。ピクトグラムの枠に着目し、正方形を抽出した後、ピクトグラムを検出するという手法をとっている。第2章で提案したアルゴリズムと同様に、ピクトグラム検出アルゴリズムも単純な局所処理の繰り返しで構成されている。さらに、駅構内画像のピクトグラムの検出を行うアルゴリズムに発展させ、11種類のピクトグラムの特徴を簡単な局所処理のみで求め、検出している。これにより、本論文で提案する局所並列処理を容易に応用することが可能であることを示している。

第4章は結論であり、本論文で得られた成果をまとめている。

第 1 章

序論

本論文は、局所的な画像処理のみにより文字やピクトグラム (図記号) の抽出を行う研究について述べるものである。このアプローチをとることにより、株式会社エッチャンデスの視覚障害者支援を目的とした並列視覚処理装置 [1] にも実装可能である。

本章では、本論文に関連する研究の歴史と背景について概観する。

1.1 節・1.2 節では画像処理のソフトウェア的な観点から、1.3 節・1.4 節ではハードウェア的な観点からそれぞれ述べる。

1.1 節では、画像処理についての概要と歴史について述べ、1.2 節では、画像から文字や図形を抽出する研究の歴史や現状について紹介する。1.3 節では、並列コンピュータの歴史について述べ、1.4 節では、並列画像処理ハードウェアアーキテクチャについて述べる。1.5 節では、本研究で扱う視覚装置ハードウェアについて述べ、1.6 節では、視覚装置ハードウェアのためのアルゴリズムについて述べる。

1.7 節では、本研究の位置付けについて述べ、1.8 節では、本論文の構成を述べる。

1.1 はじめに

画像処理とは、画像の解析・認識・加工する技術を対象とする分野である。各種の画像からそこに描かれている対象物や特徴を抽出し、その画像がどのような性質を持っているのか解析したり、その画像が何を表しているのか認識したりする。また、入力画像をより見やすい画像への変換を行う [2][3]。ここでは、画像処理の分類について述べた後、画像処理の歴史についてまとめる。

1.1.1 画像処理の分類

画像処理方式を処理範囲により分類すると、局所的処理と大域的処理に分類することができる [4]。

局所的処理は、入力画像中の局所画素だけが出力画像の一画素に影響を与えるような処理である。単純な処理が中心であり、その例として濃度変換やエッジ抽出が挙げられる。

大域的処理は、入力画像の広い範囲の画素が出力画像内の一画素に影響を与えるような処理である。画素間の相関を利用した処理が可能であり、直交変換処理やテクスチャ解析などがあげられる。

また、処理の時間的な実行形態によって画像処理を分類すると、逐次処理と並列処理に分類できる。

逐次処理は各画素ごとに順次処理を実行する形態であり、手順は簡単であるが、処理時間がかかる。一方、並列処理は画像内の全ての画素を一括して処理する形態であり、手順は複雑であるが処理時間が短いという特徴を持っている。

局所並列処理は、画像を複数の領域に分割し、各領域ごとに並列処理を行う処理形態である [5]。本研究では、この局所並列処理によって画像処理アルゴリズムを構築している。

1.1.2 画像処理の歴史

表 1.1 に画像処理に関する出来事をまとめる。

ディジタル画像処理は、1960 年代に始まった [6]。この時期の主な対象は、宇宙探査用の人工衛星であり、処理技術としては画質改善、画像復元、補正などが中心であった。

二次元パターン処理の分野では、コンピュータの処理能力の限界もあり、この時代は文字認識が研究の中心であった。そして後によく使われるようになる、細線化・しきい値処理・形状特徴抽出などの多くの手法やアルゴリズムの研究が進んだ。

1970 年代初めには局所並列処理や二値画像処理の基本的な手法が次第に確立され、基礎となる画像変換も体系化が進んだ。デジタル画像処理の基本的手法の大部分がこの時代に研究されている。

1980 年代には、それまでに大きく発展した画像処理技術がさまざまな分野で実用化され始めた。医用画像処理では、MRI(Magnetic Resonance Imaging) 機器をはじめさまざまな CT(Computerized Tomography) 機器が登場した。産業応用では目視検査に代わる機械による製品の形状検査だけでなく、仕分けや組み立ての自動化、農水産物の加工・分類の自動化、原子炉での作業など多くの領域で画像処理とロボットが組み合わせて導入された。また、新しい概念、手法としてフラクタル、カオス、ファジィ、ニューラルネットワークが登場し、これらを画像処理に応用した研究も始まった [3]。

1990 年代には、これらのさまざまな技術を結合し、さらに新しい可能性を開くものとして仮想現実 (バーチャルリアリティ) が脚光を浴びるようになった [7]。これは三次元の仮想空間をリアルタイムで体験できるものである。

2000 年代には、バイオメトリクスによる個人認証 [8] やデジタルデータ用電子透かし技術 [9]、自動運転のための車両・道路認識技術 [10]、大容量データベースでのメタデータ自動作成や画像検索 [11] などが研究され注目を集めている。

表 1.1 画像処理の歴史 (抜粋)

年	できごと
1950	文字読み取り研究開始 (米)
1955	数字 OCR 発売 (米)
1960	文字読み取り研究開始 (日本) 手書き文字読み取り研究開始
1965	FFT アルゴリズム開発 [12]
1966	OCR 開発 (IBM)
1970	かな・漢字読み取り研究開始
2002	静脈認証システム開発 (富士通)

1.2 文字抽出・看板抽出の諸研究

本研究の対象は局所並列処理で構成されたアルゴリズムである。ここでは、まず処理形態に関わらず、文字やマークの抽出手法に関する研究について述べる。

既に述べた通り、画像処理の研究が始まった当初から物体抽出の研究は行われてきた。

文字読み取りに関する研究は1950年代から始まっており、1955年にはOCR(Optical Character Reader)が発売されているが、これは文書画像のような単純な画像中の文字を読み取るものである。現在の文字認識の精度は活字であればほぼ100%となっているが、情景画像のような複雑な画像の場合は文字領域を的確に切り出さなければならない。文字抽出やマーク抽出は文字認識やロボットビジョン、ITS(高度道路交通システム)などの分野において非常に重要な処理であり、盛んに行われている[13]-[18]。

文字抽出に関する研究は、地図のような比較的簡単な背景の画像から文字領域を抽出する手法[19]-[22]の他、複雑な情景画像から文字領域を抽出する手法など、これまでも多く開発されている[23]-[37]。マーク抽出に関する研究においてもITSの推進により、道路標識の自動認識などにおいて実用化が進んでいる[38]-[40]。

文字やマーク領域の代表的な抽出手法は、抽出対象の特徴に着目すると

- (1) 色情報を利用した手法
- (2) 形状情報を利用した手法
- (3) 空間周波数を利用した手法

に分類できる。(1)の手法として、文献[41]では、文字領域の色分布特性を調べ、クラスタリングによって色空間を分割して代表色を求め、文字領域を抽出している。文献[42]では、 $L^*a^*b^*$ 空間と色相を利用してクラスタリングし、分割された領域に対してクラスタを統合させ、看板領域の絞り込みを行っている。文献[31]では、明度を利用して背景領域から看板領域を切り出している。また、文献[43]は文字領域抽出法ではないが、パターン認識においてヒストグラムを利用した高速な検索手法であり、良好な結果が得られる手法である。

(2)の手法としては、文献[13]のように2値化を基本とした抽出手法や文献[29]のようにエッジ情報を利用した手法、文献[44]のように線分密度やハフ変換を利用して文字列を抽出する手法がある。また、文献[45]のように、モルフォロジー演算を領域抽出に応用した手法もある。

(3)の手法としては、文献[24]のように空間周波数に着目し、フーリエ変換を利用して高周波成分を含む画像を切り出して文字列候補とし、形状や大きさの条件を用い

て文字列を抽出する手法や、文献 [28] のようにウェーブレット変換を利用した手法がある。

以上の手法は、PC 等の一般的なコンピュータに実装され、ソフトウェアとして評価が行われているものが多い。十分な精度が得られたとしても、実用化した際にシステムを小型化することは困難であることが多い。視覚障害者支援や ITS 等で利用される車載装置は低コストでの実現性や低消費電力であることなどが求められる。

これらを実現するために、画像処理専用プロセッサや並列処理プロセッサの開発が行われている。次節ではまず並列計算機について述べ、並列に画像処理を行うプロセッサについて述べる。

1.3 並列計算機とは

本論文では並列画像処理装置である“視覚装置”を対象としたアルゴリズムに関する研究について述べる。そのため、ここでは並列計算機について簡単にまとめる。並列計算機の分類について述べ、その中で本研究で対象としているアレイプロセッサについて述べる。

1.3.1 並列計算機の分類

M.J.Flynn は並列計算機だけではなく、計算機全体の分類を行っている [46]。この方法では、計算機を命令の流れの数とデータの流の数に着目して、SISD(Single Instruction stream Single Data stream)、SIMD(Single Instruction stream Multiple Data streams)、MISD(Multiple Instruction streams Single Data stream)、MIMD(Multiple Instruction streams Multiple Data streams) の 4 つに分類している。

SISD:単一命令流単一データ流

単一の命令を順番にメモリから取り出して、単一のデータ流に対して処理を施す、いわゆる通常の計算機である。

SIMD:単一命令流複数データ流

多数の演算装置に対し、単一の命令をブロードキャストし、複数のデータに対して同一の処理を行う方式。単純な処理に対しては、低コストで高い並列性を得ることができることから、画像処理等の専用マシンとしては有利であるが、すべての演算装置が単一の命令しか実行できないため、複雑な処理を行う汎用マシンとしては不利な点が多い。

MISD:複数命令流単一データ流

単一のデータ流に対して、同時に複数の命令が実行される方式。

MIMD:複数命令流複数データ流

独立の命令実行能力を持った複数の CPU が、互いにデータを交換しながら、複数のデータ流に対して処理を行う方式。それぞれのプロセッサが独立に動作することから

汎用性、柔軟性が高いが、プログラミングが困難な点が多い。並列計算機の多くのマシンがこのカテゴリに入る。

上記のような命令の流れとデータの流れをもとにした分類法の他にも、さまざまな分類法が存在する。以下に述べるように、プロセッサやメモリの構成をもとにした分類法もある。

マルチプロセッサ 共有メモリを持つ MIMD 型の計算機であるが、人によっては、プロセッサがたくさんある計算機全体を指す場合もある。

マルチコンピュータ 共有メモリを持たない MIMD 型の計算機を指す場合が多いが、マルチプロセッサ同様、並列計算機全体を指す場合もある。

アレイプロセッサ アレイ (配列) 状にプロセッサが並んでいることから、SIMD 型の計算機を指す場合が多い。画像処理アルゴリズムを並列に実行することに適しているとされるアーキテクチャである。

本研究で対象としている“視覚装置”のハードウェアアーキテクチャはアレイプロセッサの構造をしている。以降ではアレイプロセッサについて述べる。

1.3.2 アレイプロセッサ

アレイプロセッサは画像処理やニューラルコンピューティングの分野で注目を集めている。この方式は、単純な構造のプロセッサ要素 (PE: Processing Element) を多数接続し、その上で問題の並列性を活かして高速な処理を行おうとするものである [47]。

$N \times N$ のデータを $N \times N$ 個のプロセッサで処理を行うアレイプロセッサの研究は古く、1950 年代終りに二次元画像処理のアーキテクチャとして使用したシステムが提案されている。小規模ではあるが、ハードウェアが実際に作成され始めたのは 1960 年代初めに発表された SOLOMON コンピュータである。その後、CLIP-4 や DAP 等が開発されてきたが、アレイ規模が小さい、あるいは高価である等の理由から広く利用されるには至らなかった [47]。

しかしながら、LSI 技術の発展により極めて高いコスト性能比を得られるようになった。これは、アレイプロセッサが

- 新たに開発すべき LSI が 1 種類ですみ、構成が規則的なため設計が容易である。
- LSI を規則的に並べるだけで容易に PE アレイを構成できる。

- 同一 LSI を大量に使用するので、量産効果を活かすことができる。

という特徴を持っているからである。

この LSI 技術を本格的に利用した最初の大規模なシステムは、Goodyear Aerospace 社の MPP(Massively Parallel Processor) と NTT のプロトタイプ AAP1(Adaptive Array Processor) である。MPP は 2×4 の PE アレイを持つ LSI チップを用いて 128×132 のプロセッサアレイを、AAP1 は 8×8 の PE アレイを持つ LSI により 256×256 のプロセッサアレイを実現した。

さらに、1985 年に Thinking Machine 社がコネクションマシンを、1986 年に NTT が AAP-2 を発表している。両者とも 65536 個の 1 ビット PE アレイと大容量のローカルメモリから構成された SIMD 型のプロセッサアレイ計算機である。

近年では、NEC 社が 1 次元プロセッサアレイ型の動画造処理ボード IMAP シリーズを開発・販売している。次節ではこの IMAP の他、近年の画像処理用チップのアーキテクチャについて述べる。

表 1.2 画像処理ハードウェアアーキテクチャの諸研究

機関	名称
NEC	IMAP[48]-[51]
三菱電機	人工網膜 LSI[52]-[54]
東京大学 石川ら	S^3PE [55]-[60]
NTT	指紋認証チップ [61]
大阪大学 八木ら	シリコン網膜 [62]
東北大学 小柳ら	人工網膜チップ [63]
日本プレシジョン	高速対象追跡ビジョンチップ
浜松ホトニクス	インテリジェントビジョンシステム
Texas Instruments	Davinci
Xilinx	Virtex
IP Flex	DAPDNA

1.4 並列画像処理ハードウェアアーキテクチャの諸研究

並列画像処理チップに関する研究・開発はさまざまな機関により行われている。主なものを表 1.2 にまとめる。表 1.2 の中でも、本節では小規模な PE を多数実装し、並列処理により高速化をはかる画像処理アーキテクチャとして、NEC 社の IMAP シリーズ、三菱電機社の人工網膜 LSI、東京大学の超高速ビジョンチップ、NTT の指紋認証チップについて述べる。

1.4.1 IMAP

IMAP[48] は 2M ビットの SRAM と 64 個の 8 ビットプロセッサを 1 チップに集積した LSI であり、1100 万トランジスタを $15.1 \times 15.6 \text{mm}^2$ の面積に集積しており、8 ビット演算時に 3.84GIPS の最大性能を持つ。

IMAP アーキテクチャは次の特徴を持つ。

1. 1 チップに多数の SIMD プロセッサを集積する。
2. プロセッサ間で通信するための信号線による、ピン数増大を防ぐために 1 次元プロセッサアレイ構成をとる。複数の IMAP を 1 次元上に接続することによってマルチチップ構成が容易に実現できる。
3. メモリを同一チップ上に集積する。個々のプロセッサごとにメモリを持たせるの

ではなく、大容量メモリの横に 1 次元プロセッサアレイを集積する。

4. 高速データ入出力用のシリアルデータ入出力を持つ。

IMAP-VISION は PCI バス用のピーク性能 10GIPS を有する動画像処理ボードである。256 個の 8 ビットプロセッサ (PE) を 1 次元に接続した PE アレイ部、ビデオインタフェース部、16MB の画像メモリ、これら全体を制御する RISC プロセッサが 1 枚の PCI のフルサイズボードに搭載されている [49][50]。

さらに、その後継の IMAP-CE は 128 個の VLIW 型の PE を集積し、100MHz 動作時にはシングルチップで 51.2GOPS、16 枚のチップを接続した最大構成時では、819.2GOPS のピーク性能を有する。各 PE は 2KB のローカルメモリ、24 個の 8 ビット汎用レジスタ、2 つの 1 ビットマスクレジスタ、ALU および乗算器を持つ。また、入出力はシフトレジスタを用い、1 行分の画像データを順次シフトすることにより行われる。

1.4.2 人工網膜 LSI

三菱電機では人間の網膜機能を模擬して画像処理を行う人工網膜 LSI の開発を行っている [52]-[54]。人工網膜 LSI と他の画像センサ (CCD や他の CMOS 画像センサ等) との違いは、人工網膜 LSI にはオンチップの画像処理を可能にする画素構造があることである。これにより後段のハードウェア処理が軽減され、システム規模を小さく収めることが可能になる。人工網膜 LSI は、感度可変受光セル (VSPC) と呼ばれる、空間光変調機能を合わせ持つ受光素子で構成されている。制御信号により光電流の方向を変えることで、入力する 2 次元画像と画像処理機能選択信号の積和をチップ上で計算して画像処理結果としての出力が可能である。

1.4.3 超高速ビジョンチップ

東京大学の石川らは、超高速ビジョンチップ S^3PE (Simple and Smart Sensory Processing Element) の研究を行っている [55]-[60]。従来のビジョンチップは、集積化が容易なアナログ回路を用いて固定の処理を行っていたが、近年、半導体集積技術の進歩によりデジタル回路の PE を採用したプログラマブルなビジョンチップの研究が行われている。 S^3PE は PE に汎用な処理回路を用いながら高集積化を目指したアーキテクチャである。PE は 2 次元アレイ上に搭載され、1 つの PE が 1 画素を扱う。各 PE は光センサ、24 ビットのメモリ、1 ビットの ALU を持ち、上下左右の PE と接続されており、PE 間の通信が可能となっている。

また、PE 同士をレジスタを介さずに結合することにより、従来の超並列 SIMD プロセッサが苦手とした非局所演算を効率的に行える [58]。

1.4.4 指紋認証チップ

NTT では、指紋認証 LSI の研究を行っている [61]。一般的な指紋認証システムは、指紋センサと認証を行うマイクロプロセッサからなり、プロセッサを持たない機器に組み込むのは困難であった。指紋認証チップは、認証を行う回路を、ピクセル単位に分割した並列構成 (ピクセル・アレイ) としている。センサ素子とピクセル・アレイを積層化し、センサ素子で読み取った指紋を直下の処理回路により照合する構成となっている。このため、指紋認証チップは携帯機器のみでなく、CPU 等を搭載しない機器への導入が可能となっている。

1.5 視覚装置ハードウェアについて

本節では本研究で対象としている、株式会社エッチャンデスの、局所並列画像処理用視覚装置ハードウェアについて述べる。以下、このハードウェアを“視覚装置”と表現する。このハードウェアアーキテクチャは、1.4節で紹介したアーキテクチャと同様に、小規模なPEを多数配置した構成をしている。ここでは、視覚装置の概要や背景について述べた後、本研究の対象であるアルゴリズムの実行に関するハードウェアアーキテクチャについて述べる。

1.5.1 視覚装置の概要

株式会社エッチャンデスの味岡と慶應義塾大学天野研究室は共同で、局所的な画像処理を並列に行うハードウェアの研究を行っている [1],[64]-[65]。

味岡は、情景画像から抜き出された文字や記号情報をもとに視覚障害者の支援を行うことをターゲットとし、代読サービスや歩行支援サービスを想定している。このようなサービスは高速・高精度な性能はあまり求められない反面、長時間の使用に耐えられ、持ち運びが便利であることが要求される。そのため、低コスト、低消費電力であることが必要である。

このような性能を達成するため、味岡らは小規模なPEを多数使用し、局所並列処理を行うことでハードウェアの動作周波数を低く抑えながら、必要な能力を確保することを目指し、“視覚装置”の開発を行っている。

このシステムは、パン・チルト一体型カメラと画像処理用ハードウェアアーキテクチャで構成されている。ハードウェアアーキテクチャは、ハフ変換・フーリエ変換やヒストグラム計算のような大域処理を用いるアプローチではなく、局所処理のみを用いるアプローチに適した構造をしている。

この局所並列画像処理に適したハードウェアアーキテクチャは、格子上に配列された論理的な配列演算ユニット (AOU: Array Of Unit) から構成される。各々のAOUは少なくとも1画素を処理する論理的な単位であり、隣接のAOUに対して非同期にデータのやり取りをすることができる。

LSIに実装する場合には、PE(Processing Element)に複数のAOUを割り付ける。図1.1はPEアレイへの入力画像の割り当てイメージを表している。一つのPEが複数の画素を扱い、各PEは通信コントローラを介したパスで接続され、PEアレイ(配列)を形成している。そして、各PEが独立に動作し、局所並列画像処理を実行することができる。

以降では“視覚装置”のハードウェアアーキテクチャについて述べる。

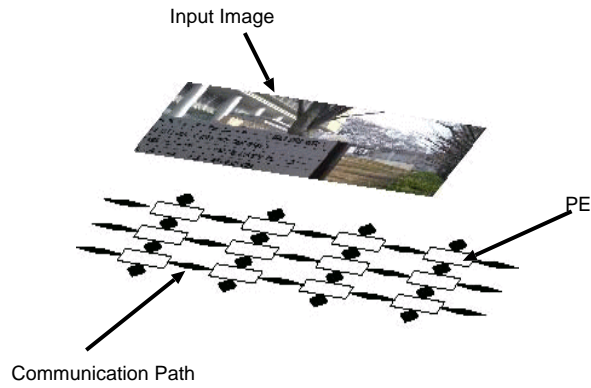


図 1.1 PE アレイの割り当て

Fig. 1.1 An assignment of PE array

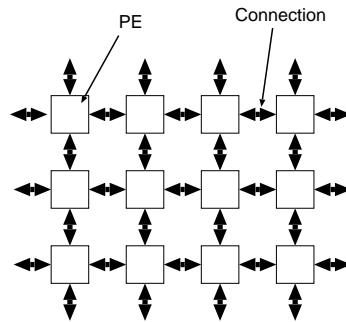


図 1.2 視覚装置ハードウェアの構成

Fig. 1.2 A framework of the Visual Device

1.5.2 視覚装置のハードウェアアーキテクチャ

視覚装置ハードウェアアーキテクチャの概要

視覚装置で用いられる格子状のハードウェアアーキテクチャでは、単純な構造の PE を多数使用し、それぞれの PE に複数の画素を処理させることでハードウェアの動作周波数を低く抑えながら、必要な能力を確保することを目指している。図 1.2 にこのアーキテクチャの概略図を示す。

PE の構成

各 PE の構成は図 1.3 に示されるように、大きく分けてローカルメモリとコアプロセッサ、隣接する 4 方向の PE と通信する通信コントローラからなる。コアプロセッサはアルゴリズムの実行を行い、ローカルメモリはデータの保持に関わっている。通信コントローラは PE 間のデータ転送の制御を行い、各 PE はこの通信コントローラを

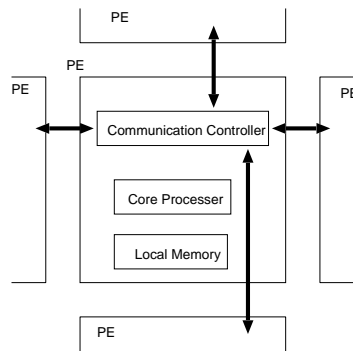


図 1.3 PE の構成

Fig. 1.3 A framework of a PE

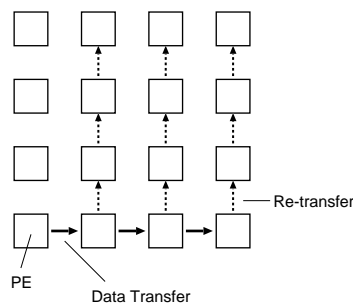


図 1.4 PE 間のデータ通信

Fig. 1.4 Data transfer between PEs

介して上下左右に隣接する PE と直接接続されている。

PE 間のデータ通信

PE 間のデータ通信を複雑な機構を用いずに効率良く行う手法が味岡により提案されている [1]。この手法では、ある PE から転送されたデータは一定方向へ転送を繰り返され、一定回数転送されると破棄される。また、転送中一度だけ転送方向を変更して転送を行う“再転送”という処理が行われる。これにより近隣の PE に渦巻状にデータが行き渡り、局所的な画像処理に必要な画素情報を PE 間で効率良く共有することができる。

データが転送される様子を図 1.4 に示す。図中の矢印はデータの流を表している。図 1.4 では、左下の PE から右方向へデータが転送される様子を表している。左下の PE から転送されたデータは順々に右方向の PE へと転送される。図中の破線矢印は再転送されたデータの流を表している。データ転送の際には“計算ステータス”という 16bit の付加情報を同時に転送する。“計算ステータス”には

- 転送回数を表す情報
- 再転送したかどうかを表すフラグ
- どの時点で再転送が行われたかを表す情報
- PE 内における画素データの位置情報

などが含まれている。この“計算ステータス”から、転送されたデータの送信元の PE や、その PE 内の画素の位置を特定することができる。

視覚装置の入出力

味岡は視覚装置ハードウェアの入出力機構として、図 1.5 のような三次元構造を想定している。最上部に画像取得用のデバイスを配置し、その下に PE アレイを配置し、画像情報を並列に入力、処理を行うとともに処理結果をさらに次の段の PE アレイに並列に出力する。各 PE アレイでは異なるプログラムが実行され、これらがパイプライン的に行われることにより、低動作周波数での実用的な処理速度の実現を想定している。

1.5.3 視覚装置の有効性

味岡が提案する視覚装置ハードウェアは、慶應義塾大学天野研究室で設計・評価が行われている [64][65]。視覚装置ハードウェアの特長は以下の通りである。

1. 局所並列性を引き出し、拡張性を持たせるため、構造を均一化した小規模の PE を多数用いている。

局所並列処理を行うため、従来の画像処理プロセッサと同様に小規模の PE を複数使い、画像を分割して並列に処理することができる。また、均一化された構造の PE を用いることで、処理できる画像サイズを拡大するために PE を追加することができる。

2. 多近傍処理を容易にするため、ローカルメモリを用いている。

8 近傍画素以上の多近傍処理を行う場合、1 つの PE で 1 画素を処理する構成では、配線が複雑になり、PE 間通信が頻繁に行われてしまう。そこで、ローカルメモリを用い、1 つの PE で複数の画素を処理することができる。

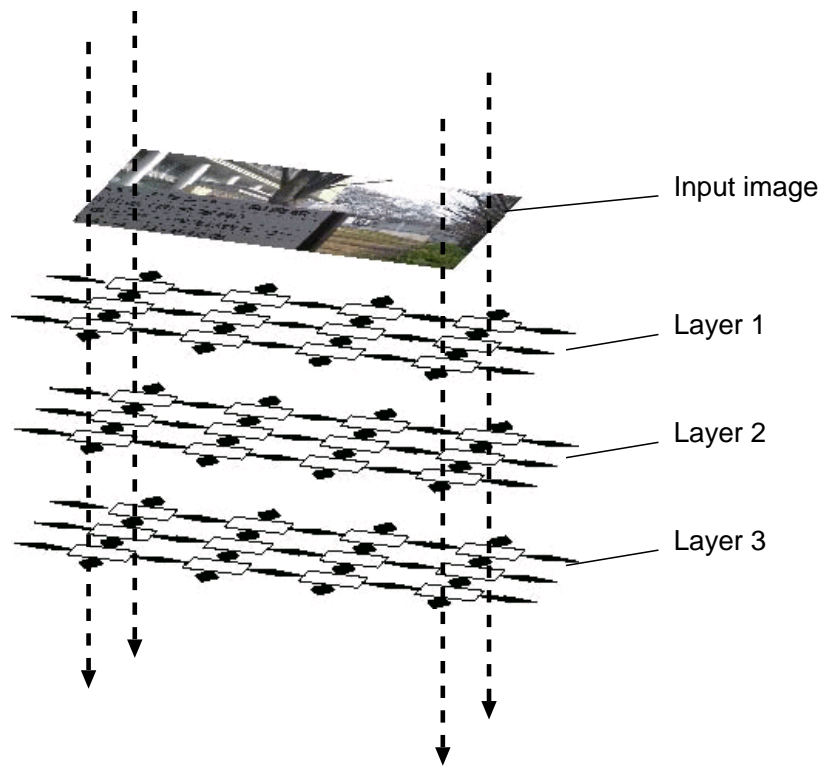


図 1.5 視覚装置の入出力構造

Fig. 1.5 Input and Output of the Visual Device

3. プログラムの自由度を高めるため、SPMD(Single Program, Multiple Data Streams)方式である。

従来多く用いられている SIMD 方式では、全ての PE で同じ命令が同時に実行されるため分岐実行などに制限がある。プログラムの自由度を高めると共に命令パスによる拡張性への制限を緩和するため、各 PE に命令メモリを持たせ、共通のプログラムを個別に実行する SPMD 方式を用いている。

4. PE 間の通信のため、演算機構とデータ共有機構を分離している。

SPMD 方式では、各 PE は個別にプログラムを実行する。このため PE 間でデータ共有を目的とした通信が必要な場合、PE 同士を同期させてから通信を行うのは非効率的である。そこで演算機構とデータ共有機構を分離し、PE 間通信をデータ共有機構に行わせることで、プログラムの実行に影響を及ぼすことなくデータの共有を行う。

慶應義塾大学天野研究室の福田らにより設計された視覚装置ハードウェア [66][67] のアーキテクチャを以下にまとめる。

1. 16bit コアプロセッサ

プログラムの実行と通信コントローラの制御を行うコアプロセッサには16bitRISCプロセッサが用いられている。このプロセッサは4ステージ構成のパイプラインを採用している。SPMD方式では、プログラムの実行は各PEで個別に行われるためそれぞれ実行位置が異なる。そのためPE間で同期を取るためのタイマを持つ。

2. ローカルメモリ

コアプロセッサにはそれぞれ8KBの命令メモリ、データメモリが接続されている。

3. 通信コントローラ

各PEは通信コントローラを持ち、近傍のPEと通信を行うことにより、他のPEの画素情報を参照することができる。

4. ステートマシン

各PEには動作状態を決めるステートマシンが備わっている。ステートマシンはコアプロセッサからの命令や外部からの信号などによって状態遷移し、“NORMAL”(通常状態)、“SLEEP”(タイマと通信コントローラのみ稼働し、プロセッサとメモリへの電源供給が止まる。)、“SYNC”(通信コントローラのみが動作し、PEの同期をとる)、“HALT”(停止状態)の4つの状態をとる。

局所並列処理のために開発された画像処理アルゴリズム [74] が、福田らによりFPGA上に実装され評価が行われ、以下のことが確認された。

- 拡張性

PEの最大動作周波数は40MHz程度であり、PE数を増加させても動作周波数の低下はほとんど生じない構造を有している。

- 消費電力

FPGA上に実装した時の1PE当たりの消費電力の見積りは33MHz動作時で150mW程度であり、“SLEEP”状態では60mW程度に抑えられる。また、ASIC(Application Specified IC)化した場合にはさらなる消費電力の低減が見込まれる。視覚装置は1~2W以下という性能が求められているが、これを十分に達成できる。

- 処理性能

フィルタ等の単純な処理であれば毎秒150フレーム程度の処理が行える。局所並列画像処理アルゴリズムで用いられるデータ共有は最長の場合でも2msec以下に抑えることができる。1PE当たりが扱う画素数は 15×15 画素程度が適当である。

以上のことから、味岡が提案する視覚装置は実用に耐え得ると考えられる。味岡はこの視覚装置を利用した視覚障害者支援サービスを想定している。本研究は視覚障害者支援のために、局所処理のみでの文字抽出・ピクトグラム抽出を行うアルゴリズムに関する研究である。次節では、味岡が提案する視覚装置に適した局所並列画像処理に関する制約について述べる。

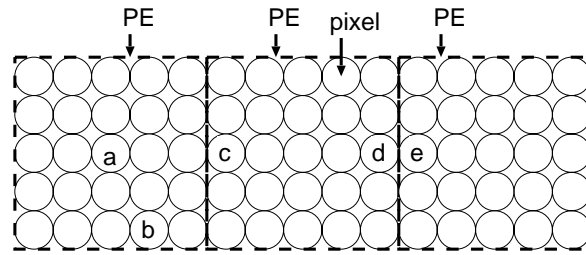


図 1.6 PE 間通信コスト

Fig. 1.6 Cost of communication

1.6 視覚装置のための局所並列画像処理アルゴリズムに関する制約

1.6.1 視覚装置における PE の仕様

本研究では、アルゴリズムを構成する上で必要となる PE の仕様を以下のように仮定した。

PE 当たりの画素数

福田らの研究結果 [66][67] より、1 PE 当たりが処理する画素数は、 15×15 画素とする。

メモリアクセス最小単位

メモリアクセスの最小単位は 4bit である。そのため、二値画像のように狭い範囲の値を扱う際には、1 枚の画像に対して 1byte 消費する必要はない。

隣接 PE との通信コスト

PE 間通信に要するコストは、画素間の距離ではなく、PE 間の距離によるものとする。すなわち、図 1.6 において、(a) の画素を処理する場合に、(b) の画素値を参照しても通信コストはかからない。(a) の画素を処理する際、(c) の画素値を参照しても (d) の画素値を参照しても通信コストは同等である。ただし、(e) の画素情報を参照するためには、転送を 2 回繰り返さなければならないため、(c),(d) の画素情報を参照するよりも通信コストがかかる。

1.6.2 ブロック処理との相違点

視覚装置ハードウェアによる局所並列画像処理は、画像をブロックに分割して処理する、一般的なブロック並列処理とは異なる。

一般的なブロック処理では、画像をブロックに分割した後、各ブロック内の情報のみで一定の処理を行う。そのため、とりわけブロック境界付近の画素の処理では、処理に必要な画素がブロック外にあるために、その情報を利用できず、精度の低下を招く。そのため、類似した画像を処理する場合でも、画像に対するブロックの分割位置のわずかな違いにより、精度が変わってしまうことがある。

それに対し、本研究で対象としている局所並列画像処理は、画像をブロックに分割して処理を行うが、必要な画素情報は、PE の範囲外であっても、PE 間の通信により得ることができる。そのため、画像の分割の仕方が処理精度に影響を及ぼすことはない。ただし、無限に PE 間通信を利用するアルゴリズムでは、1 つの CPU による処理と同様になってしまい、複数の PE が並列処理するという利点を活かすことはできない。

1.6.3 PE の統合処理

視覚装置における PE の立場は全て同等であり、視覚装置ハードウェアは全ての PE の情報を統合して処理する機構はもたない。

例えば、図 1.7 に示すように、領域 A、領域 B、領域 C は、それぞれの PE において領域として認識することは可能であるが、3 つの領域を合わせた 1 つの大きな領域として扱うことは、PE 間通信なしではできない。また、これら 3 つの領域を 1 つの領域として、抽象化して処理することはできない。PE の立場は互いに同等であるため、どれか 1 つの PE が親として動作することはなく、また PE の情報を統合して処理する機構がないため、上記のように抽象化して扱うためには、A,B,C の 3 つの PE が 3 つの領域を重複して処理しなければならない、並列処理の利点が失われてしまうためである。

1.6.4 局所並列画像処理アルゴリズムの目的

局所並列画像処理アルゴリズムの目的は、局所的な処理を並列に行うことである。そのため、1 つの画素を処理するために参照する画素は少なければ少ない程良い。すなわち、 5×5 のフィルタを利用するよりは、 3×3 のフィルタを利用の方が通信コストが低く、並列性が高い。例えば、図 1.8 に示すように、 3×3 のフィルタリングのような 8 近傍処理の場合、PE 間通信が発生するのは、各 PE の端の画素を処理する場合のみである。

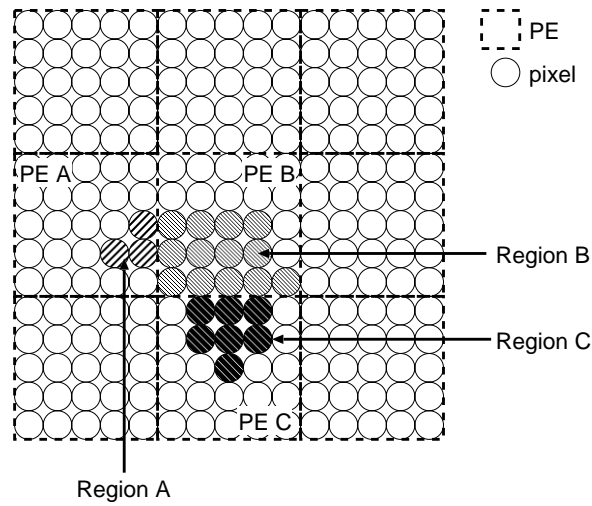


図 1.7 PE における領域の扱い

Fig. 1.7 Treatment of regions

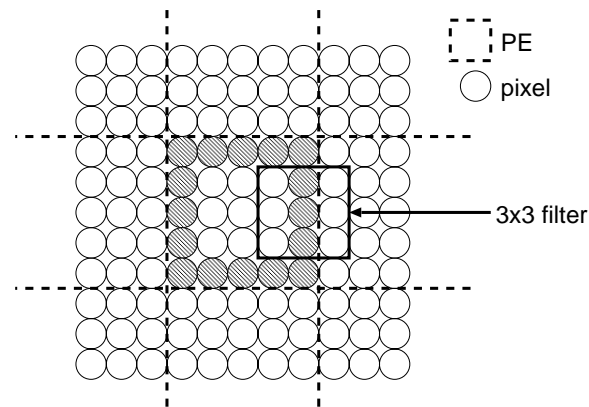


図 1.8 8 近傍処理の参照画素

Fig. 1.8 Reference pixels of eight neighbor processing

しかし、画像処理は、本質的に並列性が高い処理と、並列性が低い処理があるため、全ての処理を 3×3 のフィルタ処理で行うことは不可能であると考えられる。そこで、本研究では、ほとんど全ての処理を 8 近傍の演算で行い、8 近傍演算が不可能な場合のみ、より多くの近傍を参照する演算を行うものとする。

次節では本研究の目的と位置付けについて述べる。

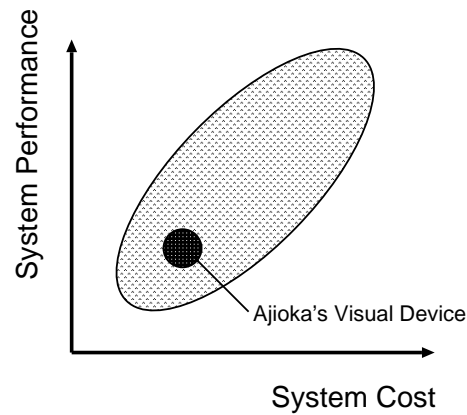


図 1.9 システムのコストとパフォーマンスの関係

Fig. 1.9 A relationship between cost and performance

1.7 本研究の目的と位置付け

本研究は、局所並列処理に適した、文字やピクトグラムを画像から抽出するアルゴリズムに関する研究である。

図 1.9 にシステムのコストと性能の比較を示す。図のように、一般的にコスト (ハードウェアコスト、消費電力等) をかけることにより、ある程度システムの性能を向上させることが可能である。味岡が提案する“視覚装置”は、なるべくコストをかけずに実用に耐え得る性能を出そうとするものである。

アルゴリズムに関しても、図 1.9 と同様の関係が成立すると考えられる。計算コストをかけ、厳密な計算を行い、高精度な結果を得る画像処理アルゴリズムが多数研究されている。本研究は、局所並列処理を行うハードウェアへの実装のためのアルゴリズムの研究である。そのため、計算コストがかかる複雑な計算や大域的な処理を行わないという制約を設けた上でのアルゴリズムの研究となっている。

本研究と、これまでの画像処理アルゴリズム、画像処理プロセッサ関連の研究との関係を図 1.10 に示した。1.4 節で述べたように、画像処理プロセッサは盛んに研究されているが、ハードウェアがあっても、その上で動作するアルゴリズムがなければならぬ。従来のプロセッサには、それぞれのアーキテクチャに合ったアルゴリズムで動作するように、本研究は、味岡が提案する視覚装置のような、局所並列処理を主に行うハードウェア上に実装されるためのアルゴリズムに関する研究である。

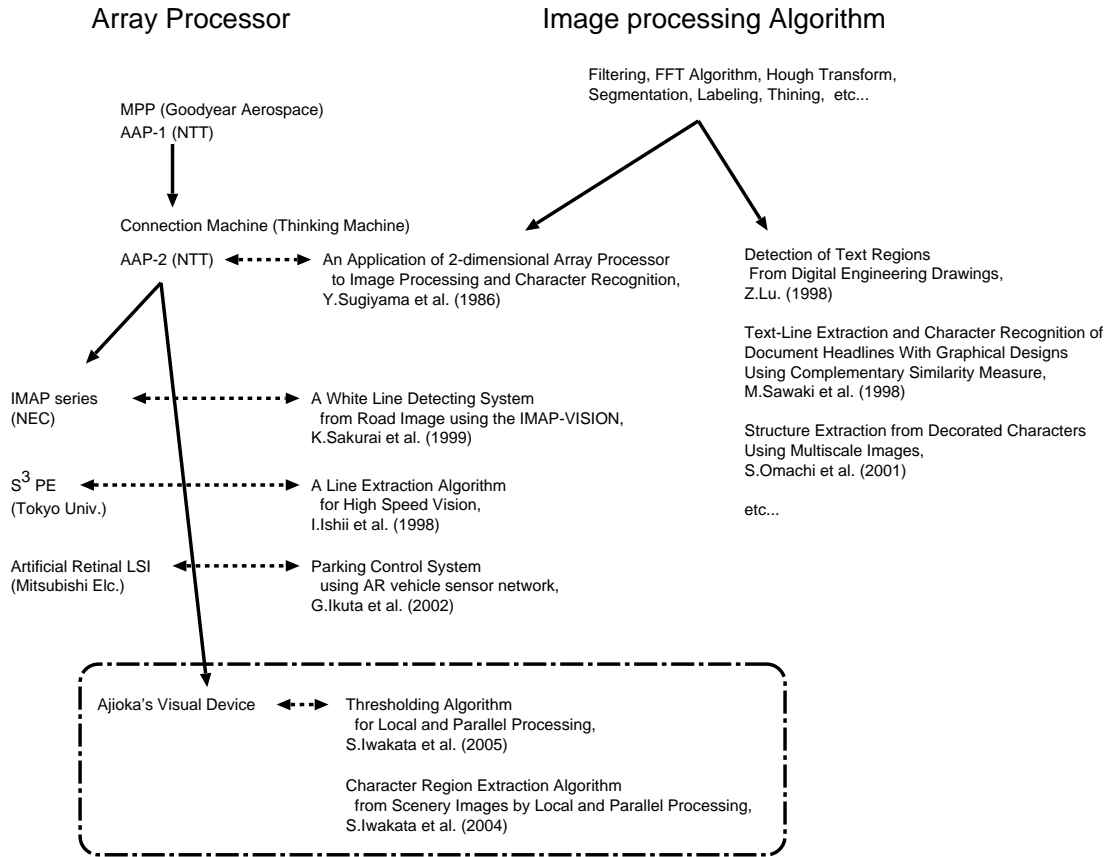


図 1.10 本研究の位置付け

Fig. 1.10 Positioning of the study

1.8 本論文の構成

本論文は全 4 章から構成される。

第 2 章では、情景画像から文字を抽出するアルゴリズムの研究について述べる。

第 3 章では、第 2 章と同様に、局所処理のみで構成されたピクトグラム検出アルゴリズムについて述べる。

第 4 章では、本論文のまとめを行う。

第 2 章

局所並列処理のための文字抽出アルゴリズム

本章では局所的な処理のみによって情景画像から文字領域を抽出するアルゴリズムを提案する。画像から文字を抽出するアルゴリズムはこれまでも多く提案されているが、そのほとんどは計算量が多く、大域的な計算を用いているため、第 1 章で述べたような、局所的な処理を並列に行うアプローチには適していない。

それに対し、本章で提案するアルゴリズムは局所的な情報のみを用いた処理により構成されており、画像を空間的に並列に処理することが可能である。

提案アルゴリズムは、輝度値変換過程・ベース画像生成過程・ノイズ抽出過程・文字候補抽出過程・文字抽出過程からなる。入力カラー画像、出力は二値画像である。輝度値変換過程では、RGB 値を輝度画像へ変換する。ベース画像生成過程では、輝度画像に対して平滑化を行った後に二値化する。ノイズ抽出過程では、輝度画像に対して文字消去を行った後に二値化する。文字候補抽出過程では、これら 2 通りの二値化結果の差分から文字領域をおおまかに抽出する。文字抽出過程では、画素値を各領域の重心の方向へ移動し合算することにより、領域の大きさを求める。その後、文字として適当な大きさの領域のみを出力する。

ベース画像生成過程とノイズ抽出過程における二値化はラプラシアンによるエッジ情報を基に行われ、輝度しきい値を外部から設定する必要はない。

以上の処理を全て近傍処理によって実現する。また、全ての PE において同一のプログラムを実行する。

計算機シミュレーションでは、従来法よりも高精度で情景画像から文字列を抽出できることを確認し、提案アルゴリズムの特性についても評価を行った。

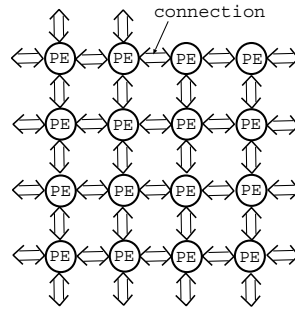


図 2.1 視覚装置ハードウェアの概形

Fig. 2.1 The visual device for local and parallel processing.

2.1 はじめに

近年、LSI 技術の発達により携帯端末の高性能化・小型化が進んでいる。それによりカメラ付き携帯電話が普及し、携帯端末による画像処理が盛んに行われている [77][78]。携帯端末の利便性を活かし、視覚障害者を支援する研究も行われている [79]。しかしながら、デスクトップ PC のような高性能なマシンと異なり、携帯端末で画像処理を行おうとすると現実的な時間で処理を終わらせることは困難である。高速に画像処理を行うためシステムをハードウェア化する研究もあり、ニューラルネットワークをハードウェア化したチップも提案されている [80]-[84]。また、ニューラルネットワークを利用した画像処理に関する研究は多く提案されている [85][86]。しかしながら、ニューラルネットワークは結合数が多いためハードウェア化するには適していないと考えられる。結線数の多さから、配線が複雑になったり、チップサイズが大きくなってしまいうため、小型化のコストもかかってしまうからである。

上記のような問題に対しては、第 1 章で述べたように小規模な PE を多数配置したアーキテクチャにより並列に画像を処理するアプローチが有効であると考えられる。このようなアプローチをとることが可能なハードウェアアーキテクチャの概形を図 2.1 に示す。

視覚装置は複数のプロセッシング・エレメント (Processing Element : PE) が格子状に並んだ構成をしている。それぞれの PE が、分割された画像の各部分を並列に処理することにより、高速な画像処理が実現できる。各 PE は 4 近傍の PE とのみ物理的な接続を通して非同期的に通信を行うが、順番にデータを転送することで論理的にはより大きな近傍と通信を行うことも可能になっている。また、PE はお互いに同期して連続して局所処理をすることができる。そのような処理を本論文では局所並列画像処理と呼んでいる。各 PE は一つにつき、最大約 15×15 画素を扱うことができる程度の記憶容量を持つ。そして、最大 16bit までの多ビットシフト、乗算などを行うことがで

き、その記憶容量は約 8KB である。

この視覚装置の特長は低電力で並列に動作することである。視覚装置の各 PE の動作周波数は約 40MHz である。一般的に CPU の消費電力 P は以下の式で近似されることが知られている。

$$P = \alpha \cdot f \cdot V_{dd}^2 + \beta \cdot V_{dd} \quad (2.1)$$

ここで、 f は動作周波数、 V_{dd} は電圧を表す。また、 α, β は定数である。この式から、動作周波数を低くすることができれば、消費電力を抑えられることが分かる。また、動作周波数を低くすれば、電圧を低くしても動作させることができるため、 V_{dd} を低くすることも可能となり、より一層消費電力を低くすることができる。

消費電力を低く抑えられるため、携帯用のバッテリーが少なくて済む。このため、システムを大幅に小型化することが可能である。また、並列動作が可能なことにより、局所的な並列計算を高速に行うことができる。したがって、局所並列画像処理を行うシステムの大幅な小型化が実現可能であり、携帯電話への組み込みなどの用途でも大きな需要が見込まれる。しかし、この視覚装置の利点を活かすには、局所並列画像処理からなるアルゴリズムの開発が必要である。局所並列画像処理アルゴリズムを実装することにより、画像サイズの増加に対する計算時間の増加を抑えることが可能である。

一方、画像処理により視覚障害者をサポートする研究が盛んに行われている [23]。文献 [23] は視覚障害者の支援のために情景画像から文字列を抽出するシステムを提案している。

文字領域抽出の代表的な手法は大きく、

1. 二値化処理を基本とした手法 [24]
2. 文字のエッジ情報を利用した手法 [29]
3. 文字線の形状特徴を利用した手法 [44][87]
4. 多値閾値処理またはクラスタリング手法 [13][88]

の 4 手法に分類される [13]。いずれの手法においても良好な結果が得られた研究がある。しかしながら、従来の研究の多くは画像をソフトウェア的に、また逐次的に処理することでこれらの手法を実現している。したがって、精度は良いが計算コストが必要となる [64]。そこで小規模な PE を多数配置したようなハードウェアで並列に処理することが望ましい。

しかし、従来のようにフーリエ変換やハフ変換を利用するアルゴリズム [24][44][13],[43]-[45] や、画像の濃度値のヒストグラムを利用するアルゴリズム [41]、クラスタリングを用いるアルゴリズム [29][26] などは小規模な PE による空間的な並列処理には適し

ていない。なぜなら、ハフ変換のように大域的な演算を行うと並列性が失われてしまい、通信量が増えてしまうからである。

局所的な処理のみで構成されたアルゴリズムであれば、小規模な PE による並列演算により、低コストで高速な処理が可能となる。その結果、システムの普及につながると考えられる。

そこで本章では、注目画素の近傍の局所的な情報のみを用いた処理によって情景画像から文字を抽出するアルゴリズムを提案し、実際の情景画像を用い計算機シミュレーションを行う。ここで対象とする文字は主に情景画像に含まれる看板中の文字である。

以下、2.2 節では文字抽出アルゴリズムについて説明する。2.3 節では計算機によるシミュレーションを行い、その有効性を確認する。2.4 節で考察を述べ、2.5 節では本章のまとめを述べる。

2.2 文字抽出アルゴリズム

2.2.1 概要

図 2.2 に本章で提案する文字抽出アルゴリズムの流れを示す。提案アルゴリズムの入力はカラー画像、出力は二値画像である。提案アルゴリズムは輝度値変換過程 (Transform process) ・ ベース画像生成過程 (Base image creation process) ・ ノイズ抽出過程 (Noise extraction process) ・ 文字候補抽出過程 (Character candidate detection process) ・ 文字抽出過程 (Character extraction process) ・ ノイズ除去過程 (Noise elimination process) からなる。

まず、カラー画像の RGB 値を輝度値へ変換する。輝度値へ変換された画像に対し 2 通りの二値化が行われる。一方はベース画像生成過程であり、平滑化後に二値化を行う。これは主に文字を抽出するための二値化である。もう一方はノイズ抽出過程であり、文字領域を消去した後に二値化を行う。これは文字を消去するための二値化である。文字候補抽出過程では、これら 2 通りの二値化結果の差分をとり、文字候補とする。文字抽出過程では、各領域の中心の方向へ画素値を移動させ領域の大きさを局所的に求めるアルゴリズムを利用する。そして求めた領域の大きさに基づき、文字かどうかの判断を行う。ノイズ除去過程では余分な領域を除去し、文字のみの二値画像が出力される。

2.2.2 近傍定義

本章では、すべての処理を近傍計算によって行う。画像 X 中の位置 (i, j) の画素値を $x(i, j)$ と表現する。 (i, j) の q 近傍集合 $P_{i,j}^q$ は次式のように表される。

$$P_{i,j}^q = \{(l, m) \mid i - r \leq l \leq i + r, \\ j - r \leq m \leq j + r, (l, m) \neq (i, j)\} \quad (2.2)$$

式 (2.2) において r は近傍長である。近傍数 q は r を用いて、以下のように表される。

$$q = (2r + 1)^2 - 1 \quad (2.3)$$

すなわち、 $r = 1, 2, 3, \dots$ のとき、 $q = 8, 24, 48 \dots$ である。

なお、近傍集合 $P_{i,j}$ のうち画像サイズをはみ出した位置 (l, m) の画素値 $x(l, m)$ には画像の端の画素値が代用される。これにより、全ての画素において近傍数は等しくなる。画像サイズをはみ出した位置 (l, m) の代用画素値 $x(l, m)$ の位置 l, m は、画像の縦横の座標を表すインデックス l, m のとる値をそれぞれ $0 \sim l_{max}$, $0 \sim m_{max}$ とすると以下の式で表される。

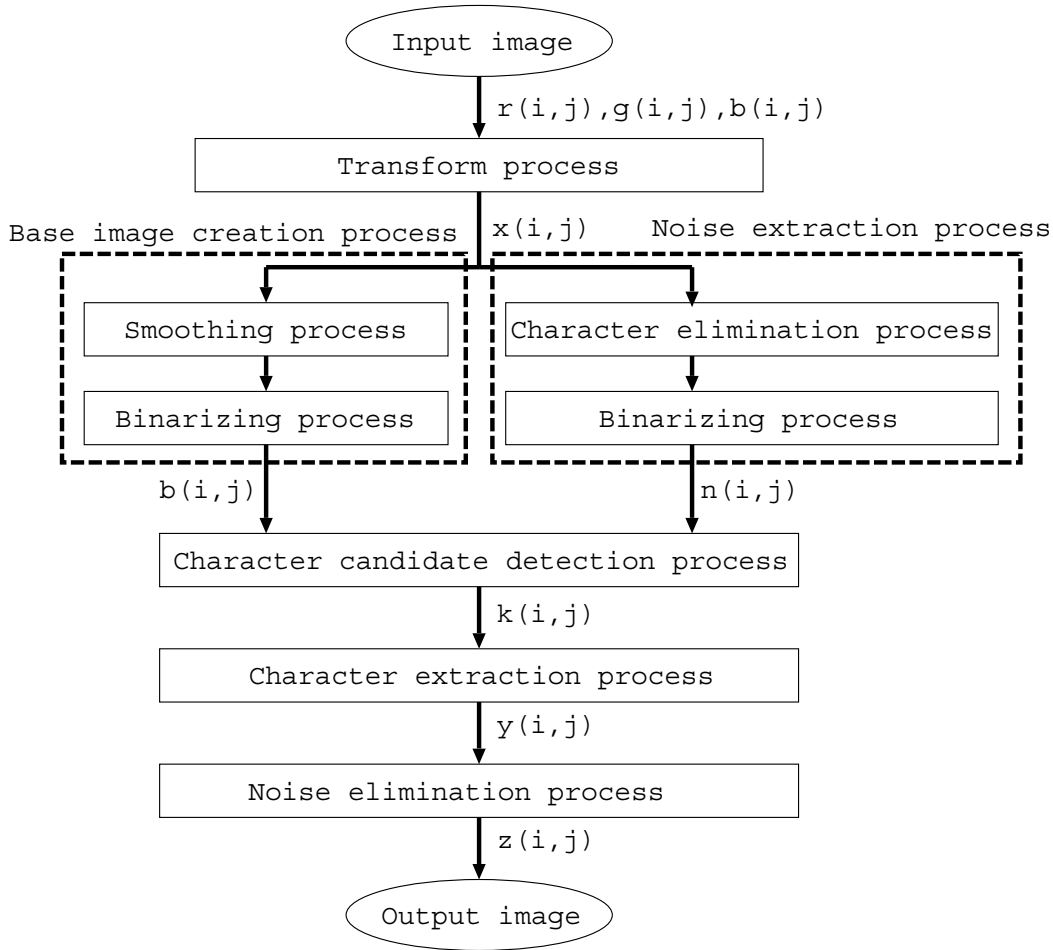


図 2.2 文字抽出アルゴリズムの流れ

Fig. 2.2 Flow of the character extraction algorithm.

$$\begin{aligned}
 l &= 0 & \text{if } l < 0 \\
 l &= l_{max} & \text{if } l > l_{max} \\
 m &= 0 & \text{if } m < 0 \\
 m &= m_{max} & \text{if } m > m_{max}
 \end{aligned} \tag{2.4}$$

2.2.3 輝度値変換

提案アルゴリズムにおいて、入力画像として想定されるのはRGBそれぞれ256階調のカラー画像である。画像処理の分野では、RGB色空間に比べ人間の色知覚に近いとされる $L^*a^*b^*$ 色空間に変換する手法[26][41]や、明度、彩度を利用するためHSL変換を行う手法[13]が多い。しかしこれらの変換は複雑な計算を必要とする。これはハードウェア的に大きな負担となるので提案アルゴリズムには適さない。したがって、本

章では輝度値を用いる。入力画像の RGB 値は以下の式に従って輝度値に変換される。

$$x(i, j) = 0.3 \cdot r(i, j) + 0.6 \cdot g(i, j) + 0.1 \cdot b(i, j) \quad (2.5)$$

輝度値の変換処理は、各画素内で閉じているため、PE 間通信を行う必要はない。

2.2.4 ベース画像生成過程

ベース画像生成過程は、平滑化と二値化からなる。

平滑化

入力画像 X の画素値 $x(i, j)$ は次式に従って平滑化され、 $x^{\text{new}}(i, j)$ となる。

$$x^{\text{new}}(i, j) = \frac{1}{8} \sum_{(l, m) \in \hat{P}_{i, j}^8} x(l, m) \quad (2.6)$$

式 (2.6) において、

$$\hat{P}_{i, j}^8 = \begin{cases} (l, m) & \text{if } |x(i, j) - x(l, m)| \leq \epsilon \\ (i, j) & \text{otherwise} \end{cases} \quad (2.7)$$

である。注目する画素との画素値の差が平滑化パラメータ ϵ を超える近傍画素値には、注目画素値が代入される。これによりエッジ部分のように急に画素値が変化する部分では平滑化が行われなくなる。

画像中においては一般に抽出対象領域とその背景領域のコントラストが高くなっていることが多い。平滑化により必要なエッジを際立たせ、同一領域内に存在する濃度のばらつきを抑えることができる。提案アルゴリズムではエッジ抽出にはラプラシアンフィルタを用いているが、このような空間フィルタは直接用いるとノイズに弱い。この平滑化を行わないと、次のエッジ抽出で不要なエッジが多く検出されてしまう。

例えば人間が見ると濃度変化を感じない部分でも、実際には濃度変化があることがある。コンピュータはそのような微妙な濃度変化を捉えてしまうので、一つの領域として捉えるべき部分を複数の別の領域として捉えてしまうことが多い。上記の平滑化により、濃度変化が緩やかな部分を徐々に一つの領域として同化させることができる。提案アルゴリズムで行われる平滑化は フィルタ [89] に似ており、 フィルタについてはその有効性が確認されている。 フィルタは、注目画素値との差がパラメータを超える場合には、その画素値は無視するといった処理を行う。しかし提案アルゴリズムでは注目画素値を代用することにより、より緩やかに平滑化することができると考

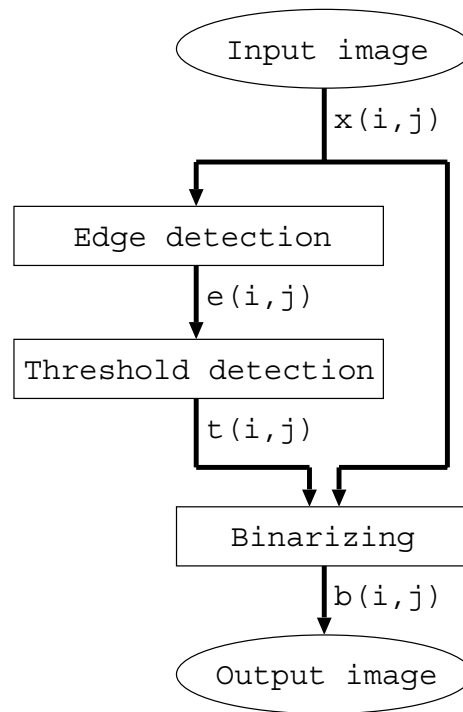


図 2.3 二値化の手順

Fig. 2.3 Flow of the binarizing process.

えられる。このような平滑化を N 回繰り返し行う。平滑化の繰り返し回数は、予備実験によって決定する。

平滑化は、8近傍のフィルタ処理であるため、各 PE の端に位置する画素を処理するためには、PE 間通信を行う必要がある。

二値化

二値化の手順 提案アルゴリズムでは、エッジ情報に基づいて局所処理のみで二値化を行う。二値化の手順を図 2.3 に示す。

提案アルゴリズムにおける二値化は、エッジ抽出・しきい値算出・二値化からなる。エッジ抽出にはラプラシアンフィルタを利用する。得られたエッジ情報に基づき、しきい値を決定し二値化する。

エッジ抽出 平滑化後の画像に対し、エッジ抽出を行う。エッジ抽出にはラプラシアンフィルタを用いる。図 2.4 にラプラシアンフィルタの 3×3 オペレータを示す。

ラプラシアンフィルタを利用し、濃淡画像 X からエッジ画像 E を生成する。エッジ画像 E の各画素 $e(i, j)$ は、次式のように表すことができる。

-1	-1	-1
-1	8	-1
-1	-1	-1

図 2.4 3×3 ラプラシアンフィルタ

Fig. 2.4 Laplacian filter.

$$e(i, j) = \begin{cases} 1 & \text{if } \left[8 \cdot x(i, j) - \sum_{(l, m) \in P_{i, j}^8} x(l, m) \right] > x(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

式 (2.8) では、ラプラシアンフィルタを利用し、画素値の変化が大きい部分を求める。その結果を注目する画素と比較し、画素値の高い領域、すなわち明るい領域の端をエッジとして抽出する。エッジ検出は、8 近傍のフィルタ処理であるため、各 PE の端に位置する画素を処理するためには、PE 間通信を行う必要がある。

しきい値候補の決定 エッジ抽出を行った後、エッジ画素において、近傍領域のしきい値の候補となる値 $c(i, j)$ を求める。しきい値候補 C の各画素 $c(i, j)$ は次式に従って算出される。

$$c(i, j) = \begin{cases} \frac{x(i, j) + \min_{(l, m) \in P_{i, j}^8} x(l, m)}{2} & \text{if } e(i, j) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.9)$$

式 (2.9) において $\min[\cdot]$ は $[\]$ 内の最小値を表している。

エッジとして抽出された画素の近傍において、最も画素値の勾配が急な部分を求め、その中間値をしきい値の候補とする。そのため、式 (2.9) では、エッジ画素の 8 近傍の最小値を求め、その値とエッジ画素との中間値を算出している。エッジ画素以外ではしきい値候補を求める必要がないため計算は行わない。

しきい値候補の算出には、各エッジ画素の 8 近傍の画素値を参照するため、エッジ画素が PE の端に位置する場合のみ、PE 間通信が必要となる。

二値化の実行 しきい値候補をもとに、各画素ごとのしきい値 $t(i, j)$ を式 (2.10) に従い決定する。ここで、しきい値候補 C の各画素 $c(i, j)$ の q 近傍 $(l, m) \in P_{i, j}^q$ のうち、 $c(l, m) > 0$ であるものの集合を $\bar{C}_{i, j}^q$ と置く。

$$t(i, j) = \begin{cases} \text{Ave}[\bar{C}_{i,j}^q] & \text{if } \sum_{(l,m) \in P_{i,j}^q} c(l, m) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.10)$$

式 (2.10) において、 $\text{Ave}[\cdot]$ は平均を表している。

その後、各画素ごとに得られたしきい値を用い、次式に従って画像の二値化を行う。二値化された画像 B の各画素の値を $b(i, j)$ とすると、

$$b(i, j) = \begin{cases} 1 & \text{if } x(i, j) < t(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (2.11)$$

式 (2.10) は、各画素においてもし近傍にエッジ画素が存在すれば、それらのしきい値候補の平均値をしきい値とし、そうでなければ、しきい値を 0 にするというものである。しきい値が 0 であれば、式 (2.11) から分かるように画素値は 0 となる。しきい値の決定では、注目画素の q 近傍の画素を参照しなければならないため、その対象画素が PE の範囲を越えた場合には、PE 間で通信を行う必要がある。

式 (2.10) において、近傍数 q を適切な値に設定することにより、任意の大きさの領域を抽出することが可能である。

以上が提案アルゴリズムの二値化手順である。ベース画像生成過程の入出力例を図 2.5 に示す。



図 2.5 二値化アルゴリズム入出力画像

Fig. 2.5 Input and output images of the thresholding algorithm.

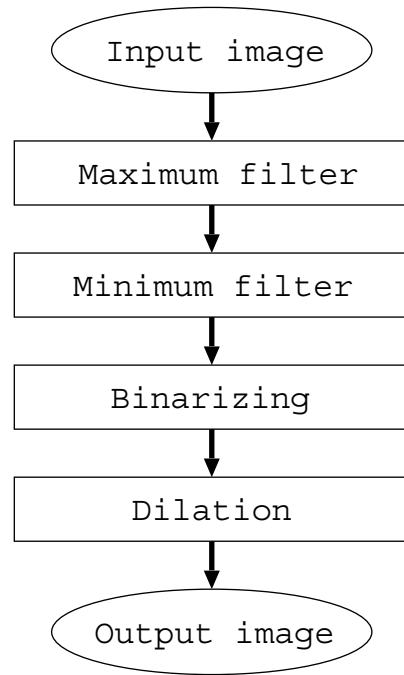


図 2.6 ノイズ抽出過程の流れ

Fig. 2.6 Flow of the noise extraction.

2.2.5 ノイズ抽出過程

次にノイズ抽出過程について述べる。ノイズ抽出過程は文字消去と二値化からなる。

文字消去

文字消去の手順 次に、文字消去過程について述べる。文字消去過程の流れを図 2.6 に示す。

入力画像に対し最大値フィルタをかけた後、最小値フィルタをかけることにより太さが一定以下の領域を消去する。その後二値化し、膨張処理を行う。

入力画像に対し最大値フィルタをかけた後、最小値フィルタをかけることにより太さが一定以下の領域を消去する。その後二値化し、膨張処理を行う。

最大値フィルタ 最大値フィルタとは、各画素において注目画素の 8 近傍の最大値を注目画素の画素値として置き換えるフィルタである。入力画像 X の画素値 $x(i, j)$ は次式に従って更新され、 $x^{\text{new}}(i, j)$ となる。

$$x^{\text{new}}(i, j) = \max[x(l, m) \mid (l, m) \in P_{i,j}^8] \quad (2.12)$$

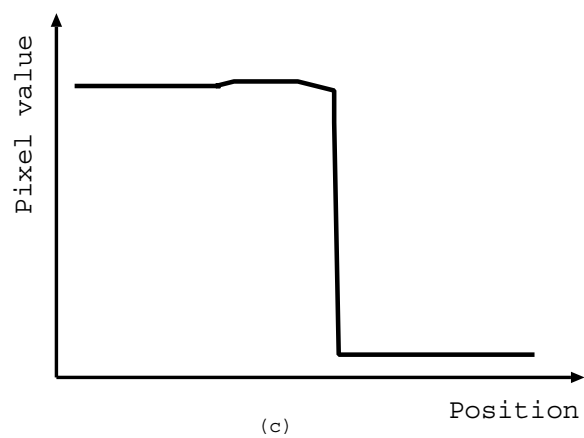
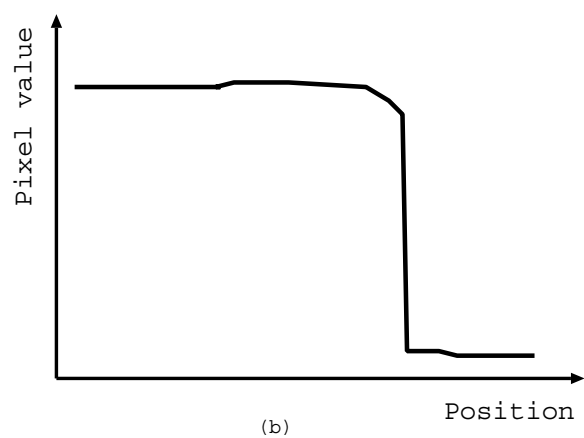
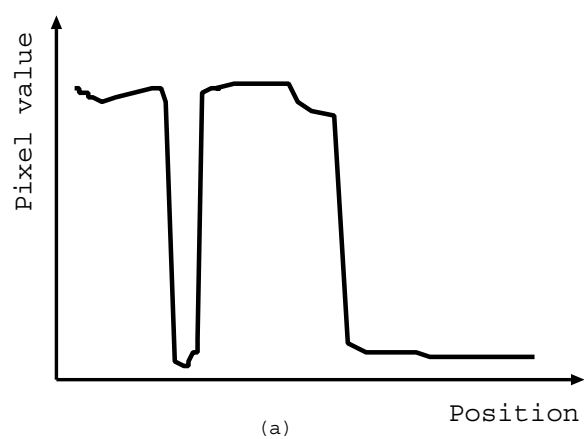
最小値フィルタ 最小値フィルタとは、最大値フィルタとは逆に各画素において注目画素の8近傍の最小値を注目画素の画素値として置き換えるフィルタである。画像 X の画素値 $x(i, j)$ は次式に従って更新され、 $x^{\text{new}}(i, j)$ となる。

$$x^{\text{new}}(i, j) = \min[x(l, m) \mid (l, m) \in P_{i,j}^8] \quad (2.13)$$

図 2.7 に最大値フィルタと最小値フィルタを用いた文字消去の様子を示す。

図 2.7 の (a) における細いくぼみの部分を文字領域と捉え、最大値フィルタを複数回かけることにより、その領域を埋めることができる。この状態が図 2.7 の (b) である。そして、最大値フィルタをかけることにより広がりすぎてしまった部分を、最小値フィルタをかけることにより図 2.7 の (c) のように戻すことができる。最大値フィルタ・最小値フィルタを適用する回数は、いずれもパラメータとして設定し、適切な回数を適用する。

最大値フィルタ・最小値フィルタは、いずれも8近傍のフィルタ処理であるため、各 PE の端の画素を処理する際には、PE 間通信が必要となる。



(a) : Original

(b) : After maximum filter

(c) : After minimum filter

図 2.7 文字消去の様子

Fig. 2.7 An example of the character elimination.

二値化 二値化は2.2.4節と同様のアルゴリズムで行う。文字消去後に二値化した画像を N 、その画素値を $n(i, j)$ と置く。文字消去後に二値化アルゴリズムを適用した結果例を以下に示す。



図 2.8 二値化アルゴリズム (ノイズ抽出過程) の入出力画像

Fig. 2.8 Input and output images of the thresholding algorithm (noise extraction process).

膨張

二値化後、以下の式に従って2～3回の膨張処理を行う。これは以降の過程でより多くの不必要領域を削除するために行う。

$$n^{\text{new}}(i, j) = \begin{cases} 1 & \text{if } \sum_{(l, m) \in P_{i, j}^8} n(l, m) > 0 \\ n(i, j) & \text{otherwise} \end{cases} \quad (2.14)$$

各 PE の端の画素を処理する場合には、PE 間通信を行う。

2.2.6 文字候補抽出過程

文字候補抽出過程では、平滑化後の二値化結果と文字消去後の二値化結果の差分をとり、文字候補領域 K 、その画素値を $k(i, j)$ とする。平滑化後の二値画像を B 、文字消去後の二値画像を N と置くと、 t ステップ目の画素値 $k^t(i, j)$ は以下の式で表される。

$$k^0(i, j) = b(i, j) \quad (2.15)$$

$$k^1(i, j) = \begin{cases} 1 & \text{if } b(i, j) = 1 \text{ and } n(i, j) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.16)$$

$$k^{t+2}((l, m) \mid (l, m) \in P_{i, j}^8) = \begin{cases} 0 & \text{if } k^{t+1}(i, j) = 0 \text{ and } k^t(i, j) = 1 \\ k(l, m) & \text{otherwise} \end{cases} \quad (2.17)$$

まず、式 (2.15) において、 B を K とすることで、初期値 $k^0(i, j)$ を計算する。次に、式 (2.16) で、 B と N の差分を取り、 $k^1(i, j)$ とする。そして、式 (2.17) を繰り返すことにより、変化のあった画素の8近傍の画素値を0にしていく。この処理の繰り返し数はパラメータとして定数で与える。これにより、 B と N の差分を取るだけでなく、消去された領域と連結している領域も消去することができる。

式 (2.17) の繰り返し処理では、8近傍の画素値を参照する。したがって、各 PE の端に位置する画素を処理する場合には、PE 間通信が行われる。

文字候補抽出過程の途中結果画像例を以下に示す。

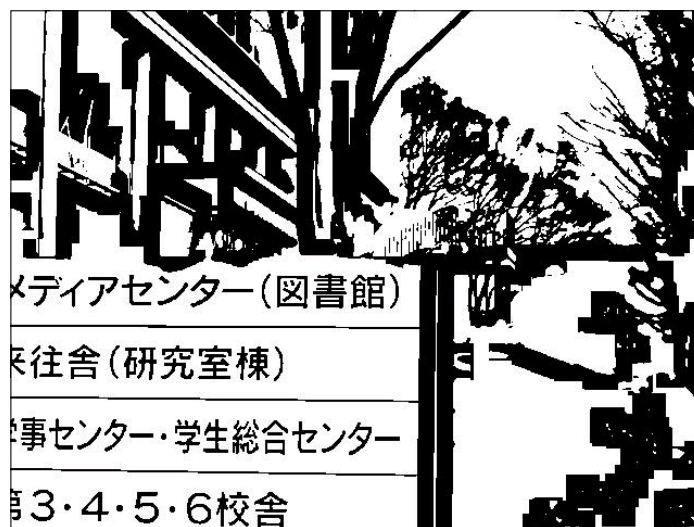


図 2.9 平滑化後の二値化結果例

Fig. 2.9 An example of the output image of the thresholding after smoothing.



図 2.10 文字消去後の二値化結果例

Fig. 2.10 An example of the output image of the thresholding after character elimination.

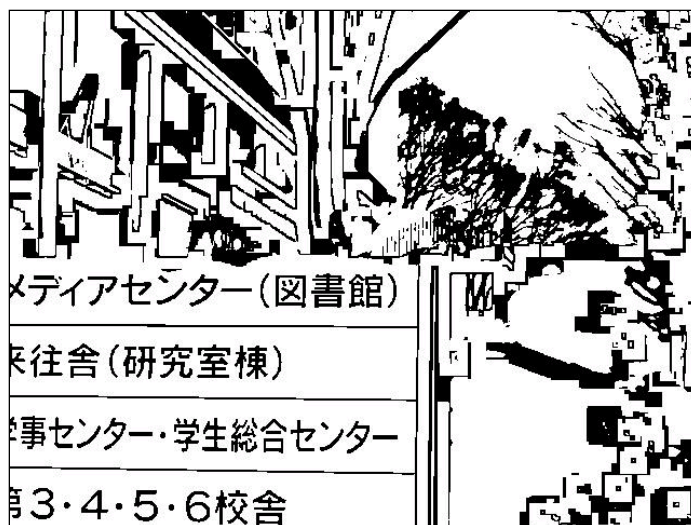


図 2.11 二値画像の差分結果例

Fig. 2.11 An example of the difference image of the binary images.

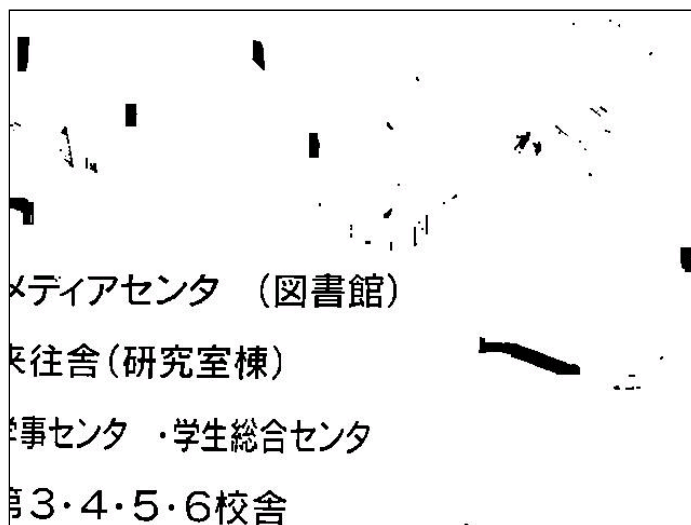


図 2.12 連結領域消去結果例

Fig. 2.12 An example of the character candidate image.

図 2.9 は平滑化をした後に二値化した結果であり、図 2.10 は文字を消去した後に二値化した結果である。図 2.11 は図 2.9 から図 2.10 を引いた結果であり、図 2.12 はさらにその連結する領域を消去した結果の例である。

2.2.7 文字抽出過程

文字候補抽出過程において大まかに文字領域の候補を抽出した後、文字抽出過程で文字が抽出される。文字抽出過程は、各領域の重心方向への画素値の移動と領域サイズの判定からなる [74]。

重心方向への移動

不必要な領域を除去するために、まず各領域の大きさが求められる。これには味岡により提案された、位置/大きさ検出アルゴリズムが利用される [1][64][74]。位置/大きさ検出アルゴリズムは、近傍画素の情報のみをもとに、局所的に各領域のおおまかな重心方向を求め、領域の大きさを検出するアルゴリズムである。以下に手順を示す。

Step1 画像 K を画像 W にコピーする。

Step2 各画素 $w(i, j)$ の 8 近傍の情報を利用し、局所的な重心方向を求める。

Step3 重心の方向の近傍画素へ二値画像の画素値 $w(i, j)$ を移動し合算する。

Step4 画素値の重心への移動が完了すれば終了し、そうでなければ Step2 へ戻る。

画像 W での、位置 (i, j) における重心方向への移動ベクトル $v(i, j)$ は式 (2.18) から式 (2.20) に従い計算される。

$$v(i, j) = (r, s) \quad (2.18)$$

$$r = \sum_{w(l, m) \in P_{i, j}} (l - i) \cdot \hat{w}(l, m) \quad (2.19)$$

$$s = \sum_{w(l, m) \in P_{i, j}} (m - j) \cdot \hat{w}(l, m) \quad (2.20)$$

$$\hat{w}(i, j) = \begin{cases} 1 & \text{if } w(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.21)$$

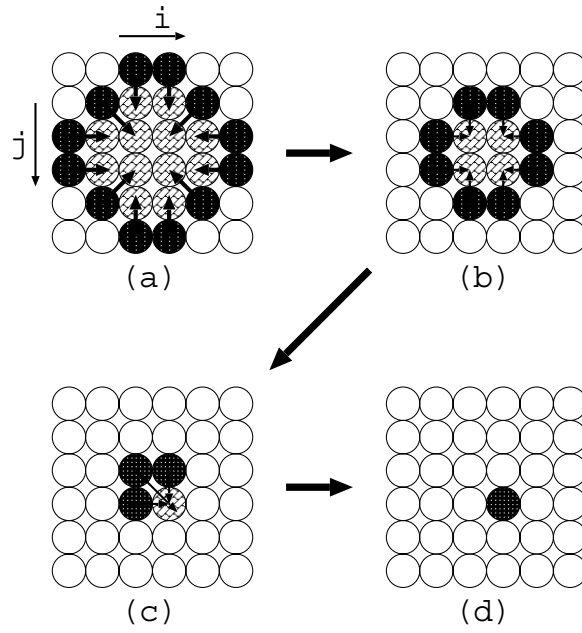


図 2.13 画素値の移動の様子

Fig. 2.13 An example of the pixel movement.

次に8近傍内への移動にするため、この移動ベクトルを式 (2.22) に従い正規化する。具体的には、 r, s の値をそれぞれ 1 または 0 にする。そして $\Theta(v(i, j))$ 方向へ画素の値を移動し、合算する。

$$\Theta(v(i, j)) = \left\{ \begin{array}{ll} (1, 0) & \text{if } r > 0, \\ & 2|s| < |r| \\ (1, 1) & \text{if } r > 0, s > 0 \\ & |r|/2 \leq |s| \leq 2|r| \\ (1, -1) & \text{if } r > 0, s < 0 \\ & |r|/2 \leq |s| \leq 2|r| \\ (-1, 0) & \text{if } r < 0, \\ & 2|s| < |r| \\ (-1, 1) & \text{if } r < 0, s > 0, \\ & |r|/2 \leq |s| \leq 2|r| \\ (-1, -1) & \text{if } r < 0, s < 0, \\ & |r|/2 \leq |s| \leq 2|r| \\ (0, 1) & \text{if } s > 0, \\ & 2|r| < |s| \\ (0, -1) & \text{if } s < 0, \\ & 2|r| \leq |s| \\ (0, 0) & \text{otherwise} \end{array} \right. \quad (2.22)$$

この移動を繰り返し、各領域の画素をその重心画素へ集める。画素値が移動する様子を図 2.13 に示す。

ここで、式 (2.22) だけでは、図 2.13 で (c) から (d) への移動には対応できないという問題点がある。例えば図 2.14(a) のような状態になった場合、式 (2.22) に従うと (b) の状態にはならず、2 つの画素値が互いに移動し (a) のままになってしまう。そこで画素値移動の最後に一回だけ、式 (2.23) を適用する。これにより上記の問題は解決され、

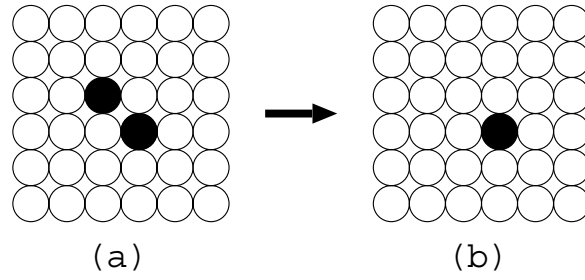


図 2.14 画素値の移動の停滞

Fig. 2.14 The stasis of the pixel movement.

図 2.14 において状態は (a) から (b) へと遷移する。

$$\Theta'(v(i, j)) = \left\{ \begin{array}{ll} (1, 0) & \text{if } r > 0, \\ & 2|s| < |r|, \\ (1, 0) & \text{if } r > 0, s < 0 \\ & |r|/2 \leq |s| \leq 2|r|, \\ (1, 1) & \text{if } r > 0, s > 0 \\ & |r|/2 \leq |s| \leq 2|r|, \\ (0, 1) & \text{if } r < 0, s > 0 \\ & |r|/2 \leq |s| \leq 2|r|, \\ (0, 1) & \text{if } s > 0, \\ & 2|r| < |s|, \\ (0, 0) & \text{otherwise} \end{array} \right. \quad (2.23)$$

以上のような画素値の移動の結果、各領域の重心画素ではその領域の大きさを検出できる。 $w(i, j)$ の値は重心以外の多くの画素においては 0 である。ただし、本アルゴリズムは 8 近傍という限られた情報のみを用いて重心を求めているため、実画像に適用すると一つの領域に複数の重心が求まる。しかし、この問題は次のサイズ判別において、しきい値を適切に設定することにより逆に文字領域の判断基準として利用することができる。また、画素値の移動方向の検出、画素値の移動には、8 近傍画素の情報が必要となるため、各 PE の端に位置する画素の処理には、PE 間通信が行われる。

サイズ判別

領域の大きさに対するしきい値として、 Th_{big} 、 Th_{small} を設定する。各領域の重心となった画素ではこれらのしきい値と領域の画素数 $w(i, j)$ を比較する。その結果、

$Th_{\text{small}} \leq w(i, j) \leq Th_{\text{big}}$ の場合、もし $k(i, j) = 1$ であれば画素値を 1 にするよう 8 近傍画素に信号を送る。この信号を受けた画素はメモリに蓄えておいた自身の持つ二値画素 $k(i, j)$ が 1 の場合、さらに自身の属する領域の 8 近傍の画素に同じ信号を送る。以上を繰り返すことにより、文字としては不適当な大きさの領域は削除されることになる。 t ステップ目の画像 Y の画素値を $y^t(i, j)$ とし、この過程を式で表すと以下のようになる。

$$y^t(i, j) = \begin{cases} 1 & \text{if } \sum_{i-1 \leq l \leq i+1, j-1 \leq m \leq j+1} y^t(l, m) > 0 \text{ and } k(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2.24)$$

$$y^0(i, j) = \begin{cases} 1 & \text{if } Th_{\text{small}} \leq w(i, j) \leq Th_{\text{big}} \\ 0 & \text{otherwise} \end{cases} \quad (2.25)$$

まず、式 (2.25) により、適当な大きさの領域の中心のみが画素値 1 を持つことになる。次に、式 (2.24) を繰り返すことにより、その領域が復元されていく。式 (2.24) の繰り返し計算には、各画素の 8 近傍の値を参照する必要がある。そのため、注目画素が PE の端に位置する場合には、PE 間通信が行われる。

文字領域が復元されていく様子を図 2.15 に示す。

図 2.15 の例において、(a) ではうろこ模様に塗られた画素が重心として検出された画素を表している。斜線の画素が、重心方向への移動をする前において画素値を持つ画素である。(b)(c)(d) と、重心として検出された画素から信号が伝わる様子を表している。図 2.15 右下の領域は、大きさが小さいために削除されている。

この過程において、画像中の文字特有の重心の分裂の仕方を考慮し、適切なしきい値を設定することにより文字を抽出することが可能となる。

2.2.8 ノイズ除去過程

最後に次式に従い、ノイズの除去・図形の整形を行う。

$$z^{\text{new}}(i, j) = \begin{cases} 1 & \text{if } \sum_{y(l, m) \in P_{i, j, k}^8} y(l, m) \geq 3, \\ 0 & \text{otherwise} \end{cases} \quad (2.26)$$

式 (2.26) は、画素値が 1 の画素において、連結する画素が 2 つ以下の場合、その画素値を 0 にするものである。この操作を 2~3 回行う。これにより、文字領域に隣接

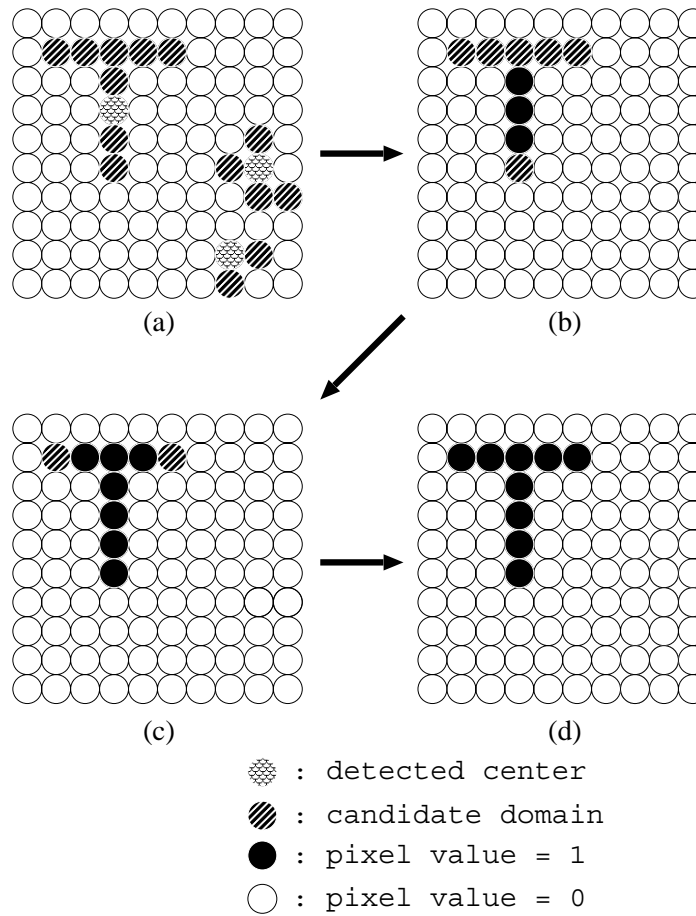


図 2.15 サイズ判別の様子

Fig. 2.15 The judgement of the size.

したノイズを除去し、抽出した図形の形を整えることが可能である。ノイズ除去は、8近傍のフィルタ処理であるため、各 PE の端に位置する画素を処理するためには、PE 間通信が必要となる。

以上が本章で提案する文字抽出アルゴリズムである。

2.3 計算機シミュレーション

提案アルゴリズムの有効性を示すため、情景画像を用いて文字を抽出する計算機シミュレーションを4通り行った。

1つ目は、通常の情景画像から文字を抽出する計算機シミュレーションを行い、著者による従来法と抽出精度を比較した。2つ目のシミュレーションでは、提案アルゴリズムのパラメータを変えて文字抽出シミュレーションを行い、抽出される文字の大きさとの関係について調査した。次に、光の当たり具合やコントラストの影響により抽出が困難と考えられる画像を集め、それらに対して提案アルゴリズムを実行し、提案アルゴリズムの頑健性を検討した。最後に、提案アルゴリズムを適用した場合の各PEの命令回数とPE間通信の回数を調べた。

2.3.1 情景画像からの文字抽出

シミュレーション条件

通常のデジタルカメラで撮影した256階調、縦480画素×横640画素、RGB各256階調のカラー画像を使用した。画像撮影時の天候は晴れまたは曇であり、日中あるいは夜間に撮影した画像である。景画像103枚の内、20枚を表2.1に示すパラメータ設定のために使用し、残りの83枚を用いて評価した。評価指標として、抽出率を次式のように定義した。

$$\text{抽出率 (\%)} = \frac{\text{抽出に成功した文字数}}{\text{画像に含まれる文字数}} \times 100 \quad (2.27)$$

式(2.27)において、「画像に含まれる文字」とは抽出すべき文字であり、入力画像の段階で小さすぎてつぶれてしまった文字などは含まない。また、「抽出に成功した文字」とは人間が読める程度に抽出できた文字である。本論文では、白抜きの文字は対象から外している。白抜きの文字に対しても、反転して抽出処理を行えば、抽出が可能であると考えられ、アルゴリズムの評価としては、白抜き文字を対象外としても問題ないと考えられるからである。

式(2.27)の抽出率だけでは、評価指標としては不十分である。文字に限らずなるべく多くのものを抽出すれば文字の抽出率も上がってしまうからである。そこで、文字以外のものを抽出した「誤抽出率」を定義する必要がある。誤抽出率を定量的に評価することは困難であるが、本章では次式のように定義する。

$$\text{誤抽出率 (\%)} = \frac{\text{抽出した非文字の画素数}}{\text{入力画像の全画素数}} \times 100 \quad (2.28)$$

提案アルゴリズムの目的は抽出率の向上と誤抽出率の抑制である。第 1 章で述べたように、本研究は視覚障害者支援として、代読サービスのように最終的には人間の判断が加わるものに対して使用されることを想定している。したがって、上記の抽出率のように、やや主観的な評価でもあまり問題はないと考えられる。しかしながら、他のアルゴリズムと比較して評価を行う場合は、より客観的な指標が必要となる。そこで、precision(適合率)とrecall(再現率)という指標を取り入れた。precision/recall は画像処理の分野だけでなく、情報検索の分野などでも用いられており、それぞれ以下の式で表される。

$$precision = \frac{\text{検索された正解数}}{\text{検索された数}} \quad (2.29)$$

$$recall = \frac{\text{検索された正解数}}{\text{全正解数}} \quad (2.30)$$

これらを提案アルゴリズムの評価のために置き換えると、それぞれ以下のように表すことができる。

$$precision(\%) = \frac{\text{抽出した文字の画素数}}{\text{抽出した画素数}} \times 100 \quad (2.31)$$

$$recall(\%) = \frac{\text{抽出した文字の画素数}}{\text{入力画像中の文字の画素数}} \times 100 \quad (2.32)$$

precision は、抽出した画素にどれだけ文字として正しく抽出した画素が含まれているかを示し、recall は抽出すべき画素をどれだけ抽出できたかを表す。正解が多ければ、precision と recall は両方とも高くなるが、文字以外の画素を抽出すると precision が下がる。本研究では、最終的に人間が介在するサービスを想定しているので、precision よりも recall の方が重要視される。余分なものが含まれていても、人間の判断で除くことができるが、必要な情報が含まれていない画像の文字を読むことは不可能だからである。

比較対象として文献 [90] のアルゴリズムを従来法として利用した。文献 [90] は提案アルゴリズムと同様に、局所処理のみで構成された文字抽出アルゴリズムである。これは他に味岡が提案する“視覚装置”のために開発された文字抽出手法がないからである。従来法の詳細は付録 A で述べる。

計算機シミュレーションの条件を表 2.1 に示す。パラメータ値は予備実験により、最適と思われる値を設定した。

表 2.1 文字抽出シミュレーションのパラメータ

parameter name	value
平滑化パラメータ ϵ	30
平滑化回数	10
最大値フィルタ回数	13
最小値フィルタ回数	10
二値化近傍長 (ベース画像生成過程)	15
二値化近傍長 (ノイズ抽出過程)	10
Th_{small}	5
Th_{big}	40

表 2.2 計算機シミュレーション結果

	提案アルゴリズム	従来法
抽出率 (%)	92.1	80.9
誤抽出率 (%)	1.97	4.15
precision(%)	60.9	42.6
recall(%)	77.9	50.1

シミュレーション結果

計算機シミュレーションの結果、提案アルゴリズムの抽出率は 92.1%、誤抽出率は 1.97% であり、従来法の抽出率は 80.9%、誤抽出率は 4.15% であった。提案アルゴリズムは従来法に比べ抽出率が高く、誤抽出率が低いことが分かる。また、precision/recall の指標も共に提案アルゴリズムが従来法を上回った。recall は“抽出率”と同様の評価指標であるが、提案法・従来法ともに、recall は抽出率よりも低い値となった。これは、抽出率は文字単位で計算し、recall は画素単位で計算しているためである。そのため、少しくらい欠けて抽出された文字を抽出率の計算の際に抽出された文字として数えている。

結果を表 2.2 にまとめる。また、評価に用いた入出力画像の例を図 2.16、図 2.17 に示す。そして結果画像をその次のページより掲載する。

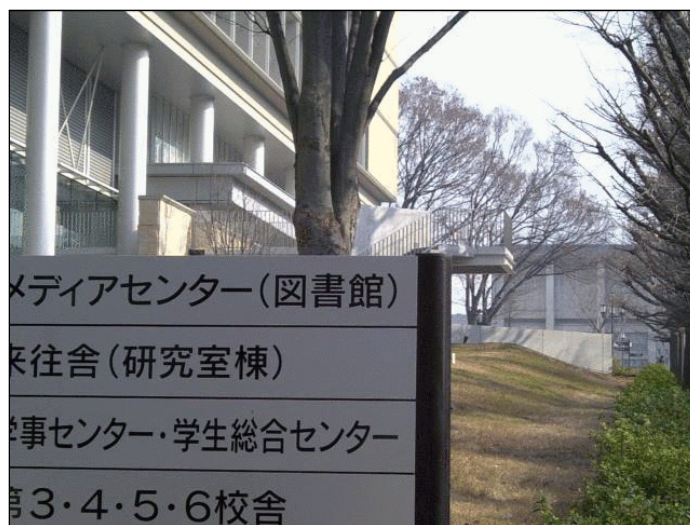


図 2.16 入力画像例

Fig. 2.16 An example of the input image.

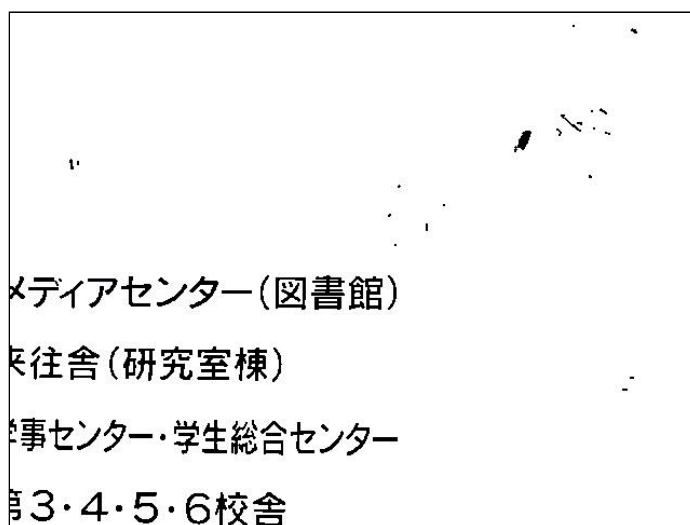
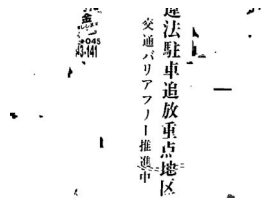


図 2.17 出力画像例

Fig. 2.17 An example of the output image.



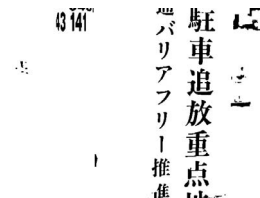
in01



out01



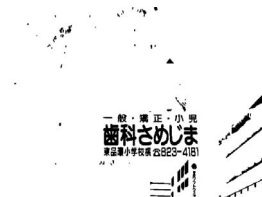
in02



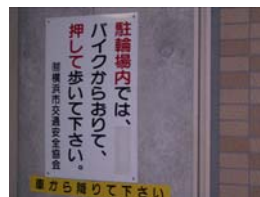
out02



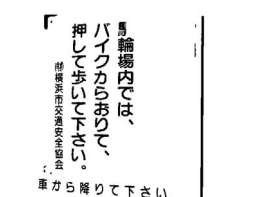
in03



out03



in04



out04



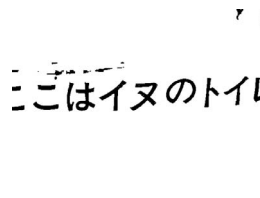
in05



out05



in06



out06



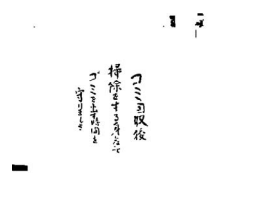
in07



out07



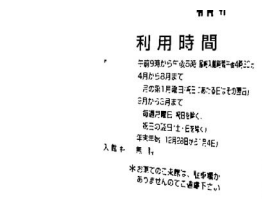
in08



out08



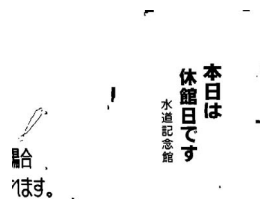
in09



out09



in10



out10

図 2.18 入出力画像 1-1

表 2.3 シミュレーション 2 のパラメータ

	Big	Middle	Small
平滑化パラメータ ϵ	30	30	30
平滑化回数	10	10	10
最大値フィルタ回数	15	13	10
最小値フィルタ回数	12	10	7
二値化近傍長 (平滑化後)	15	15	10
二値化近傍長 (文字消去後)	10	10	5
Th_{small}	10	5	5
Th_{big}	70	40	40

2.3.2 パラメータ変更による抽出率の変化

近傍長を変化させ文字抽出シミュレーションを行い、抽出される文字の大きさとパラメータとの関係を調査した。

シミュレーション条件

入力に用いた画像は 2.3.1 節と同じ通常のデジタルカメラで撮影した 256 階調、 640×480 画素の情景画像 103 枚であり、評価指標も同様に抽出率と誤抽出率を利用した。パラメータを表 2.3 のように設定し、それぞれのパラメータ条件において抽出率と誤抽出率を文字サイズごとに調べた。

シミュレーション結果

それぞれのパラメータセットで計算機シミュレーションを行い、それぞれの文字抽出率を文字サイズごとにまとめたグラフを図 2.19 に示す。また、このグラフにおいて、Conventional は文献 [90] のアルゴリズム適用結果である。図 2.19 のグラフの横軸は抽出対象の文字のおおよその大きさの一边を表している。例えば、横軸が 20 であれば文字の大きさが 20×20 の枠にちょうど収まる程度である。縦軸はそれぞれの大きさごとの文字の抽出率を表している。

この結果より、提案アルゴリズムではパラメータを変更することにより、任意の大きさに特化した文字抽出を行うことができることがわかる。

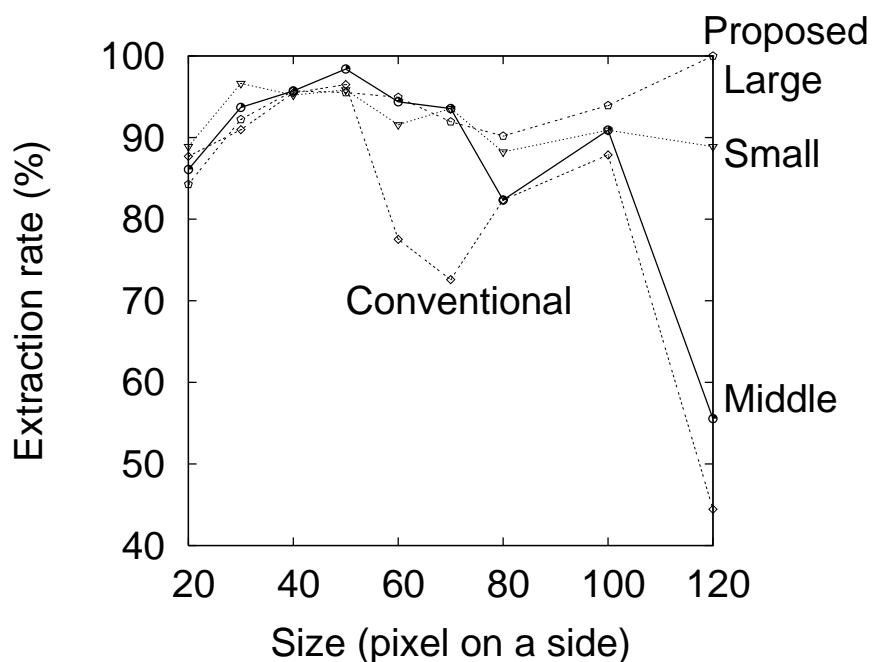


図 2.19 文字抽出特性

Fig. 2.19 The characteristic of the character extraction.

2.3.3 難易度の高い画像からの文字抽出

提案アルゴリズムの明るさの変化に対する頑健性を確認するため、比較的抽出が困難であると思われる照明条件で撮影した画像を対象にして計算機シミュレーションを行った。

シミュレーション条件

入力に用いた画像は通常のデジタルカメラで撮影した 256 階調、 640×480 画素の情景画像であるが、夕焼け時に撮影した画像やフラッシュを反射している画像など、比較的抽出が困難と思われる画像 35 枚である。提案アルゴリズムのパラメータは 1 つ目の文字抽出シミュレーションと同じ表 2.1 の設定で行った。比較対象としては、文字抽出シミュレーションでも用いた固定しきい値を使用する従来法と、従来法のしきい値を画像ごとに判別分析法 [91] を用いて決定した手法を用いた。従来法のしきい値は 100 とした。判別分析法は統計的に最適なクラスタリングを行う手法であり、良好な二値化結果が得られる手法として知られている。

表 2.4 計算機シミュレーション結果

	提案アルゴリズム	従来法	従来法 + 判別分析法
抽出率 (%)	83.3	63.1	70.6
誤抽出率 (%)	1.67	1.53	1.73

シミュレーション結果

シミュレーションの結果を表 2.4 に示す。

シミュレーションの結果から、提案アルゴリズムは従来法に比べ、照明条件に対して頑健であることが分かる。次ページより、シミュレーション結果を掲載する。



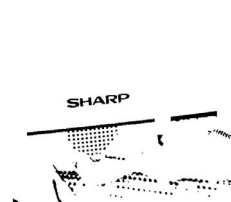
入力画像 01



提案法



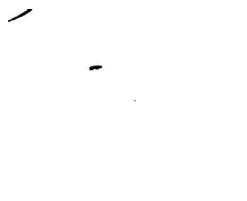
従来法



従来法+判別分析法



入力画像 02



提案法



従来法



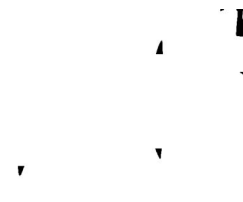
従来法+判別分析法



入力画像 03



提案法



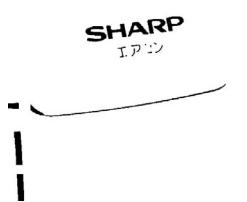
従来法



従来法+判別分析法



入力画像 04



提案法



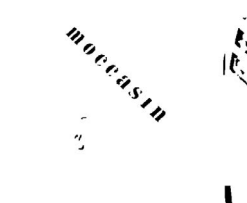
従来法



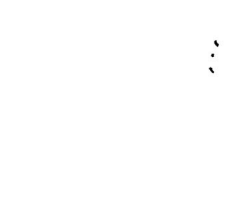
従来法+判別分析法



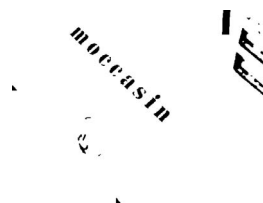
入力画像 05



提案法



従来法



従来法+判別分析法

図 2.20 入出力画像 2-1

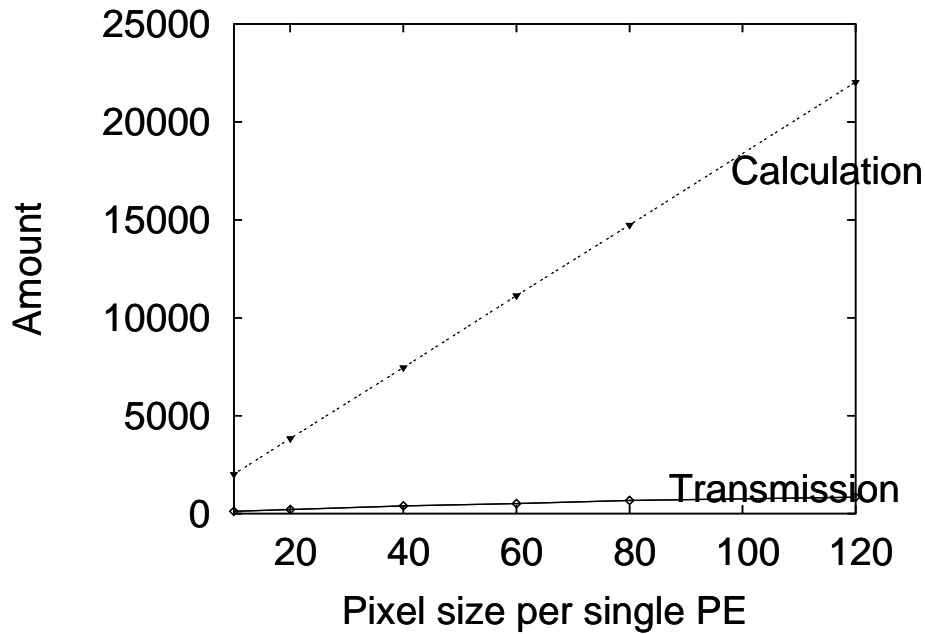


図 2.21 文字抽出アルゴリズムにおける各 PE の計算量と通信量

Fig. 2.21 Amount of calculation and transmission.

2.3.4 PE ごとの計算量および通信量の測定

提案アルゴリズムを実行した場合の、各 PE の計算回数と PE 間の通信回数を測定した。

シミュレーション条件

入力に用いた画像、パラメータ等はシミュレーション 1 と同様である。1PE 当たりが扱う画素数を変化させ、文字抽出アルゴリズムを実行した際の各 PE 計算回数と通信回数を測定した。

シミュレーション結果

シミュレーション 1 で用いたものと同じ画像 83 枚に対して文字抽出アルゴリズムを実行した結果を図 2.21 に示す。

図 2.21 において、横軸は一つの PE 当たりが扱う画素数を表し、縦軸は各 PE の四則演算・比較命令の回数を表している。このグラフでは、PE が扱う画素数が増えるにつれて、ほぼ比例するように計算回数・通信回数ともに増加している。計算回数が増加するのは各 PE が処理する画素数が増加するからである。通信回数が増加するのは

各 PE が扱う画素が増加するにつれて、PE の端に当たる画素が増加するためである。PE の端に当たる画素は、処理のために近傍画素情報を得るため PE 間通信しなければならない。各 PE が扱う画素が増加すれば、各 PE が扱う画素において、処理に通信が必要な画素が増えるため通信回数も増加する。

また、通信回数は計算回数に比べると非常に少ないことがわかる。このことから、提案アルゴリズムは局所並列処理を行う“視覚装置”に向いていると考えられる。また、このグラフから処理する画像サイズの大きさの増加には PE の数を増加することで対応できることがわかる。

2.4 考察

本章の提案アルゴリズムは、二値化処理を除き、全ての処理が8近傍以内の処理となっている。二値化処理においても、多近傍を参照するのは、式(2.10)の処理のみである。他の8近傍処理は何回も繰り返し演算を行うことが多いが、二値化処理内では、式(2.10)を繰り返し計算することではなく、一度だけしか計算しない。また、表2.1からも分かるように、式(2.10)で参照する近傍長は15である。1.6.1節で示した通り、各PEのサイズは 15×15 画素であるため、近傍長15の近傍画素を参照する場合には、ほとんどの画素の処理は必要なPE間通信が1回である。ただし、各PEは4近傍のPEとのみ接続していることから、PEの四隅に位置する画素の処理には、1.5.2節で述べた再転送が必要となる。各PEの四隅に位置する画素の処理では、8近傍を参照する演算においても再転送が必要であることから、二値化処理のみが再転送を必要とするわけではない。

上記の理由から、アルゴリズム全体の通信量に比べると、多近傍を参照する二値化処理で生じる通信量は少ないと考えられる。このことから1.6.4節で述べたように、提案アルゴリズムはほとんどの処理が8近傍以内の計算の繰り返しで構成されており、局所並列処理に向いていると言える。

提案アルゴリズムは、8近傍演算を繰り返すことで、多近傍を参照する場合と同様の情報を参照しながら計算している。そのため、計算コストの観点からは無駄な計算も行っている。例えば、2.2.6節の文字候補抽出過程では、式(2.17)を何度も繰り返すことにより不必要領域を削除している。しかしながら、計算コストを考慮すれば、各PE内において、該当領域を削除した方が良いと思われる。提案アルゴリズムにおいて、8近傍処理で表現しているのは、各処理を領域単位ではなく、画素単位で表現するためである。領域として処理すると、PEを跨いだ領域の処理など、表現が繁雑になってしまうが、画素単位で式を表現することにより、これを回避できる。

提案アルゴリズムでは、各処理を局所並列処理で実現するため、画素単位の計算に落とし込んで表現したものであり、実際にハードウェアに実装する際に、最適化すれば良いと言える。

提案アルゴリズムでは、ほとんどの処理が8近傍以内の処理であることから、1つのPEで必要となるPE間通信の量は、ほぼPEの一辺の長さに比例すると考えられる。8近傍処理で通信が必要となる画素は、PEの端に位置する画素のみなので、各PEが $r \times r$ の画素を処理する場合、PE間通信を必要とする画素数は、約 $4r$ となる。この時、各PEが処理しなければならない画素数は r^2 である。PEの一辺のサイズを2倍とした場合、PE間通信が必要となる画素数は約 $8r$ となり、各PEが処理しなければならない画素数は、 $4r^2$ である。この関係を図2.22に示す。

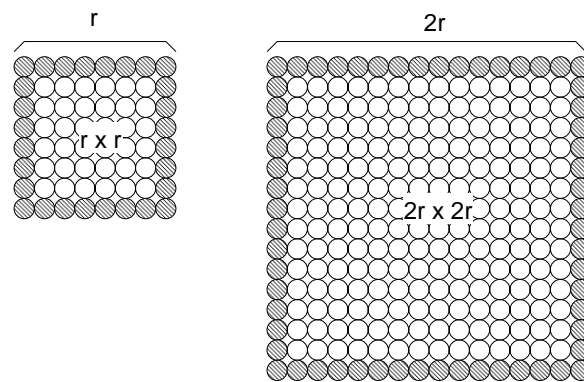


図 2.22 通信量と画素数の関係

Fig. 2.22 Relation between calculation amount and transfer amount.

各 PE に着目した場合、PE サイズを変更することにより増減する割合は、計算量と通信量で異なる。そのため、PE のサイズを小さくすれば各 PE の通信量は減るが、計算量が減る割合の方が大きくなるため、PE の計算量に対する通信量の割合は増加することになり、非効率な構成となる。逆に、PE サイズを大きくすると、各 PE の計算量が増えるため、処理時間は多くかかってしまう。

よって、PE のサイズは、福田らの研究結果 [67] から、 15×15 程度が適していると考えられる。

2.5 むすび

本章では局所的な処理のみで構成され並列化可能な、情景画像からの文字抽出アルゴリズムを提案した。

提案アルゴリズムは、視覚障害者の支援を前提にし、低コスト化、高速化のためにハードウェアに実装することを考慮している。高い精度を維持しながら、コスト削減のため画素ごとに情報を扱うデバイスを対象に処理の簡略化を重視したアルゴリズムとなっている。局所的な画素の情報のみを用いるという制約を課し、結果を高速に求めることにより、従来の画像処理で用いられているアルゴリズムとの差別化を図った。その結果、よりハードウェア化に向けたアルゴリズムであるといえる。

そして、計算機シミュレーションにより十分な文字抽出が可能であること、輝度に対するしきい値の設定が不要であり、照明条件に対して頑健であることを確認した。

第 3 章

局所並列処理のためのピクトグラム検出アルゴリズム

本章では、情景画像からピクトグラムを検出するアルゴリズムを提案する。ピクトグラムとは、意味するものの形状を使って、その意味概念を理解させる記号であり、国際的、即時的にわかる表現として注目を集めている。

提案アルゴリズムには全ての処理を局所的な計算のみで行うという制約が設けられている。このような制約により、空間的に画像を分割し、並列にアルゴリズムを実行することができる。

情景画像から、四角形領域を抽出し、ピクトグラムの候補とする。さらに、抽出されたピクトグラム候補から、東急電鉄の駅構内で使用されているピクトグラムを検出する。ピクトグラムの視認性を利用し、単純な計算から各ピクトグラムの特徴を求め、11 種類のピクトグラムを検出するアルゴリズムである。

提案アルゴリズムは、平滑化過程・多色分類過程・四角形抽出過程・輝度値変換過程・二値化過程・ピクトグラム分類過程からなる。平滑化過程では、エッジを保存しながら平滑化を行い、多色分類過程では、カラー画像を複数の二値画像へ変換する。四角形抽出過程では、画像中の四角形領域をピクトグラム候補として抽出する。二値化過程では、エッジ情報を利用して画像を二値化する。ピクトグラム分類過程では、各ピクトグラムの特徴を単純な計算で求め、ピクトグラムの種類を検出する。

実際に、駅構内で撮影した画像を用いた計算機シミュレーションを行い、提案アルゴリズムの有効性が確認された。



図 3.1 ピクトグラム例

Fig. 3.1 An example of pictogram

3.1 はじめに

近年、視覚障害者支援を目的とする画像処理の研究が盛んに行われている [23][26][28]。ピクトグラム抽出もその一つである。ピクトグラムとは、意味するものの形状を使って、その意味概念を理解させる図記号である。ピクトグラムは、駅やショッピングモールの中にあるトイレマークのように事前の学習なしでも、見ただけで誰にとってもすぐわかるという特徴を持っている。言語の違いというコミュニケーションの壁や教育や年齢や経験の差を超えて、国際的、即時的にわかる表現として注目を集めている。その有効性は広く認識され、多くの国で使用されている [92][93]。

図 3.1 に本論文で対象とするピクトグラムの例を示す。

ピクトグラムのような図形や看板などを画像から抽出する研究はあるが、従来の手法は計算量が多いという問題点がある [28][94]。これらはデスクトップ PC などで処理すれば処理時間にはさほど問題はないが、視覚障害者支援等の用途では携帯性が重要になる [77][78]。このような用途に対しては組み込みハードウェアによる処理が適していると考えられる。アプリケーションに対応した処理を直接ハードウェアで実行することにより、高い処理能力を実現することが画像処理の分野で盛んに行われている [48]–[67][83][84]。これらの多くは、処理速度を上げるために、並列処理を行っている。

ハードウェア化により処理の高速化が可能であるが、従来のアルゴリズムのハードウェア化は困難である。計算量が多く、大域的な処理や複雑な処理を必要とするからである。

そうした中、第 1 章で述べた通り、味噌は“視覚装置”という新しい概念の視覚システムを提案している [1][64]。

既に述べた通り、視覚装置はカメラと画像処理ハードウェアから構成されており、ハードウェアアーキテクチャは複数のプロセッシング・エレメント (PE) が格子状に並んだ構成をしている。それぞれの PE が、分割された画像の各部分を並列に処理することにより、高速な画像処理が実現できる。各 PE は 4 近傍の PE とのみ物理的な接続を通して非同期的に通信を行うが、順番にデータを転送することで論理的にはより大

きな近傍と通信を行うことも可能になっている。また、PEはお互いに同期して連続して局所処理をすることができる。そのような処理を本論文では局所並列画像処理と呼ぶことにする。この視覚装置の特長は低電力で並列に動作することである。そのため局所的な並列計算を高速に行うことができる。局所並列画像処理を行うシステムの大幅な小型化が実現可能であり、携帯電話への組み込みなどの用途でも大きな需要が見込まれる。

しかし、このような利点を活かすには、局所並列画像処理のみで完結するアルゴリズムの開発が不可欠である。従来のようにフーリエ変換やハフ変換などを利用するアルゴリズムやクラスタリングを用いるアルゴリズムなどのように、大域的な計算を必要とするアルゴリズムは局所並列処理を行うハードウェアへの実装には適していない。

そこで本章では、注目画素の近傍の局所的な情報のみを用いた処理によって情景画像からピクトグラムを抽出する新しいアルゴリズムを提案する。

筆者らは視覚障害者支援を目的とし、局所並列処理で構成されたアルゴリズムを既に提案している [74]–[76]。しかしながら、これらは具体的な応用について十分に述べられていない。そこで、本章では目的を絞り、駅構内で使用されているピクトグラムを検出し、種類の分類を行うアルゴリズムを提案する。より実用的なアプリケーションへの応用という側面から、既に提案したアルゴリズムを発展させることにより、局所並列画像処理を容易に応用できることを示す。

本章で対象とするのは、駅構内(東急電鉄)で使用されているピクトグラムである。図 3.2 に本章で検出対象としているピクトグラムを示す。検出対象のピクトグラムは東急電鉄のほぼ全駅で使用されているピクトグラムであり、ごく一部の駅でしか使用されていないピクトグラムは検出対象から除いている。図 3.2 のピクトグラムは、上段左からトイレ(男女)・乳幼児施設・車椅子・コインロッカー、中段左から、トイレ(男)・トイレ(女)・切符売場・エレベーター、下段左から、電車・上りエスカレーター・下りエスカレーターを表している。

なお、本章では図 3.3 のように、ピクトグラムの背景が白地のものを検出対象としている。

以下、3.2 節ではピクトグラム検出アルゴリズムについて説明する。3.3 節では計算機によるシミュレーションを行い、その有効性を示す。3.4 節で考察を述べ、3.5 節で本章をまとめる。

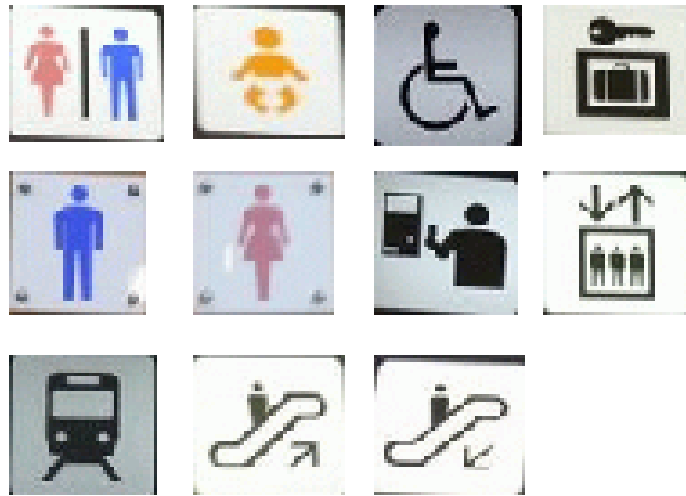


図 3.2 本章で対象とするピクトグラム

Fig. 3.2 Target pictograms.

3.2 駅構内画像からのピクトグラム検出アルゴリズム

3.2.1 概要

図 3.4 に提案アルゴリズムのおおまかな流れを示す。提案アルゴリズムは、ピクトグラムを抽出する処理と、抽出したピクトグラムの種類を分類する処理に分けられる。

ピクトグラム抽出処理では、入力画像に対し平滑化・多色分類・四角形抽出が行われる。多色分類過程では、文献 [74] の方法により、RGB の 3 帯域のカラー画像が白・黒・赤・青・橙の 5 帯域に分類される。多色分類の結果は、ピクトグラム分類処理でも利用するが、ピクトグラム領域抽出では、白色の帯域が利用される。四角形抽出過程では、ピクトグラムの枠を線分の長さや傾きから推定し、ピクトグラム領域の候補として抽出する。

ピクトグラム分類処理では、抽出されたピクトグラム候補領域において、輝度値変換・二値化・分類が行われる。輝度値変換された後、2.2.4 節の方法でエッジ情報に基づく二値化が行われる。この二値画像と多色分類で得られたカラー情報から、各ピクトグラムの特徴量を求め、ピクトグラムの種類を分類する。

3.2.2 節の近傍定義、3.2.3 節の平滑化、3.2.6 節の輝度値変換、3.2.7 節の二値化は第 2 章と同様である。次節より各過程について詳細に説明する。

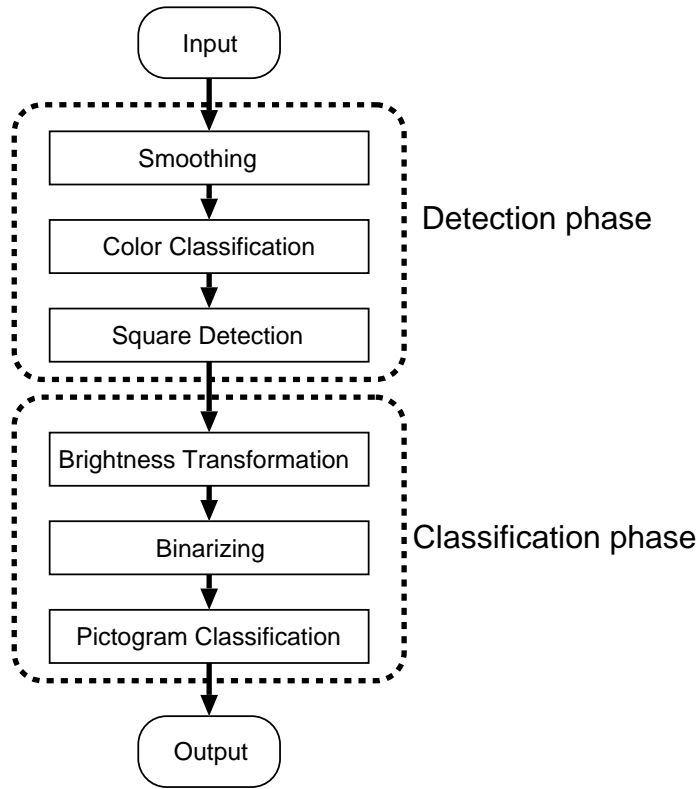


図 3.4 提案アルゴリズムの流れ

Fig. 3.4 Flow of the proposed algorithm.

3.2.3 平滑化過程

入力画像 X の画素値 $x(i, j)$ は次式に従って平滑化され、 $x^{\text{new}}(i, j)$ となる。

$$x^{\text{new}}(i, j) = \frac{1}{8} \sum_{i-1 \leq l \leq i+1, j-1 \leq m \leq j+1, l \neq i, m \neq j} \hat{x}_{i,j}(l, m) \quad (3.4)$$

式 (3.4) において、

$$\hat{x}_{i,j}(l, m) = \begin{cases} x(l, m) & \text{if } |x(i, j) - x(l, m)| \leq \epsilon \\ x(i, j) & \text{otherwise} \end{cases} \quad (3.5)$$

である。注目する画素との画素値の差が平滑化パラメータ ϵ を超える近傍画素値には、注目画素値が代入される。これによりエッジ部分のように急に画素値が変化する部分では平滑化が行われなくなる。このような平滑化を $x = r, g, b$ に対し繰り返す。平滑化において参照する画素は、注目画素の 8 近傍であるため、各 PE において、割り当てられた領域の端に位置する画素を処理する場合には、隣の PE との通信を行う必要がある。

3.2.4 多色分類過程

多色分類過程では、カラー画像から色の成分を抽出する。 $k = \{r, g, b\}$ の3帯域の入力画像 $k(i, j)$ から、補色関係にある白色 (*Wht*)、黒色 (*Blk*)、赤色 (*Red*) 及び青色 (*Blu*) の抽出を行う。このとき、各補色関係は q 近傍を用いて式 (3.6) ~ (3.9) のように計算される。

$$d_{Wht}(i, j) = \max_k \{q \cdot k(i, j) - \sum_{k(l, m) \in P_{i, j}^q} k(l, m)\} \quad (3.6)$$

$$d_{Blk}(i, j) = -\max_k \{q \cdot k(i, j) - \sum_{k(l, m) \in P_{i, j}^q} k(l, m)\} \quad (3.7)$$

$$d_{Red}(i, j) = q\{r(i, j) - g(i, j)\} \quad (3.8)$$

$$d_{Blu}(i, j) = -q\{r(i, j) - g(i, j)\} \quad (3.9)$$

ここで式 (3.6) は白色らしさ、式 (3.7) は黒色らしさ、式 (3.8) は赤色らしさ、式 (3.9) は青色らしさを表している。

次に各色らしさにおいて、値が最も高い画素を選択する。 $k = \{Wht, Blk, Red, Blu\}$ の4つの帯域に変換された二値画像 Y_k の各画素 $y_k(i, j)$ は以下のように求められる。

$$y_{Wht}(i, j) = \begin{cases} 1 & \text{if } d_{Wht}(i, j) = \max_k \{d_k(i, j)\}, \\ & d_{Wht}(i, j) > f_1 \\ 0 & \text{otherwise} \end{cases} \quad (3.10)$$

$$y_{Blk}(i, j) = \begin{cases} 1 & \text{if } d_{Blk}(i, j) = \max_k \{d_k(i, j)\}, \\ & d_{Blk}(i, j) > f_2, \\ & d_{Blk}(i, j) \neq d_{Wht}(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (3.11)$$

$$y_{Red}(i, j) = \begin{cases} 1 & \text{if } d_{Red}(i, j) = \max_k \{d_k(i, j)\}, \\ & d_{Red}(i, j) > f_3, \\ & d_{Red}(i, j) \neq d_{Wht}(i, j), \\ & d_{Red}(i, j) \neq d_{Blk}(i, j), \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

$$y_{Blu}(i, j) = \begin{cases} 1 & \text{if } d_{Blu}(i, j) = \max_k \{d_k(i, j)\}, \\ & d_{Blu}(i, j) > f_4, \\ & d_{Blu}(i, j) \neq d_{Wht}(i, j), \\ & d_{Blu}(i, j) \neq d_{Blk}(i, j), \\ & d_{Blu}(i, j) \neq d_{Red}(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (3.13)$$

ここで、 $f_1 \sim f_4$ はしきい値である。この結果、式 (3.10) により白色、式 (3.11) により黒色、式 (3.12) により赤色、式 (3.13) により青色でありそうな画素が選択される。式 (3.10) ~ 式 (3.13) は順序関係を持って定義されている。まず式 (3.10)、式 (3.11) で白黒の明度を基準に抽出し、その後に式 (3.12)、式 (3.13) で色を抽出しようとしている。これは色の差よりも明度の差の方が、輪郭を抽出するのに重要であると考えられるからである。

式 (3.6) ~ 式 (3.13) によって、 $\{r, g, b\}$ の3帯域の入力画像は、 $k = \{Wht, Blk, Red, Blu\}$ の4帯域の2値画像 Y_k へ変換される。変換後の各帯域では画素値は0または1の2値となる。

さらに、以下の式に従い、赤色 (*Red*) を赤色 (*Red*) と橙色 (*Ora*) に再分類する。

$$y_{Ora}(i, j) = \begin{cases} 1 & \text{if } y_{Red}(i, j) = 1 \\ & \text{and } g(i, j) > 1.5 \cdot b(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (3.14)$$

$$y_{Red}(i, j) = \begin{cases} 0 & \text{if } y_{Ora}(i, j) = 1 \\ y_{Red} & \text{otherwise} \end{cases} \quad (3.15)$$

結果的に、 $\{r, g, b\}$ の3帯域の入力画像は、 $k = \{Wht, Blk, Red, Blu, Ora\}$ の5帯域の2値画像 Y_k へ変換される。

多色分類した画像の例を図 3.5 に示す。本論文で提案するピクトグラム抽出アルゴリズムにおいては、このうち *Wht* の帯域を利用する。これはピクトグラムの枠には視認性を良くするため高いコントラストが存在するためである。*Wht* の帯域は周囲よりも明るい部分を抽出するので、ピクトグラムの枠を抽出するのに適していると考えられる。

多色分類過程では、各色らしさを計算するため、式 (3.6) ~ (3.9) で、注目画素の q 近傍の画素値を参照している。そのため、式 (3.6) ~ (3.9) の計算では、その近傍画素が自己 PE が扱う範囲外に位置する場合には、PE 間通信が必要となる。また、式 (3.10) ~ 式 (3.15) において、各画素の色を決定する際には、近傍画素を参照する必要はないため、PE 間通信を行う必要はない。



図 3.5 多色分類例

Fig. 3.5 An example of color classification.

3.2.5 四角形抽出過程

図 3.6 に四角形抽出過程の流れを示す。

四角形抽出過程では、ピクトグラムを構成する四角形を推定するために、文献 [76] と同様の手法で、水平方向・垂直方向で画素値が連続している値を求めることにより、水平・垂直の線分の長さを推定する。提案アルゴリズムでは文献 [76] の手法を拡張し、多色分類過程で得た Y_{Wht} から水平・垂直の線分の長さや傾きを 2 通りの方法で推定する。一方は右回りに求める方法であり、水平方向に画素値を集積した後、垂直方向に画素値を集積する。もう一方は左回りに長さを推定するため、水平方向と垂直方向の順序を入れ換え、垂直方向に画素値を集積した後、水平方向に画素値を集積する。この様子を図 3.7 に示す。また、画素値の連続数だけではなく傾きも求めることにより、精度の向上を図る。計測された画素集積値・傾きをもとに四角形を推定し、ピクトグラム領域として抽出する。

右回り画素集積法

座標 (i, j) における水平方向の連続画素数 $h(i, j)$ 、垂直方向の連続画素数 $v(i, j)$ 、水平方向計測時の傾き $gr_h(i, j)$ 、垂直方向計測時の傾き $gr_v(i, j)$ を以下の式に従い初期化する。

$$h^0(i, j) = v^0(i, j) = y_{Wht}(i, j) \quad (3.16)$$

$$gr_h^0(i, j) = gr_v^0(i, j) = 0 \quad (3.17)$$

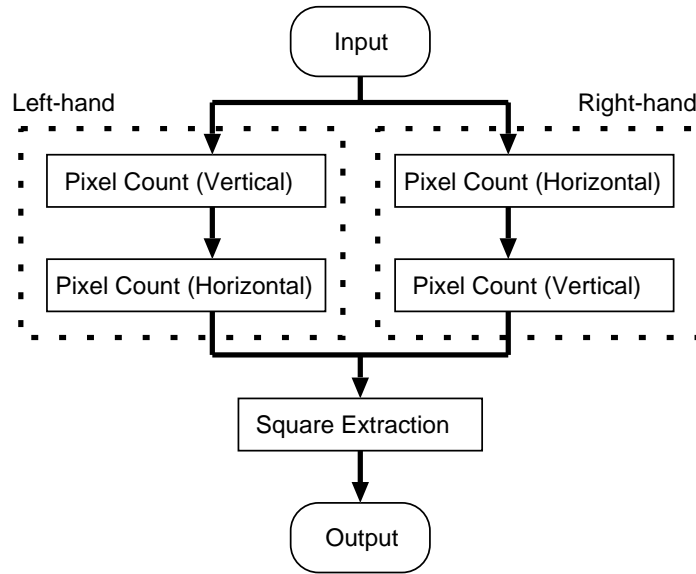


図 3.6 四角形抽出過程の流れ

Fig. 3.6 Flow of square extraction.

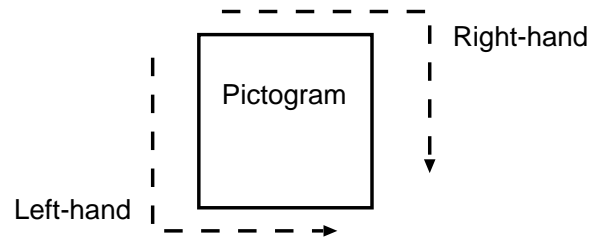


図 3.7 画素集積値・傾きの計算

Fig. 3.7 Calculation of connection and gradient.

次に、繰り返し回数を t として、以下の式を繰り返し、水平方向の連続数を数え、画素値を右方向へ集積する。

$$h^{t+1}(i, j) = \begin{cases} h^t(i, j) + 1 & \text{if } \max_{j-1 \leq m \leq j+1} \{h^t(i-1, m)\} = h^t(i, j) \\ & \text{and } h^t(i, j) \neq 0 \\ h^t(i, j) & \text{otherwise} \end{cases} \quad (3.18)$$

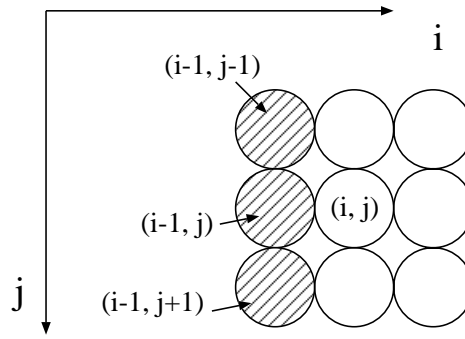


図 3.8 水平方向画素集積のための参照画素

Fig. 3.8 Referential pixels for horizontal pixel count.

ここで、 $\max_{j-1 \leq m \leq j+1} \{h^t(i-1, m)\}$ のときの m を m_{max} とすると、

$$gr_h^{t+1}(i, j) = \begin{cases} gr_h^t(i-1, j) & \text{if } gr_h^t(i-1, j) > gr_h^t(i, j) \\ & \text{and } m_{max} = j \\ gr_h^t(i-1, m_{max}) + 1 & \text{if } gr_h^t(i-1, m_{max}) > gr_h^t(i, j) \\ & \text{and } m_{max} \neq j \\ gr_h^t(i, j) & \text{otherwise} \end{cases} \quad (3.19)$$

式 (3.18) では、注目画素がその左の画素値と同じ値ならば連続数を増やす。式 (3.19) では、その時に参照した画素が真横ではない場合、傾きを検出しながら伝達する。 $h(i, j)$ は最初は 0 または 1 の二値画像なので、式 (3.18)、(3.19) を繰り返すことにより、一番右側の画素に連続数と傾きが集積されることになる。また、傾きのみならずノイズや歪みなどにも対応するため、図 3.8 に示すように式 (3.18) では m の定義域を $j-1 \leq m \leq j+1$ とすることにより、斜め方向も考慮されている。図 3.8 では、注目画素を (i, j) は左隣の画素 $(i-1, j)$ の他に、その上下の画素 $(i-1, j-1)$ と $(i-1, j+1)$ を参照している。図中の斜線部が参照画素を表している。しかしながら、無条件に斜め方向の画素の参照してしまうと、水平に対し斜め 45° の直線も水平の直線と同等に扱ってしまうので、式 (3.19) により、傾きも同時に計算している。

続いて、同様の手法で、垂直方向の連続数 $v(i, j)$ および傾き $gr_v(i, j)$ の算出と、水平方向の連続数と傾き $(h(i, j), gr_h(i, j))$ の伝達を行う。垂直方向の連続数を集積すると同時に、水平方向の連続数と傾きを伝達することにより、右下の画素に水平および垂直の連続数と傾きが集積されることになる。

画素の連続数を水平方向に集積する際に参照する画素は、注目画素の左・左上・左下に位置する画素のみである。これらの画素が PE の範囲外にある場合には、PE 間に

において通信が必要となる。

左回り画素集積法

水平方向と垂直方向の順序を変えて同様に連続数と傾きを計算する。すなわち、先に垂直方向へ連続数と傾きを計算した後、水平方向へ連続数と傾きを計算する。水平方向へ計算する時には、垂直方向の連続数と傾きの伝達も同時に行う。

四角形領域抽出

これまでに求めた連続数と傾きをもとに、四角形領域の推定を行う。右回りに求めた場合は添え字を 1 とし、水平方向の連続数と傾き・垂直方向の連続数と傾きをそれぞれ $h_1(i, j)$, $gr_{h1}(i, j)$, $v_1(i, j)$, $gr_{v1}(i, j)$ とする。同様に左回りに求めた水平方向の連続数と傾き・垂直方向の連続数と傾きをそれぞれ $h_2(i, j)$, $gr_{h2}(i, j)$, $v_2(i, j)$, $gr_{v2}(i, j)$ とする。以下の式に従い、四角形領域であるかの判定と、一辺の長さの推定を行い、 $Pict^0$ とする。

$$pict^0(i, j) = \begin{cases} \min_{k=1,2} [h_k(i, j), v_k(i, j)] & \text{if } \min_{k=1,2} [h_k(i, j), v_k(i, j)] > S_{pict} \\ & \text{and } rate[h_k(i, j), v_k(i, j)]_{k=1,2} > R_{pict} \\ & \text{and } rate[h_k(i, j), gr_{hk}(i, j)]_{k=1,2} > R_{grad} \\ & \text{and } rate[v_k(i, j), gr_{vk}(i, j)]_{k=1,2} > R_{grad} \\ 0 & \text{otherwise} \end{cases} \quad (3.20)$$

$$rate[x, y] = \min[x, y] / \max[x, y] \quad (3.21)$$

式 (3.20) では、連続数がしきい値 S_{pict} 以下の小領域の場合はピクトグラムではない、あるいは確信度が低いとみなし抽出しない。また、求めた四角形の水平方向・垂直方向の連続数と傾きの比がそれぞれ R_{pict} 、 R_{grad} 以上であれば、その領域を四角形とみなす。そして、四角形領域の右下頂点にあたる画素 (i, j) の $pict^0(i, j)$ が四角形の一辺の長さを示すことになる。

以下の式を繰り返すことにより、ピクトグラム候補領域 $Pict$ の各画素 $pict(i, j)$ が値

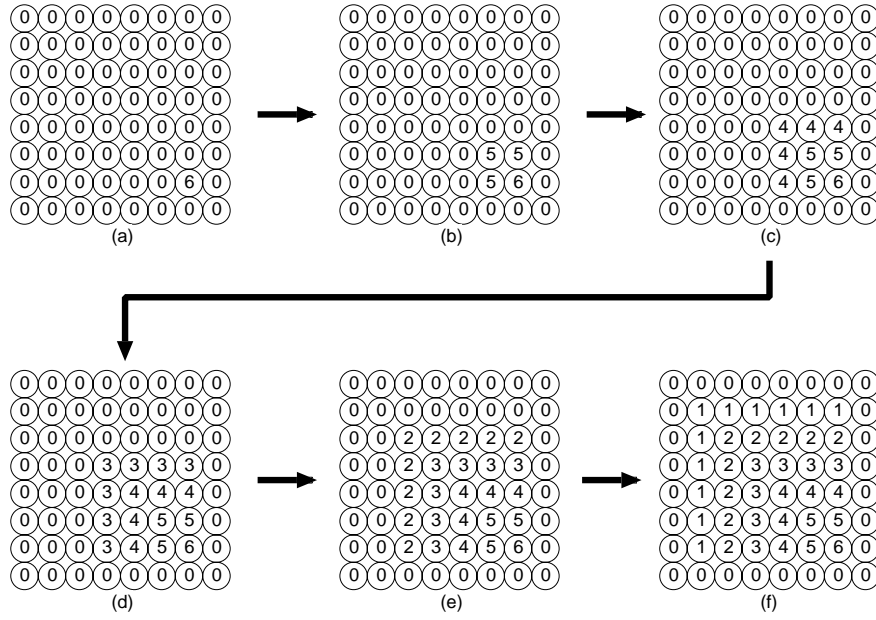


図 3.9 ピクトグラム抽出処理

Fig. 3.9 Pictogram extraction

を持つ。

$$pict^{t+1}(i, j) = \begin{cases} \max_{i \leq l \leq i+1, j \leq m \leq j+1} pict^t(l, m) - 1 & \text{if } \begin{aligned} &\max_{i \leq l \leq i+1, j \leq m \leq j+1} pict^t(l, m) > pict^t(i, j) \\ &\text{and } \max_{i \leq l \leq i+1, j \leq m \leq j+1} pict^t(l, m) > 0 \end{aligned} \\ pict^t(i, j) & \text{otherwise} \end{cases} \quad (3.22)$$

式 (3.22) では、注目画素の右・右下・下の画素値 $pict(l, m)$ が注目画素の $pict(i, j)$ よりも大きい場合、 $pict(i, j)$ を $pict(l, m)$ の最大値から 1 を引いた値に更新する。これにより、右下から左上方向に画素値が減少しながら伝達され、求めた連続数分だけ、伝達を繰り返すことができる。この様子を図で表すと図 3.9 のようになる。

次に、四角形領域内の画素に、その四角形の一辺の長さを設定する。以下の式により、四角形領域の大きさ $size(i, j)$ を初期化する。

$$size^0(i, j) = pict(i, j) \quad (3.23)$$

次式を繰り返すことにより、同一四角形領域内の画素に、四角形の一辺の長さ $size(i, j)$ を伝達することができる。

$$size^{t+1}(i, j) = \begin{cases} \max_{i \leq l \leq i+1, j \leq m \leq j+1} size^t(l, m) & \text{if } size^t(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.24)$$

次に、各画素にピクトグラム領域での x 座標 Pos_x 、 y 座標 Pos_y を設定する。

$$pos_x^0(i, j) = size(i, j) \quad (3.25)$$

$$pos_x^{t+1}(i, j) = \begin{cases} pos_x^t(i+1, j) - 1 & \text{if } pos_x^t(i+1, j) = pos_x^t(i, j) \\ & \text{and } pos_x^t(i, j) > 0 \\ pos_x^t(i, j) & \text{otherwise} \end{cases} \quad (3.26)$$

式 (3.25) により初期化を行い、式 (3.26) を繰り返すことにより、ピクトグラム領域内における x 座標を各画素に設定することができる。 y 座標についても、式 (3.25)、(3.26) と同様に設定する。

式 (3.20) の四角形判定には、近傍画素を参照する必要はないので、PE 間の通信は行われない。しかし、式 (3.22) と式 (3.24) では、注目する画素の右・右下・下の画素を参照するため、これらが PE の範囲を越える場合には、PE 間において通信が必要となる。

以上が本章で提案するアルゴリズムのピクトグラム抽出処理である。

3.2.6 輝度値変換過程

抽出したピクトグラムの種類を分類するため、入力画像の RGB 値を輝度値に変換する。輝度値への変換に使用する情報は、各画素の情報のみであるため、PE 間の通信が行われることはない。

3.2.7 二値化過程

次に、輝度画像を二値画像 Bin へと変換する。提案アルゴリズムにおける二値化は局所処理のみによる二値化処理 [75] を行う。詳細な説明は 2.2.4 節で述べたので、本章では要点のみを述べる。

二値化過程の流れ

画像中のエッジ情報を利用して二値化を行う。二値化する際に重要な手がかりとなるのは、領域の境界であるエッジ情報だと考えるからである。入力される輝度画像に対し、平滑化・エッジ抽出・しきい値決定・二値化を行う。図 3.10 に、提案アルゴリズムにおける二値化処理の大まかな流れを示す。

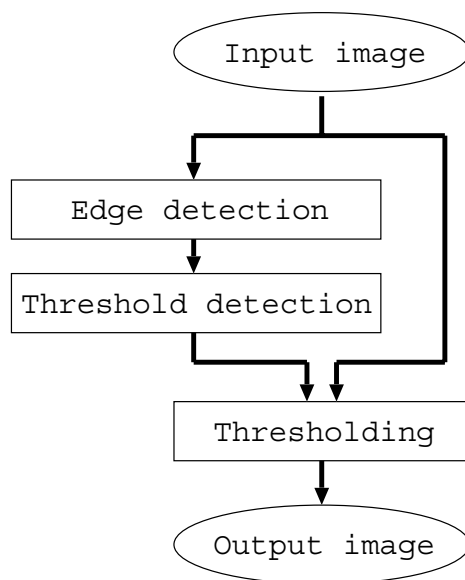


図 3.10 二値化の流れ

Fig. 3.10 Procedure of thresholding.

エッジ抽出

輝度画像に対し、ラプラシアンフィルタを利用しエッジ抽出を行い、輝度画像からエッジ画像を生成する。

しきい値の算出

エッジ抽出を行った後、エッジ画素において、近傍領域のしきい値の候補となる値を求める。エッジとして抽出された画素の近傍において、最も画素値の勾配が急な部分を求め、その中間値をしきい値の候補とする。次に、しきい値候補からしきい値を求める。全ての画素において、注目画素の近傍のしきい値候補の平均を求め、その値を各画素のしきい値とする。

二値化

各画素ごとに得られたしきい値を用い、輝度画像を二値化し、二値画像 Bin を求め、 Bin の各画素を $bin(i, j)$ と表す。

3.2.8 ピクトグラム分類過程

図 3.11 にピクトグラム分類の流れを示す。図 3.11 は上から順に分類処理が適用さ

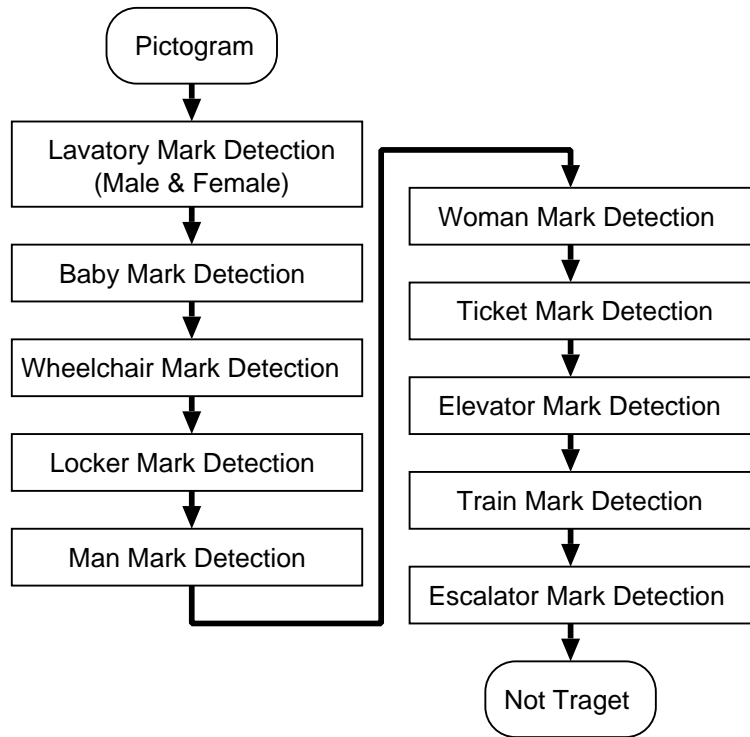


図 3.11 ピクトグラム分類の流れ

Fig. 3.11 Flow of the pictogram classification.

れる。まずトイレマーク (男女) の検出処理が行われ、トイレマーク (男女) として検出されなかったピクトグラム領域に対し、乳幼児施設マークの検出処理が行われる。すなわち、図 3.12 に示すように、各ピクトグラムの検出処理直前に、そのピクトグラムの特徴量を求める。これにより、余計な計算コストをかけることを防いでいる。各分類処理では単純な計算によりピクトグラムの特徴を求めている。したがって、分類処理の順序が結果に影響を与えることとなる。提案アルゴリズムでは、類似性の高いピクトグラムの場合、より細かい特徴を持つピクトグラムを先に検出するという方針で順序を決定した。

次節より、各ピクトグラムの検出方法について、アルゴリズム内の処理順に述べる。

トイレマーク (男女) 検出

トイレマーク (男女) は、ピクトグラム領域内に赤と青を含むかどうかにより検出を行う。この処理には 3.2.4 節の多色分類で求めた赤と青の帯域を利用する。

まず、ピクトグラム抽出過程で得られたピクトグラム画像 $Pict$ のうち、内部に Y_{Red} 成分が含まれる領域のみを取り出し、 U とする。 U の各画素 $u(i, j)$ は以下の式により

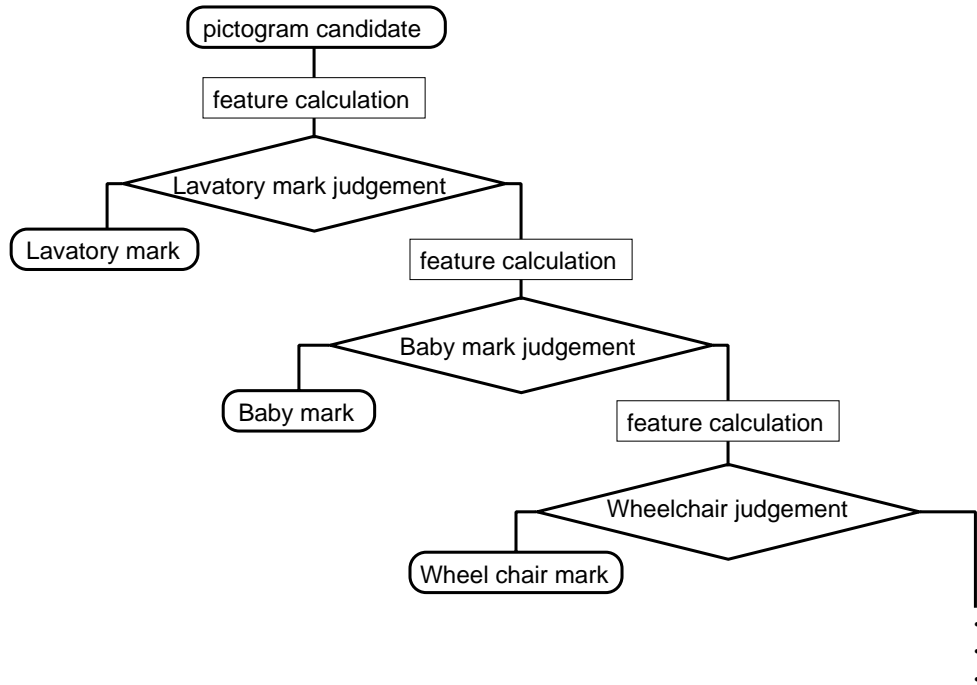


図 3.12 ピクトグラム分類の判断木

Fig. 3.12 Pictogram classification

計算される。

$$u^0(i, j) = \begin{cases} 1 & \text{if } y_{Red}(i, j) = 1 \text{ and } pict(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.27)$$

$$u^{t+1}(i, j) = \begin{cases} 1 & \text{if } \sum_{i-1 \leq l \leq i+1, j-1 \leq m \leq j+1} u^t(l, m) > 0 \\ & \text{and } pict(i, j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.28)$$

式 (3.28) を繰り返すことにより、 Y_{Red} 成分が含まれる四角形領域を取り出す事ができる。

次に、 U からさらに Y_{Blu} 成分が含まれる四角形領域を抽出し、出力画像 Z を得る。 Z の各画素 $z(i, j)$ は繰り返し回数を t として以下の式により計算される。

$$z^0(i, j) = \begin{cases} 1 & \text{if } y_{Blu}(i, j) = 1 \text{ and } u(i, j) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.29)$$

$$z^{t+1}(i, j) = \begin{cases} 1 & \text{if } \sum_{i-1 \leq l \leq i+1, j-1 \leq m \leq j+1} z^t(l, m) > 0 \\ & \text{and } u(i, j) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.30)$$

式 (3.30) も式 (3.28) と同様に繰り返し行う。 Z がトイレマーク (男女) のピクトグラムとなる。

式 (3.28)、式 (3.30) の繰り返し処理においては、注目画素の 8 近傍を参照するため、その参照画素が PE の範囲を越えた場合は、PE 間通信によって値を取得しなければならない。

乳幼児施設マーク検出

乳幼児施設はトイレマークの場合と同様に、ピクトグラム領域内に橙色帯域 Y_{Ora} の画素 $y_{Ora}(i, j)$ を含むもののみ抽出することにより乳幼児施設マーク $Baby$ として検出する。

車椅子マーク検出

車椅子マークは、マークが一定の太さの線で描かれていることに着目し、検出を行う。細線化を一定回数行い、車椅子マークを消去する。細線化を行った後もピクトグラム領域内に画素値が残っていれば、車椅子マークではないと判断する。ピクトグラム領域内の各画素に以下の式に従い、カウンタ $count(i, j)$ を設定する。

$$count(i, j) = size(i, j) \cdot C_{chair} \quad (3.31)$$

式 (3.31) において、 C_{chair} はパラメータである。そして、 $count(i, j)$ 回だけピクトグラム領域内の二値画像 Bin に対し、細線化処理を行う。 $count(i, j)$ 回だけ細線化した後も $bin(i, j) > 0$ の画素がピクトグラム領域内にある場合は、そのピクトグラムは車椅子マークではないとみなす。式 (3.28) と同様の手法で、細線化後も $bin(i, j)$ に値を持つ画素を含むピクトグラム領域を \hat{C}_{chair} とする。その後、 $Pict$ から \hat{C}_{chair} を引き、車椅子マーク $Chair$ を求める。

細線化では、注目画素の 8 近傍を参照するため、その参照画素が PE の範囲を越えた場合は、PE 間通信によって値を取得しなければならない。

次に、上記の処理で車椅子マークとして検出されなかったピクトグラム領域に対し、3.2.8 節と同様の手法で青色帯域 Y_{Blu} を含むピクトグラム領域を検出する。図 3.3 の左上にあるように、車椅子マークには青色が使用されることが多いからである。そして、青色を含むピクトグラムであると検出された場合、以下の式に従い、該当ピクトグラム領域の下に乳幼児施設マークがあるか調べる。図 3.3 の左上に示すように、車椅子

マークの下には乳幼児施設マーク *Baby* がある場合がほとんどだからである。

$$chair^0(i, j) = \begin{cases} 1 & \text{if } baby(i + (size(i, j)/2), j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.32)$$

青色を含むピクトグラム領域の各画素に対し、式 (3.32) を適用することにより、その下に乳幼児施設マークがあれば $chair^0(i, j)$ が値を持つ。 $chair^0(i, j)$ をもとに、式 (3.28) と同様の手法で車椅子領域を検出する。下に乳幼児施設マークがあるか調べるためには、参照する画素の位置は、式 (3.32) において定義されており、その位置が PE の範囲を越える場合には、PE 間通信が行われる。

以上の 2 通りの処理で得た *Chair* の和集合を車椅子マークのピクトグラムとして検出する。

コインロッカーマーク検出

コインロッカーマークは、ピクトグラム領域下部分の四角形 *Box* と上部分の鍵マーク *Key* に着目して検出を行う。ピクトグラム領域内において、再び 3.2.5 節で行った、水平方向および垂直方向の画素値の連続数の計算を行う。連続数の計算後、式 (3.20) の代わりに以下の式を用いて判定を行う。この判定では、PE 間通信は行われない。

$$box(i, j) = \begin{cases} \min_{k=1,2} [h_k(i, j), v_k(i, j)] & \text{if } \min_{k=1,2} [h_k(i, j), v_k(i, j)] > size(i, j) \cdot S_{box} \\ & \text{and } R_{box}^{min} < rate[h_k(i, j), v_k(i, j)]_{k=1,2} < R_{box}^{max} \\ & \text{and } h_1(i, j) > v_1(i, j), \quad h_2(i, j) > v_2(i, j) \\ & \text{and } rate[h_k(i, j), gr_{hk}(i, j)]_{k=1,2} > R_{grad} \\ & \text{and } rate[v_k(i, j), gr_{vk}(i, j)]_{k=1,2} > R_{grad} \\ 0 & \text{otherwise} \end{cases} \quad (3.33)$$

つづいて、式 (3.22) と同様の手順で四角形領域 *Box* を求める。*Box* はコインロッカーマークの下部分にある箱部分を表す。ここで式 (3.28) と同様の手順で $box(i, j)$ を含む

ピクトグラム領域を求め、*Locker*とし、以下の式に従い、鍵部分 Key を抜き出す。

$$key(i, j) = \begin{cases} 1 & \text{if } locker(i, j) > 0 \\ & \text{and } box(i, j) = 0, \ bin(i, j) > 0 \\ & \text{and } pos_x(i, j) > size(i, j) \cdot P_{key}^{xmin} \\ & \text{and } pos_x(i, j) < size(i, j) \cdot P_{key}^{xmax} \\ & \text{and } pos_y(i, j) < size(i, j) \cdot P_{key}^y \\ 0 & \text{otherwise} \end{cases} \quad (3.34)$$

ピクトグラム領域内の $key(i, j)$ の総和がしきい値 $size(i, j) \cdot Th_{key}$ を超えれば、その領域をコインロッカーマークとして抽出する。

総和を求めるには、以下の式を利用する。このとき、注目画素の左、上の画素値をそれぞれ参照するので、その範囲が PE を越える場合には、PE 間で通信を行う必要がある。

$$Key^{t+1}(i, j) = \begin{cases} Key^t(i, j) + Key^t(i-1, j) & \text{if } t = Pos_x(i, j) \\ Key(i, j)^t & \text{otherwise} \end{cases} \quad (3.35)$$

$$Key^{t+1}(i, j) = \begin{cases} Key^t(i, j) + Key^t(i, j-1) & \text{if } t = Pos_y(i, j) \\ Key(i, j)^t & \text{otherwise} \end{cases} \quad (3.36)$$

トイレマーク (男) 検出

トイレマーク (男) は 3.2.8 節のトイレマーク (男女) と同様に、ピクトグラム領域内に青色帯域 Y_{Blu} の画素 $y_{Blu}(i, j)$ を含むもののみに抽出することにより検出する。

トイレマーク (女) 検出

トイレマーク (女) も 3.2.8 節のトイレマーク (男) と同様に、ピクトグラム領域内に赤色帯域 Y_{Red} の画素 $y_{Red}(i, j)$ を含むもののみに抽出することにより検出する。

切符売場マーク検出

切符売場マーク $Booth$ は、ピクトグラム領域内の右下にある人の体部分 $Body$ と左上にある券売機部分 $Ticket$ に着目して検出を行う。3.2.8 節同様、ピクトグラム領域内において、水平方向および垂直方向の画素値の連続数の集積を行う。画素値の連続数集積には、8 近傍画素を参照するため、PE 間通信が必要となる場合がある。

式 (3.33) の代わりに以下の式を用いて人間の体部分の判定を行う。

$$body(i, j) = \begin{cases} 1 & \text{if } \min_{k=1,2} [h_k(i, j), v_k(i, j)] > size(i, j) \cdot S_{body} \\ & \text{and } rate[h_k(i, j), v_k(i, j)]_{k=1,2} > R_{body} \\ & \text{and } rate[h_k(i, j), gr_{hk}(i, j)]_{k=1,2} > R_{grad} \\ & \text{and } rate[v_k(i, j), gr_{vk}(i, j)]_{k=1,2} > R_{grad} \\ & \text{and } pos_x(i, j), pos_y(i, j) > size(i, j) \cdot P_{body} \\ 0 & \text{otherwise} \end{cases} \quad (3.37)$$

つづいて、以下の式に従い、ピクトグラム領域内左上の券売機部分の判定を行う。

$$ticket(i, j) = \begin{cases} 1 & \text{if } \min_{k=1,2} [h_k(i, j), v_k(i, j)] > size(i, j) \cdot S_{ticket} \\ & \text{and } 1.5 \cdot h_1(i, j) < v_1(i, j) \\ & \text{and } 1.5 \cdot h_2(i, j) < v_2(i, j) \\ & \text{and } rate[h_k(i, j), gr_{hk}(i, j)]_{k=1,2} < R_{ticket} \\ & \text{and } rate[v_k(i, j), gr_{vk}(i, j)]_{k=1,2} < R_{ticket} \\ & \text{and } pos_x(i, j) < size(i, j) \cdot P_{ticket}^x \\ & \text{and } pos_y(i, j) > size(i, j) \cdot P_{ticket}^y \\ 0 & \text{otherwise} \end{cases} \quad (3.38)$$

その後、3.2.8 節と同様の手法を用い、 $body(i, j)$, $ticket(i, j)$ をピクトグラム領域内に含むものを切符売場マーク *Booth* とする。

エレベーターマーク検出

エレベーターマークは、上部にある上向き矢印と下向き矢印に着目し、検出する。矢印の検出するため、3.2.5 節と同様に画素集積法を用いる。まず、上向き矢印の判定を行うため、画素値の連続数を下から上 ($d2u$)、左下から右上 ($ld2ru$)、右下から左上 ($rd2lu$) へと集積する。二値画像 *Bin* を用い、以下の式に従い初期化する。

$$d2u^0(i, j) = ld2ru^0(i, j) = rd2lu^0(i, j) = bin(i, j) \quad (3.39)$$

次に、以下の式を繰り返し、それぞれの方向に連続数を集積する。画素値の連続数集積には、8 近傍画素を参照するため、PE 間通信が必要となる場合がある。

$$d2u^{t+1}(i, j) = \begin{cases} d2u^t(i, j) + 1 & \text{if } d2u^t(i, j+1) = d2u^t(i, j) \\ & \text{and } d2u^t(i, j) > 0 \\ d2u^t(i, j) & \text{otherwise} \end{cases} \quad (3.40)$$

$$ld2ru^{t+1}(i, j) = \begin{cases} ld2ru^t(i, j) + 1 & \text{if } ld2ru^t(i-1, j+1) = ld2ru^t(i, j) \\ & \text{and } ld2ru^t(i, j) > 0 \\ ld2ru^t(i, j) & \text{otherwise} \end{cases} \quad (3.41)$$

$$rd2lu^{t+1}(i, j) = \begin{cases} rd2lu^t(i, j) + 1 & \text{if } rd2lu^t(i+1, j+1) = rd2lu^t(i, j) \\ & \text{and } rd2lu^t(i, j) > 0 \\ rd2lu^t(i, j) & \text{otherwise} \end{cases} \quad (3.42)$$

式 (3.39) ~ 式 (3.42) で求めた $d2u(i, j), ld2ru(i, j), rd2lu(i, j)$ をもとに、以下の式に従い、矢印かどうかの判断を行う。矢印の判定は、注目画素内で行うため、PE 間通信の必要はない。

$$arrow_u(i, j) = \begin{cases} 1 & \text{if } ld2ru(i, j), rd2lu(i, j) > size(i, j) \cdot S_{ar}^{short} \\ & \text{and } d2u(i, j) < size(i, j) \cdot S_{ar}^{long} \\ & \text{and } pos_x(i, j) < size(i, j) \cdot P_{elv}^x \\ & \text{and } pos_y(i, j) < size(i, j) \cdot P_{elv}^y \\ 0 & \text{otherwise} \end{cases} \quad (3.43)$$

式 (3.43) では、集積された連続数の比や位置により、矢印 $Arrow_u$ を求めている。エレベーターマークの上向き矢印はピクトグラムの左上にあるのでピクトグラム内の座標 $pos_x(i, j), pos_y(i, j)$ を利用し、位置による判定を行っている。ピクトグラム内の左上部分に上向き矢印があれば、 $arrow_u(i, j)$ が画素値を持つことになる。

下向きの矢印 $Arrow_d$ についても、式 (3.39) ~ 式 (3.43) と同様の手順で $u2d(i, j)$ 、 $lu2rd(i, j)$ 、 $ru2ld(i, j)$ を求め、以下の式に従い、比や位置による判定を行う。

$$arrow_d(i, j) = \begin{cases} 1 & \text{if } lu2rd(i, j), ru2ld(i, j) > size(i, j) \cdot S_{ar}^{short} \\ & \text{and } u2d(i, j) < size(i, j) \cdot S_{ar}^{long} \\ & \text{and } pos_x(i, j) > size(i, j) \cdot P_{elv}^x \\ & \text{and } pos_y(i, j) < size(i, j) \cdot P_{elv}^y \\ 0 & \text{otherwise} \end{cases} \quad (3.44)$$

式 (3.28) と同様の手法を用い、 $arrow_u(i, j)$ 、 $arrow_d(i, j)$ を含むピクトグラムをエレベーターマークとして検出する。

電車マーク検出

電車マーク $Train$ は、ピクトグラムの電車部分がほぼ四角形であることに着目し、検出を行う。ピクトグラム領域内において、再び 3.2.5 節で行った水平方向および垂直方向の画素集積法を適用する。画素値の連続数の計算後、式 (3.20) の代わりに以下の式を用いて電車マークの判定を行う。

$$train^0(i, j) = \begin{cases} \min_{k=1,2} [h_k(i, j), v_k(i, j)] \\ \quad \text{if } \min_{k=1,2} [h_k(i, j), v_k(i, j)] > size(i, j) \cdot S_{train} \\ \quad \text{and } rate[h_k(i, j), v_k(i, j)]_{k=1,2} > R_{train} \\ \quad \text{and } rate[h_k(i, j), gr_{hk}(i, j)]_{k=1,2} > R_{train} \\ \quad \text{and } rate[v_k(i, j), gr_{vk}(i, j)]_{k=1,2} > R_{train} \\ 0 & \text{otherwise} \end{cases} \quad (3.45)$$

判定後、式 (3.28) と同様の手法で、ピクトグラム領域内の $train^0(i, j)$ に値を持つ領域を $Train$ として検出する。

上りエスカレーターマーク検出

上りエスカレーターのマークはピクトグラム領域右下にある右上向きの矢印に着目し、検出する。3.2.8 節のエレベーターマーク検出で用いた式 (3.39) ~ 式 (3.42) と同様の手順で下から上、左から右、左下から右上へと画素値の連続数を集積し、それぞれ $d2u(i, j)$, $l2r(i, j)$, $ld2ru(i, j)$ とし、式 (3.43) と同様に以下の式で判定を行い、 $Arrow_{ru}$ とする。画素の連続数集積には、隣の PE との通信が必要となるが、式 (3.46) で判定を行う際

には、通信は必要ない。

$$arrow_{ru}(i, j) = \begin{cases} 1 & \text{if } d2u(i, j), rd2lu(i, j) > size(i, j) \cdot S_{ar}^{short} \\ & \text{and } ld2ru(i, j) < size(i, j) \cdot S_{ar}^{long} \\ & \text{and } pos_x(i, j) > size(i, j) \cdot P_{up}^x \\ & \text{and } pos_y(i, j) > size(i, j) \cdot P_{up}^y \\ 0 & \text{otherwise} \end{cases} \quad (3.46)$$

そして、3.2.8 節のトイレマーク抽出と同様の手法を用い、 $arrow_{ru}(i, j)$ を含むピクトグラムを上りエスカレーターマークとして検出する。

下りエスカレーターマーク検出

下りエスカレーターのマークはピクトグラム領域右下にある左下向きの矢印に着目し、検出する。上りエスカレーターマーク検出と同様の手順で $Arrow_{ld}$ を求める。この際、式 (3.46) において、 P_{up}^x, P_{up}^y の代わりに P_{down}^x, P_{down}^y をそれぞれ用いる。そして、3.2.8 節のトイレマーク抽出と同様の手法を用い、 $arrow_{ld}(i, j)$ を含むピクトグラムを下りエスカレーターマークとして検出する。

以上が本章で提案するピクトグラム検出アルゴリズムである。

3.3 計算機シミュレーション

提案アルゴリズムの有効性を示すため、情景画像を用いて計算機シミュレーションを行った。行った計算機シミュレーションは、ピクトグラム抽出、計算量・通信量の測定、東急線駅構内画像のピクトグラム検出の3種類である。

3.3.1 ピクトグラムの抽出

3.2.5 節で述べた四角形抽出アルゴリズムを用い、情景画像からピクトグラムを抽出する計算機シミュレーションを行い、筆者らの従来法 [76] と比較した。

シミュレーション条件

通常のデジタルカメラで撮影した、縦 480 画素 × 横 640 画素、256 階調のカラー画像を使用した。画像は屋内外で撮影したものである。撮影時の天候は晴れまたは曇であり、日中あるいは夜間に撮影した画像である。情景画像 100 枚の内、20 枚を表 3.1 に示すパラメータ設定のために使用し、残りの 80 枚を用いて評価した。評価指標として、抽出率と誤抽出率を次式のように定義した。

$$\text{抽出率 (\%)} = \frac{\text{抽出に成功した領域数}}{\text{画像に含まれるピクトグラム数}} \times 100 \quad (3.47)$$

$$\text{誤抽出率 (\%)} = \frac{\text{抽出した非ピクトグラムの領域数}}{\text{抽出した領域数}} \times 100 \quad (3.48)$$

480 × 640 画素の入力画像に対し、大きさが 40 × 40 画素以下のピクトグラムは抽出対象とはしていない。

計算機シミュレーションの条件を表 3.1 に示す。パラメータ値は予備実験により、最適と思われる値を設定した。

シミュレーション結果

表 3.2 に示すように計算機シミュレーションの結果、抽出率は 95.0%、誤抽出率は 6.0% であった。入出力画像の例を図 3.13 に示す。図 3.13 では、それぞれ左側の入力画像に対し、ピクトグラムとして抽出された領域が右側の図に示されている。

提案法では従来法に比べ誤検出率が大幅に下がっていることがわかる。これは、3.2.5 節で示したように、ピクトグラムを検出する際、傾きの計算を行っているためであると考えられる。傾きの大きさを四角形抽出の判断に利用することにより、誤検出が減

表 3.1 ピクトグラム抽出のパラメータ

Table 3.1 Parameters of pictogram extraction

Parameter name	Value
ϵ in equation(3.5)	20
Times of smoothing	7
q in color classification	120
f_1 in equation(3.10)	1500
f_2 in equation(3.11)	2000
f_3 in equation(3.12)	1500
f_4 in equation(3.13)	1500
R_{pict}	0.85
S_{pict}	40
Times of square adjustment	10

表 3.2 ピクトグラム抽出結果

Table 3.2 A result of pictogram detection

	Detection rate(%)	False detection rate(%)
Proposed	95.0	6.0
Convensional[76]	94.1	16.4

少しした。図 3.13 の一番目の画像 (Input 1) では、画像中の右側にあるエスカレーターマークや矢印の抽出に失敗している。これはエッジ検出過程においてエッジが切れてしまい、多色分類過程においてマークの枠と看板の外枠が連結してしまったためであると考えられる。図 3.13 の一番下の画像 (Input 4) が誤抽出の例である。

また、計算機シミュレーションにおいてピクトグラムの大きさ毎の抽出率を表 3.3 にまとめた。

表 3.3 において、Size(pixel on a side) はピクトグラムの一辺の大きさを表す。例えば、 $40 \times 40 \sim 70 \times 70$ のピクトグラムの抽出率が 96.0% である。抽出率はいずれの大きさにおいても 90% を超えていることから、提案アルゴリズムでは四角形領域抽出過程において S_{pict} によるしきい値処理で小さすぎる四角形領域を削除している以外では、

表 3.3 ピクトグラムの大きさ別抽出率

Table 3.3 Extraction rate of size

Size (pixel on a side)	40 ~ 60	60 ~ 80	80 ~ 100	100 ~ 120	120 ~
Extraction rate (%)	97.1	93.8	91.3	98.0	93.0

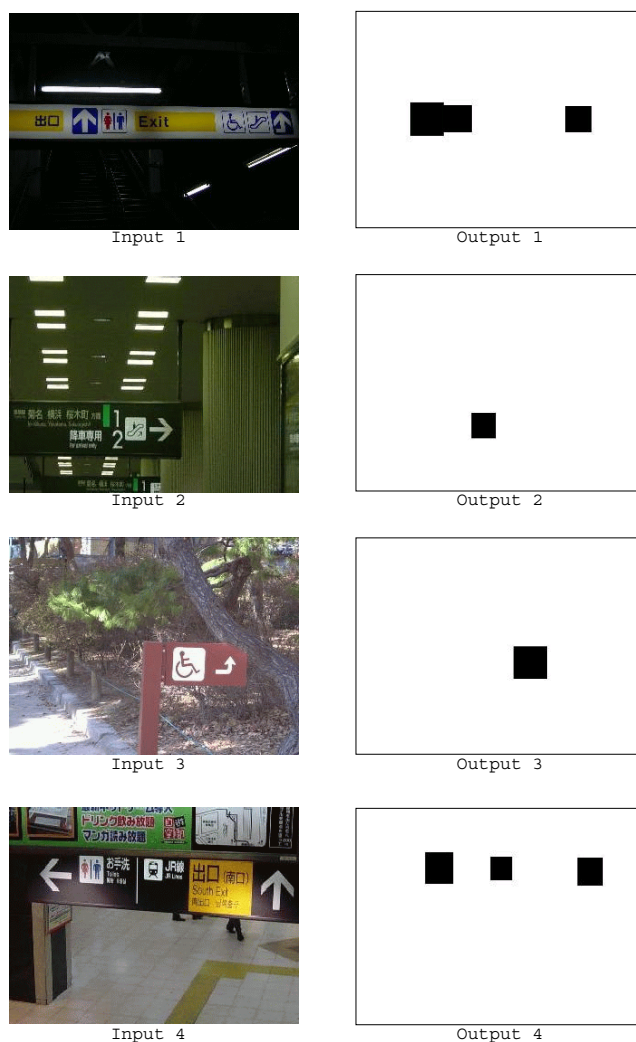


図 3.13 ピクトグラム入出力画像例

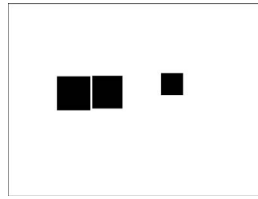
Fig. 3.13 Input and output examples of pictogram extraction.

抽出率はピクトグラムの大きさによらないと考えられる。

次ページよりピクトグラム抽出シミュレーションの入出力画像例を示す。



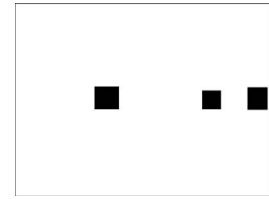
入力画像 1



出力画像 1



入力画像 2



出力画像 2



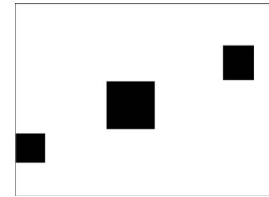
入力画像 3



出力画像 3



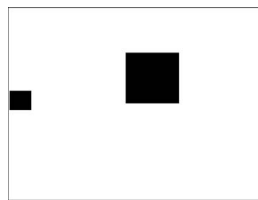
入力画像 4



出力画像 4



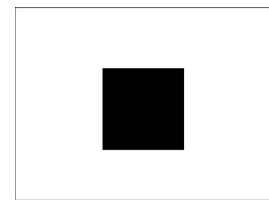
入力画像 5



出力画像 5



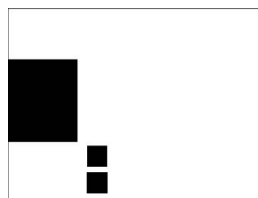
入力画像 6



出力画像 6



入力画像 7



出力画像 7



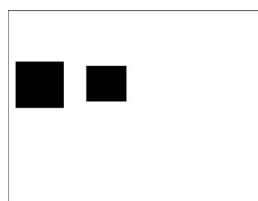
入力画像 8



出力画像 8



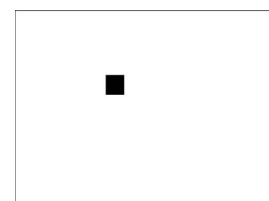
入力画像 9



出力画像 9



入力画像 10



出力画像 10

図 3.14 入出力画像 3-1



入力画像 11



出力画像 11



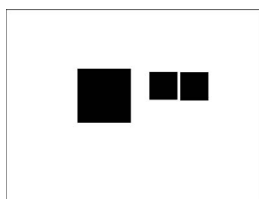
入力画像 12



出力画像 12



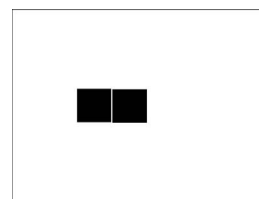
入力画像 13



出力画像 13



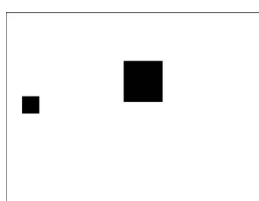
入力画像 14



出力画像 14



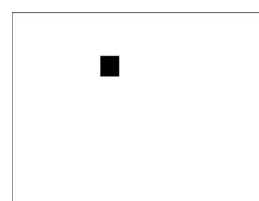
入力画像 15



出力画像 15



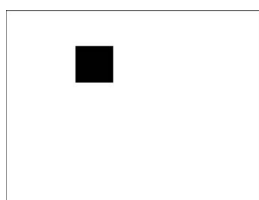
入力画像 16



出力画像 16



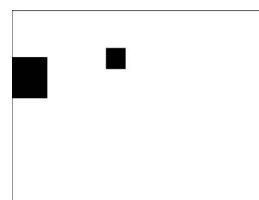
入力画像 17



出力画像 17



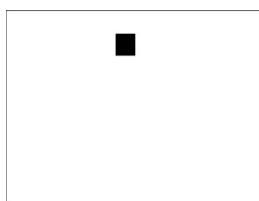
入力画像 18



出力画像 18



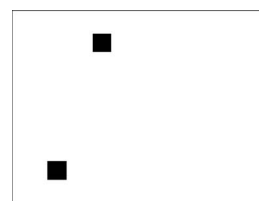
入力画像 19



出力画像 19



入力画像 20



出力画像 20

図 3.15 入出力画像 3-2

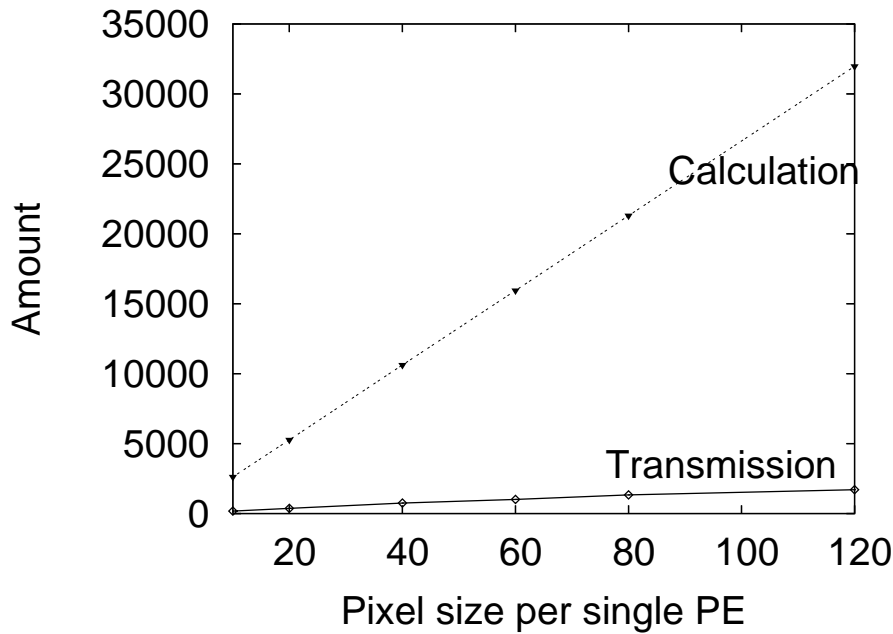


図 3.16 ピクトグラム抽出アルゴリズムにおける各 PE の計算量と通信量

Fig. 3.16 Amount of calculation and transmission.

3.3.2 PE の計算量および通信量の測定

2.3.4 節と同様に、ピクトグラム抽出アルゴリズムにおいても、情景画像 80 枚に対してアルゴリズムを実行した結果を図 3.16 に示す。

図 3.16 において、横軸は一つの PE 当たりが扱う画素数を表し、縦軸は各 PE の四則演算・比較命令の回数を表している。図 3.16 のグラフでは、PE が扱う画素数が増えるにつれて、ほぼ比例するように計算回数・通信回数ともに増加している。2.3.4 節では文字抽出アルゴリズムの実行に対して、計算量や通信量の測定を行ったが、ピクトグラム抽出アルゴリズムの実行に対しても同様の結果が得られた。計算回数が増加するのは各 PE が処理する画素数が増加するからである。通信回数が増加するのは各 PE が扱う画素が増加するにつれて、PE の端に当たる画素が増加するためである。PE の端に当たる画素は、処理のために近傍画素情報を得るため PE 間通信しなければならない。各 PE が扱う画素が増加すれば、各 PE が扱う画素において、処理に通信が必要な画素が増えるため通信回数も増加する。このことから、PE の数を増加させて PE が扱う画素を減少させることにより、画像サイズの増加に対応できると考えられる。また、図 3.16 より、通信回数は計算回数に比べると非常に少ないことがわかる。このことから、提案アルゴリズムは局所並列処理を行う“視覚装置”に向いていると考えられる。

表 3.4 ピクトグラム検出のパラメータ

Table 3.4 Parameters of pictogram detection

Parameter name	Value	Parameter name	Value
ϵ	30	Times of smoothing	10
S_{pict}	40	R_{pict}	0.6
R_{grad}	0.4	C_{Chair}	0.05
S_{box}	0.4	R_{box}^{min}	0.6
R_{box}^{max}	0.8	P_{key}^{xmin}	0.2
P_{key}^{xmax}	0.8	P_{key}^y	0.3
Th_{key}	0.03	S_{body}	0.2
R_{body}	0.7	P_{body}	0.6
S_{ticket}	0.1	R_{ticket}	0.6
P_{ticket}^x	0.5	P_{ticket}^y	0.3
S_{ar}^{short}	0.1	S_{ar}^{long}	0.3
P_{elv}^x	0.5	P_{elv}^y	0.3
S_{train}	0.4	R_{train}	0.7
P_{up}^x	0.6	P_{up}^y	0.5
P_{down}^x	0.5	P_{down}^y	0.6

3.3.3 東急線駅構内のピクトグラム検出

本章で提案するピクトグラム検出アルゴリズムを用い、駅構内の画像からピクトグラムを検出する計算機シミュレーションを行った。

シミュレーション条件

入力に用いたのはデジタルカメラで撮影した東急東横線駅構内のカラー画像である。カメラは RICOH Caplio R1 を用い、サイズは縦 480 画素 × 横 640 画素である。250 枚の画像の内、50 枚をパラメータ設定のために使用し、残りの 200 枚を用いて評価した。シミュレーションに用いたパラメータを表 3.4 に示す。480 × 640 画素の入力画像に対し、大きさが 40 × 40 画素以下のピクトグラムは検出対象とはしていない。評価指標として、検出率と誤検出率を定義した。検出率は画像中のピクトグラムのうち、検出に成功したピクトグラムの割合である。誤検出率は各手法で検出した領域のうち、ピクトグラムでない領域の割合である。

シミュレーション結果

ピクトグラム検出シミュレーションの結果を表 3.5 に示す。

表 3.5 ピクトグラム分類結果

Table 3.5 A result of pictogram classification

Pictogram	Detection rate(%)	Pictogram	Detection rate(%)
Toilet	84.0	Ticket booth	91.0
Baby	98.5	Elevator	60.0
Wheel chair	94.0	Train	62.0
Locker	96.0	Escalator()	92.0
Man	94.4	Escalator()	84.0
Woman	89.0	Total	85.8

全 11 種類のピクトグラム検出率は 85.8%であった。エレベーターマークと電車マーク以外は 80%以上の検出率を得ることができた。エレベーターマークと電車マークの検出率が低くなった大きな原因は、マークの形状が四角形である、もしくは四角形に近いからである。そのためコインロッカーとして検出されてしまう。したがって、この問題を解決すれば、さらに精度を向上させることができると考えられる。

図 3.17 にシミュレーション結果の例を示す。

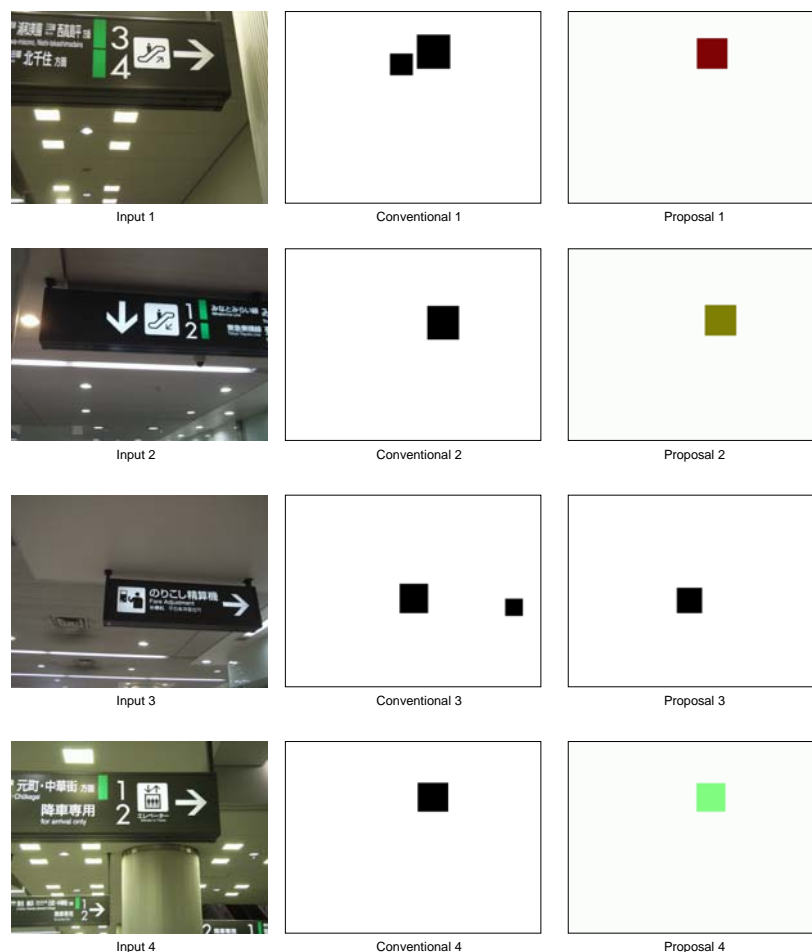


図 3.17 ピクトグラム検出シミュレーション結果

Fig. 3.17 An example of simulations.

3.4 考察

本章のピクトグラム検出アルゴリズムも、第2章で述べた文字抽出アルゴリズム同様に、ほとんどの処理が8近傍以下の処理となっている。そのため、2.4節で述べたように、局所並列処理に適していると考えられる。

また、ピクトグラム検出アルゴリズム内の処理にも、余計な計算コストをかけて並列性を求めている処理がある。例えば、3.2.5節の水平方向連続数の計算では、式(3.18)のように表現している。これを図を用いて表すと、図3.18のように、画素の連続数が求まっていく。

しかしながら、大域的な処理を行う場合と同様に処理すると、図3.19のように、短い計算ステップで画素の連続数を求めることが可能である。

提案アルゴリズムで、この手法をとらなかったのは、並列性を維持するためである。例えば、図3.20のように、PEをまたがった線分が複数存在した場合、図3.19のよう

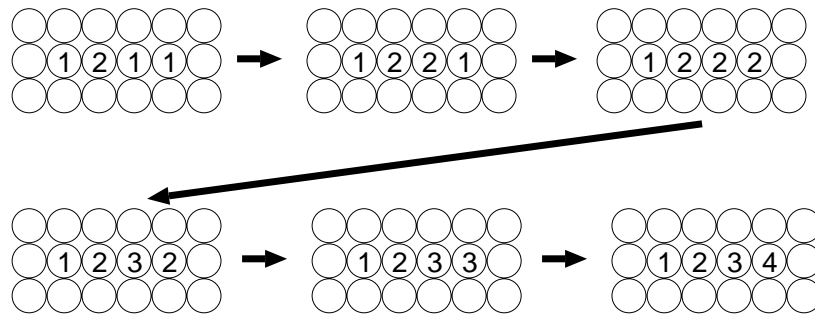


図 3.18 提案アルゴリズムの水平方向画素計算

Fig. 3.18 Horizontal calculation of the proposed algorithm.

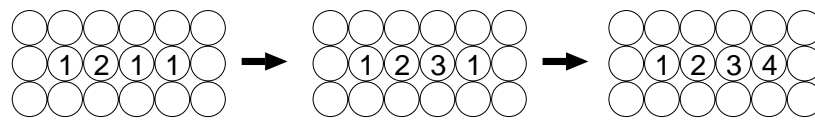


図 3.19 水平方向画素計算例

Fig. 3.19 An example of horizontal calculation.

な処理をしてしまうと、PE A, PE B, PE C の順に処理をしなければならない。そのため、線 A の連続数を計算した後に線 B の連続数を計算し、最後に線 C の連続数を求めることになる。これでは、PE に分割して並列処理を行うという本研究の目的から外れてしまう。そのため、式 (3.18) のように、画素単位に連続数を増加していくことにより、全ての計算を同時に開始することができる構成とした。

ただし、これは画素の連続数の計算という、本来並列性の低い処理を画素単位の処理へと落とし込んだ表現であり、ハードウェアに実装する際には、チューニングして、よりコストの低い実装方法があることも考えられる。

また、PE のサイズについては、福田らの研究 [67] から、単純なフィルタであれば、33MHz 動作時で、 30×30 の画像を毎秒 150 フレーム程度処理できること、多色分類は毎秒 15 ~ 30 フレーム処理するには $8 \times 8 \sim 12 \times 12$ 画素程度が好ましいという結果が出ている。

本研究で対象とする視覚障害者支援のためには、ロボットの視覚のように毎秒 30 フレームというような処理速度は必要なく、毎秒 1 フレーム程度でも実用に耐え得ると考えられる。よって、アルゴリズムで参照する最大近傍サイズが 15×15 であることから、PE のサイズも 15×15 画素程度が適当であると考えられる。また、PE のサイズとピクトグラムのサイズの関係には、検出率には影響がない。

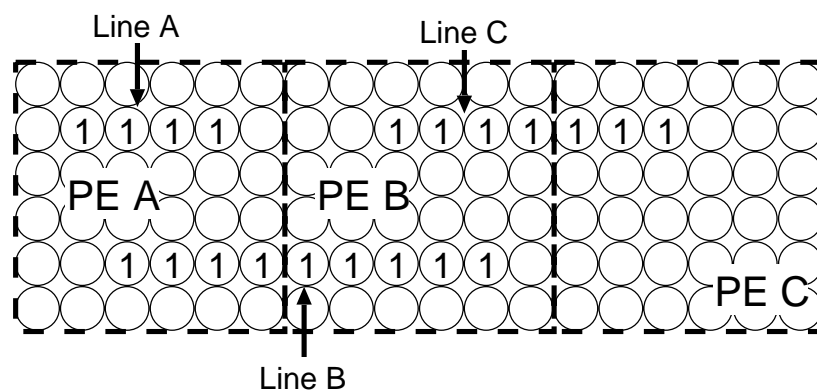


図 3.20 不適切な例

Fig. 3.20 An example of improper situations.

3.5 むすび

本章では局所的な処理のみで構成されたピクトグラム検出アルゴリズムを提案した。

提案アルゴリズムは、並列処理を考慮し、すべての処理が局所計算のみで完結している。四角形抽出アルゴリズムをピクトグラム検出に応用したものであり、ピクトグラムの視認性を利用し、単純な計算で構成された簡易なアルゴリズムである。

計算機シミュレーションにより、従来に比べ誤抽出を少なくすることができることを確認した。

さらに、特定のピクトグラムに特化した情報を使うことにより、単純な計算の繰り返しで 11 種類のピクトグラムを検出できることが確認されたが、検出対象が持つ特徴量と、検出に利用する特徴量を適切に設定しないと、検出率が下がってしまうことも確認した。

単純な計算のみで構成されたアルゴリズムであっても、検出対象の特徴を適切に求めることで、他の対象を検出するアルゴリズムへの応用が容易であることを示した。

第 4 章

結論

本論文では、局所並列処理のための画像処理アルゴリズムとして、文字抽出アルゴリズムとピクトグラム検出アルゴリズムの二種類の局所並列画像処理アルゴリズムを提案した。

本論文で提案したアルゴリズムは、局所的な処理を並列に行うハードウェアへの実装のために開発されたものである。対象とするハードウェアアーキテクチャは通常の PC とは異なり、低コストで実用的な結果を得ることを目的に設計されている。そのため、小規模な PE を多数配置し、個々の PE が独立に動作することにより画像を並列に処理するものである。提案アルゴリズムはこのようなハードウェアアーキテクチャの設計概念に沿った局所並列画像処理のみで構成されている。

情景画像からの文字抽出アルゴリズム提案では、輝度に対するしきい値を設定することなく二値化を行うことが可能である。計算機シミュレーションを行い、照明条件が悪い画像からも文字を抽出することができることが確認できた。もちろんアルゴリズムは全て近傍処理から構成されており、視覚装置ハードウェアへの実装が可能である。また、第 2 章で提案したアルゴリズムのうち、一部のアルゴリズムは慶應義塾大学天野研究室の福田らにより FPGA に実装され、消費電力や処理速度の測定が行われ、有効性が確認されている。本論文においても、計算量や通信回数の測定を行い、アルゴリズムが視覚装置に向いていることを確認した。

ピクトグラムとは、意味するものの形状を使って、その意味概念を理解させる記号である。本論文では、ピクトグラムの外枠の正方形に着目し、情景画像から近傍処理のみにより四角形を抽出するという手順で情景画像からピクトグラムを検出するアルゴリズムを構成した。また、東急線駅構内のピクトグラムに特化して検出するアルゴリズムについて述べた。四角形に着目したピクトグラム抽出アルゴリズムに、特定のピクトグラム固有の特徴を抽出する処理を加えることにより、ピクトグラムの種類を絞って抽出できることを示した。計算機シミュレーションを行い、11 種類のピクトグラム検出アルゴリズムの有効性を確認した。しかしながら、類似した特徴量を持つ検出対象に対して、検出に利用する特徴量が類似している場合等には、検出率が上から

ないケースもあり、検出に利用する特徴量の設定が重要となることも確認した。より多くの種類の特徴量を、局所計算のみで求める手法を考案することも、今後の課題となる。

近年、カメラ付き携帯電話が普及し、モバイル端末による画像処理が一般的になっている。また、ITS やロボットビジョンの需要からも、組み込みハードウェアによる画像処理は今後もさらに研究・開発が進むことが予想される。しかしながら、より多くの利用者への普及には、パフォーマンスとコストのトレードオフの関係は避けられない。パフォーマンスを追求しなければならない分野・用途もあるが、本論文で扱った、味岡らが提案する視覚装置は世の中への普及に有効なアプローチの一つであると考えられる。本論文では、開発中の視覚装置を対象とした画像処理アルゴリズムを提案し、その有効性を確認した。これはハードウェア開発にとっても、開発したハードウェアで動作するものが存在するという点で重要なことであると考えられる。高性能なハードウェアが完成したとしても、その上で動作するアプリケーションがなければ、そのハードウェアの良さを活かすことができないからである。

謝辞

萩原将文先生に。本研究を進めるにあたり常に適切なアドバイスを下さっただけでなく、大学生活および社会人生活全般について親身になって相談に乗って下さったことに心から感謝致します。先生の今後ますますのご活躍を祈願致します。

査読をして頂いた先生方、小沢慎治先生、浜田望先生、天野英晴先生に。論文を完成させるにあたり、先生方には様々な角度からの多数の貴重な御指摘を頂きました。小沢先生には、度重なるディスカッションで、多くのアドバイスを頂きました。浜田先生には、論文に対する有益なコメントを頂きました。天野先生には、ハードウェアの観点から、非常に大きなアドバイスをして頂きました。心より御礼申し上げます。

株式会社エッチャンデス代表取締役 味岡義明社長に。大変興味深い研究テーマを提供して下さい、アドバイスをして頂いただけでなく、共同研究を通して様々な経験をさせて頂きました。心より御礼申し上げます。

慶應義塾大学天野研究室の福田静人君に。研究を進めるにあたり、多くのアドバイスを頂き、様々な問題点を解決することができました。どうもありがとうございました。

萩原研究室の皆様。皆様にディスカッション・発表練習・論文添削といったことで助けて頂いただけでなく、暖かい雰囲気の中で研究をさせて頂いたことがこの成果につながりました。どうもありがとうございました。

父と母に。今の私があるのも二人のお蔭です。私を生み育ててくれたことに感謝を捧げます。

参考文献

- [1] 味岡義明, “視覚装置,” PCT 国際公開番号 WO00/16259.
- [2] 村上伸一, “画像処理工学”, 東京電機大学出版局, 1996.
- [3] 安居院猛, 長尾智晴著, “画像の処理と認識”, 昭晃堂, 1992.
- [4] 長尾真, “画像認識論”, コロナ社, 1983.
- [5] 本多庸悟他著, “画像処理と視覚認識”, オーム社, 1995.
- [6] 下田陽久他著, “画像処理標準テキストブック”, 財団法人 画像情報教育振興協会, 1997.
- [7] 苗村健, “バーチャルリアリティ技術の最新動向”, 電気学会誌, Vol.124, No.6, pp.353-356, 2004.
- [8] 内田薫, “バイオメトリクスによる個人認証技術とその応用”, 電気学会誌, Vol.120, No.7, pp.407-410, 2000.
- [9] 松元勉, “電子透かしと関連技術”, 電気学会誌, Vol.120, No.7, pp.430-433, 2000.
- [10] 鷲見和彦, “ITS 分野への応用”, 電気学会誌, Vol.120, No.7, pp.419-421, 2000.
- [11] 中村裕一, “映像データの知的生成・編集・検索”, 電気学会誌, Vol.120, No.7, pp.415-418, 2000.
- [12] J.W.Cooley and J.W.Tukey, “An algorithm for the machine calculation of complex fourier series,” Mathematics of Computation, Vol.19, pp.297-301, 1965.
- [13] 松尾賢一, 上田勝彦, 梅田三千雄, “局所対象領域の2値化による情景画像からの文字列領域抽出”, 電学論 (C), Vol.122-C, No.2, pp.232-241, 2002.
- [14] Yihong Xu, George Nagy, “Prototype Extraction and Adaptive OCR,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.21, No.12, pp.1280-1296, 1999.

- [15] Huiping Li, Daved Doermann, and Omid Kia, "Automatic Text Detection and Tracking in Digital Video," IEEE Trans. Image Processing, Vol.9, No.1, pp.147-156, 2000.
- [16] Xiangyun Ye, Mohamaed Cheriet, and Ching Y.Suen, "Stroke-Model-Based Character Extraction from Gray-Level Document Images," IEEE Trans. Image Processing, Vol.10, No.8, pp.1152-1161, 2001.
- [17] Kwok-Wai Cheung, Dit-Yan Yeung, and Roland T.Chin, "Bidirectional Deformable Matching with Application to Handwritten Character Extraction," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.24, No.8, pp.1133-1139, 2002.
- [18] 清田公保, 櫻井敏彦, 山本眞司, "ストローク代表点の相対位置情報に基づく視覚障害者用オンライン文字認識", 信学論 (D-II), Vol.J80-D-II, No.3, pp.715-723, 1997.
- [19] Zhaoyang Lu, "Detection of Text Regions From Digital Engineering Drawings," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.20, No.4, pp.431-439, 1998.
- [20] Minako Sawaki, Norihiro Hagita, "Text-Line Extraction and Character Recognition of Document Headlines With Graphical Designs Using Complementary Similarity Measure," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.20, No.10, pp.1103-1109, 1998.
- [21] Liang-Hua Chen, Hong-Yuan Liao, Jiing-Yuh Wang, Kuo-Chin Fan, "Automatic Data Capture for Geographic Information Systems," IEEE Trans. Systems, Man, and Cybernetics Part C, Vol.29, No.2, pp.205-215, 1999.
- [22] Hau YANG, Shinji OZAWA, "Extraction of Text Region From Color Image Sequence of A Book Cover Containing Gilt Characters," 電学論 (C), Vol.122-C, No.4, pp.630-637, 2002.
- [23] 平岩裕康, 竹内義則, 松本哲也, 工藤博章, 劉詠梅, 大西昇, "視覚障害者のための環境内の文字情報理解支援システム", 信学技報, PRMU2001-269, pp.195-202, 2002.
- [24] 劉詠梅, 山村毅, 大西昇, 杉江昇, "シーン内の文字列領域の抽出について", 信学論 (D-II), Vol.J81-D-II, No.4, pp.641-650, 1998.

- [25] 高橋裕樹, 葛西浩一, 中嶋正之, “複合特徴を用いた情景画像からの文字領域抽出手法”, 信学技報, HCS2000-30, pp.47-54, 2000.
- [26] 傅杭, 長井隆行, 金子正秀, 樽松明, “屋内外の情景画像を対象とした看板文字の自動抽出”, 信学技報, IE2000-157, pp.13-18, 2001.
- [27] 堀修, 三田雄志, “テロップ認識のための映像からのロバストな文字部抽出法”, 信学論 (D-II), Vol.J84-D-II, No.8, pp.1800-1808, 2001.
- [28] 長井隆行, 影広達彦, 金子正秀, 樽松明, “情景画像中の文字及び看板領域の抽出”, 信学技報, CS2000-145, pp.103-108, 2001.
- [29] 金大祐, 高橋裕樹, 中嶋正之, “情景画像からのハングル文字列領域抽出と字素識別”, 信学技報, PRMU2002-18, pp.65-70, 2002.
- [30] Rainer Lienhart, Axel Wernicke, “Localizing and Segmenting Text in Images and Videos,” IEEE Trans. Circuits and Systems for Video Technology, Vol.12, No.4, pp.256-268, 2002.
- [31] 周景龍, 渡部広一, 河岡司, “情景画像からの看板領域と看板中の情報抽出 -安定背景領域の推定-”, 信学技報, PRMU2002-217, pp.37-42, 2003.
- [32] 梅崎智, 高橋貞夫, “雑誌表紙からの文字切り出し”, 信学技報, NC2002-154, pp.125-130, 2003.
- [33] 馬場洋一郎, 廣瀬明, “スペクトル勾配法:一般的な画像からの文字領域の抽出方法”, 信学技報, PRMU2003-88, pp.79-84, 2003.
- [34] 梅田三千雄, 橋本智広, “背景領域の細線化に基づく古文書の文字切り出しと認識”, 情報処理学会論文誌, Vol.45, No.4, pp.1188-1197, 2004.
- [35] 鈴木悠司, 平岩裕康, 竹内義則, 松本哲也, 工藤博章, 大西昇, 山村毅, “視覚障害者のための環境内の文字情報抽出システム”, 電学論 (C), Vol.124, No.6, pp.1280-1286, 2004.
- [36] 下村正夫, 佐藤俊治, 三宅章吾, 阿曾弘具, “新しいネオコグニトロン型ネットワークとICA・PCAを用いた学習法”, 信学論 (D-II), Vol.J88-D-II, No.4, pp.769-777, 2005.
- [37] 山口拓真, 丸山稔, “階層型識別器を用いた情景画像からの文字抽出手法”, 信学論 (D-II), Vol.J88-D-II, No.6, pp.1047-1055, 2005.

- [38] 小泉寿男, “ITS と情報処理技術”, 情報処理学会誌, Vol.40, No.10, pp.978-985, 1999.
- [39] 青木正喜, “ITS における画像認識の現状と課題”, 人工知能学会誌, Vol.15, No.2, pp.234-241, 2000.
- [40] 小橋雄一郎, 中島真人, “道路標識の自動認識 -夜間における道路標識認識-”, 信学技報, IE2003-244, Vol.103, No.644, pp.57-62, 2004.
- [41] 長谷博行, 米田政明, 酒井充, 丸山博, “カラー文書画像中の文字領域抽出を目的とした色分割についての検討”, 信学論 (D-II), Vol.J83-D-II, No.5, pp.1294-1304, 2000.
- [42] 傅杭, 長井隆行, 金子正秀, 樽松明, “情景画像からの看板領域および看板文字の自動抽出”, 映像メディア学会誌, Vol.57, No.7, pp.819-828, 2003.
- [43] 村瀬洋, V.V.Vinod, “局所色情報を用いた高速物体探索 -アクティブ探索法-”, 信学論 (D-II), Vol.J81-D-II, No.9, pp.2035-2042, 1998.
- [44] 志久修, 中村彰, 黒田英夫, “局所のおよび大域的な線分密度を用いた文字列抽出”, 情報処理学会論文誌, Vol.40, No.8, pp.3230-3237, 1999.
- [45] 伊東正安, 椿井正義, 野村明良, “局所可変構造要素による適応モフォロジー演算と超音波画像における領域抽出への応用”, 信学論 (D-II), Vol.J84-D-II, No.9, pp.1990-2000, 2001.
- [46] 天野英晴, “並列コンピュータ”, 昭晃堂, 1996.
- [47] 高橋義造編, “並列処理機構”, 丸善, 1989.
- [48] 藤田善弘, 山下信行, 木村亨, 中村和之, 岡崎信一郎, “メモリ集積型 SIMD プロセッサ IMAP”, 信学論 (D-I), Vol.J78-D-I, No.2, pp.82-90, 1995.
- [49] 許昭倫, 古賀拓也, 櫻井和之, 岡崎信一郎, “動画画像処理ボード IMAP-VISION による天候にロバストな先行車検出・追跡システム”, 信学技報, PRMU98-96, pp.29-36, 1998.
- [50] 櫻井和之, 許昭倫, 岡崎信一郎, “IMAP-VISION を用いた道路画像からの白線検出”, 信学技報, PRMU99-66, pp.17-22, 1999.

- [51] 京昭倫, “128 個の VLIW プロセッシングエレメントを集積した動画像認識プロセッサ IMAP-CE”, 情報処理学会研究報告, Vol.2002, No.8(ARC149), pp.37-42, 2002.
- [52] 宇井博貴, 有馬裕, 村尾文秀, 小守伸史, 久間和生, “感度自動調整機能を有する人工網膜 LSI”, 電学論 (E), Vol.120, No.5, pp.197-203, 2000.
- [53] 船津英一, 倉持宗一郎, 長渕洋二, 鹿毛裕史, 坂下徳美, 村尾文秀, 久間和生, “射影演算機能を持つ人工網膜 LSI”, 映像情報メディア学会誌, Vol.55, No.10, pp.1329-1336, 2001.
- [54] 生田剛一, 田村俊之, 田中健一, 久間和生, “人工網膜カメラネットワークを用いた駐車場監視システム”, 映像情報メディア学会技術報告, Vol.26, No.50, pp.15-18, 2002.
- [55] 石井抱, 石川正俊, “高速ビジョンのための直線抽出法”, 信学論 (D-II), Vol.J81-D-II, No.8, pp.1920-1926, 1998.
- [56] 鏡慎吾, 小室孝, 石井抱, 石川正俊, “実時間視覚処理のためのビジョンチップシステムの開発”, 信学論 (D-II), Vol.J84-D-II, No.6, pp.976-984, 2001.
- [57] 石川正俊, 小室孝, 鏡慎吾, “ディジタルビジョンチップの新展開”, 信学技報, IDC2002-39, pp.23-28, 2002.
- [58] 小室孝, 鏡慎吾, 石川正俊, “ビジョンチップのための動的再構成可能な SIMD プロセッサ”, 信学論 (D-II), Vol.J86-D-II, No.11, pp.1575-1585, 2003.
- [59] 鏡慎吾, 小室孝, 渡辺義浩, 石川正俊, “ビジョンチップを用いた実時間視覚処理システム VCS-IV”, 信学論 (D-I), Vol.J88-D-I, No.2, pp.134-142, 2005.
- [60] 鏡慎吾, 石川正俊, “通信遅延を考慮したセンサ選択手法”, 信学論 (A), Vol.J88-A, No.5, pp.577-587, 2005.
- [61] 重松智志, 森村浩季, 羽田野孝裕, 中西衛, 藤井孝治, 池田奈美子, 島村俊重, 町田克之, 岡崎幸夫, “新しい SoC 技術を用いた指紋認証チップ”, 第 19 回 回路とシステム軽井沢ワークショップ, pp.105-pp110, 2006.
- [62] 下ノ村和弘, 井上恵介, 亀田成司, 八木哲也, “シリコン網膜と F P G A を用いた動物体の追跡”, 映像情報メディア学会誌, Vol.58, No.1, pp.61-68, 2004.

- [63] Hiroyuki Kurino, Yoshihiro Nakagawa, Kang Wook Lee, Tomonori Nakamura, Yusuke Yamada, Ki-Tae Park and Mitsumasa Koyanagi, "Smart Vision Chip Fabricated Using Three Dimensional Integration Technology," Abst. of Neural Information Processing System, 2000 MIT press, July 2001.
- [64] 味岡義明, 嶋田雅洋, 天野英晴, "視覚装置における位置/大きさ検出アルゴリズムの検証", 信学技報, VLD2001-135, CPSY2001-94, 2002.
- [65] 嶋田雅洋, 味岡義明, 天野英晴, "視覚処理並列システムの予備評価", 信学技法, VLD2001-154, pp.59-66, 2002.
- [66] 福田静人, 味岡義明, 天野英晴, "位置/大きさ検出アルゴリズムのFPGAデバイスへの実装", FPGA/PLD03, 2003.
- [67] 福田静人, "局所並列画像処理アルゴリズム向けハードウェアの研究", 慶應義塾大学 修士論文, 2005.
- [68] 中村納, 岡本教佳, 庭田剛, 南敏, "欧文テキスト画像における文字領域の抽出アルゴリズム", 信学論 (D), Vol.66, No.4, pp.437-444, 1983.
- [69] 杉山吉, 木村隆, "2次元アレイプロセッサLSIの画像処理・文字認識への応用", 情報処理学会全国大会講演論文集, 33-2, pp.1637-1638, 1986.
- [70] H.Kawai, Y.Inoue, R.Streitenberger, M.Yoshimoto, "A highly-parallel DSP architecture for image recognition," Proc. IEEE Int. Conf. Acoust. Speech Signal Process, Vol.5, pp.3199-3202, 1995.
- [71] Rao P S, Srinivas C, Chand B H, Agarwal A, "Parallel Preprocessing Algorithms for Document Image Analysis," J. Inst. Electron Telecommun Eng, Vol.42, No.3, pp.165-173, 1996.
- [72] Chutatape O, Li L, Qian X, "Automatic license number extraction and its parallel implementation," Annu. Conf. IEEE Ind Electron Soc, Vol.25, No.2, pp.704-709, 1999.
- [73] 阿曾弘具, "文字認識と並列処理", 信学技報, PRMU2003 215-230, pp.95-99, 2004.
- [74] 岩片智, 味岡義明, 萩原将文, "局所並列処理による情景画像からの文字抽出アルゴリズム", 電学論 (C), Vol.124, No.4, pp.959-965, 2004.

- [75] 岩片智, 味岡義明, 萩原将文, “局所並列処理のための二値化アルゴリズム”, 電学論 (C), Vol.125, No.1, pp.7-13, 2005.
- [76] 岩片智, 藤田大介, 味岡義明, 萩原将文, “局所並列処理のためのピクトグラム抽出アルゴリズム”, 電学論 (C), Vol.126, No.2, pp.220-227, 2006.
- [77] 胡軼, 長尾智晴, 岡崎雅則, 茅根大嗣, “カメラ付携帯電話で撮影された画像のマッチング処理”, 映像情報メディア学会誌, Vol.57, No.9, pp.1177-1186, 2003.
- [78] 澤口元英, 山本和彦, 加藤邦人, “携帯電話における低解像度文字認識手法の提案”, 信学技報, PRMU2002-247, pp.25-30, 2003.
- [79] 鈴木悠司, 平岩裕康, 竹内義則, 松本哲也, 工藤博章, 大西昇, “視覚障害者のための環境内の文字情報抽出システム”, 信学技報, WIT2002-72, pp.13-18, 2003.
- [80] 時田洋輔, 横井博一, “超大規模ニューラルネットワークの Folthret による実現法”, 信学技報, NC2002-75, pp.55-60, 2002.
- [81] 肥川宏臣, “区分線形関数を活性化関数に持つパルスニューロン”, 信学技報, NC2002-75, pp.61-66, 2002.
- [82] 芳賀琢哉, 石田文彦, 金城光永, 佐藤茂雄, 中島康治, “量子化結合非単調ニューラルネットワークのハードウェア化設計とその閾値学習アルゴリズム”, 信学技報, NC2002-75, pp.67-72, 2002.
- [83] 平井有三, 落合辰男, 安永守利, “1000 ニューロン 100 万シナプスで構成されたニューラルネットワークハードウェアシステム”, 信学論 (D-II), Vol.J84-D-II, No.6, pp.1185-1193, 2001.
- [84] 安永守利, 高見知親, 吉原郁夫, “FPGA を用いたナノ秒オーダ画像認識ハードウェア”, 信学論 (D-II), Vol.J84-D-II, No.10, pp.2280-2292, 2001.
- [85] 中村清実, 木下智樹, 金山宏和, “物体の 2 次元的位置と形状を同時認識する 2 次元拡散連想型ニューラルネットワーク”, 信学論 (D-II), Vol.J84-D-II, No.10, pp.2258-2269, 2001.
- [86] 佐藤紀章, 萩原将文, “視覚システムに基づく動物体認識ニューラルネットワーク”, 信学論 (D-II), Vol.J85-D-II, No.4, pp.622-629, 2002.
- [87] 顧力栩, 田中直樹, 金子豊久, R.M.Haralick, “表紙画像からの文字領域抽出方式”, 信学論 (D-II), Vol.J80-D-II, No.10, pp.2696-2704, 1997.

- [88] 後藤英昭, 平山理継, 阿曾弘具, “局所多値しきい値処理による濃淡文書画像からの文字パターンの抽出”, 信学論 (D-II), Vol.J82-D-II, No.11, pp.2188-2192, 1999.
- [89] 原島博, 小田島薫, 鹿喰善明, 宮川洋, “ α -分離非線型デジタルフィルタとその応用”, 信学論 (A), Vol.J65-A, No.4, pp.297-304, 1982.
- [90] 岩片智, “近傍処理による情景画像からの文字抽出アルゴリズム,” 慶應義塾大学理工学部情報工学科 卒業論文, 2003.
- [91] N.Otsu, “A Threshold Selection Method from Gray-Scale Histograms,” IEEE Trans. on Systems, Man, and Cybernetics, Vol.SMC-9, No.1, pp.62-66, 1999.
- [92] 村越愛策, “図記号のおはなし”, 日本規格協会, 1987.
- [93] 太田幸夫, “ピクトグラムのおはなし”, 日本規格協会, 1995.
- [94] 松浦大祐, 山内仁, 高橋浩光, “特定色判別と領域限定を用いた円形道路標識の抽出”, 信学論, Vol.J85-D-II, No.6, pp.1075-1083, 2002.

付録 A

近傍処理による情景画像からの文字抽出アルゴリズム

本章では、2.3 節の計算機シミュレーションで従来法として用いた「近傍処理による情景画像からの文字抽出アルゴリズム」について述べる。

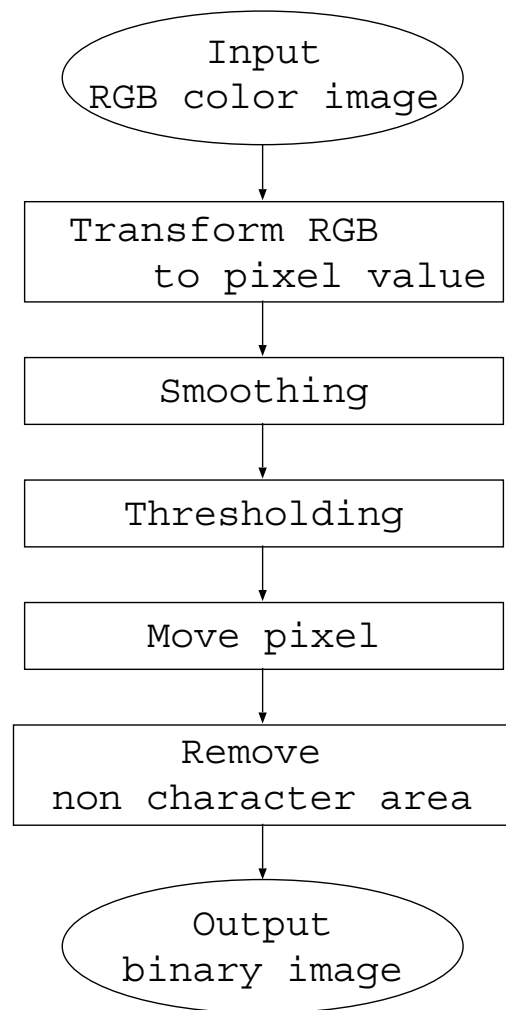


図 A.1 従来法の流れ

Fig. A.1 Flow of the conventional algorithm.

A.1 概要

図 A.1 にアルゴリズムの流れを示す。

入力カラー画像である。入力された RGB 値は提案アルゴリズムと同様に輝度値へ変換される。次に平滑化され、あらかじめ設定された閾値により二値化される。最後に画素値を各領域の中心の方向に移動させ領域の大まかなサイズを求め、適切なサイズの領域のみを文字とみなして復元させる。

A.2 詳細

従来法の詳細について述べる。輝度値変換、平滑化は提案法と同様であり、それぞれ式 (2.5)、式 (2.6) に従って行われる。

二値化は設定された閾値 Th を用い、以下の式に従って行われる。

$$x(i, j) = \begin{cases} 1 & \text{if } x(i, j) < Th \\ 0 & \text{else} \end{cases} \quad (\text{A.1})$$

最後の文字抽出過程は提案アルゴリズムとほぼ同様である。2.2.7 節で述べた手順で領域の重心の方向へ画素を移動させる。そして以下の式に従ってサイズ判別を行い、まず大き過ぎる領域を消去する。

$$y^{t+2}((l, m) \mid (l, m) \in P_{i,j}^8) = \begin{cases} 0 & \text{if } y^{t+1}(i, j) = 0 \text{ and } y^t(i, j) = 1 \\ y(l, m) & \text{else} \end{cases} \quad (\text{A.2})$$

$$y^1(i, j) = \begin{cases} 0 & \text{if } Th_{\text{large}} \leq w(i, j) \\ 1 & \text{else} \end{cases} \quad (\text{A.3})$$

$$y^0(i, j) = k(i, j) \quad (\text{A.4})$$

式 (A.4) では、元の二値画像を y^0 にコピーし、式 (A.3) では、重心において領域の画素数が大き過ぎるとみなされた画素の画素値を消去している。さらに式 (A.2) において、この処理で新たに画素値が 0 となった画素の 8 近傍を消去していく。この処理を繰り返すことにより、大き過ぎるサイズの領域を消去していく。

次に 2.2.7 節と同様の手順で適当なサイズの領域のみを復元し、文字領域として出力する。

以上が本論文で比較対象として用いた従来法である。

付録 B

文字抽出シミュレーションの入出力画像

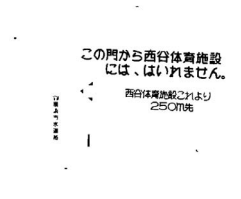
2.3.1 節で掲載しきれなかった入出力画像を掲載する。

B.1 情景画像からの文字抽出

2.3.1 節の、情景画像からの文字抽出シミュレーションの入出力画像を次ページより掲載する。



in11



out11



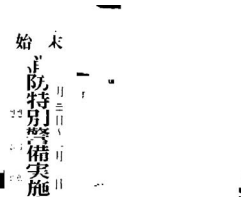
in12



out12



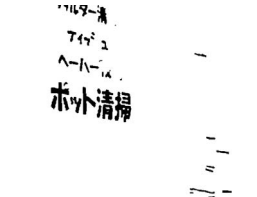
in13



out13



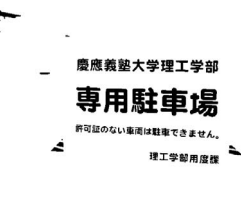
in14



out14



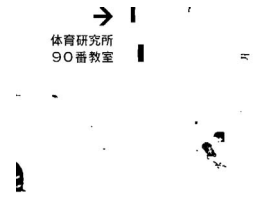
in15



out15



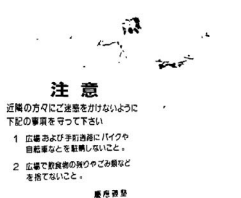
in16



out16



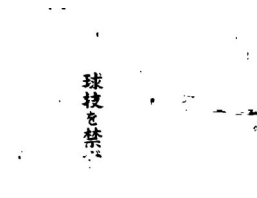
in17



out17



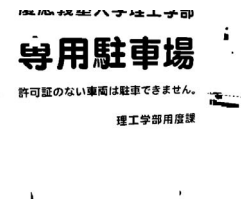
in18



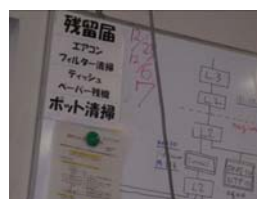
out18



in19



out19



in20



out20

図 B.1 入出力画像 1-2


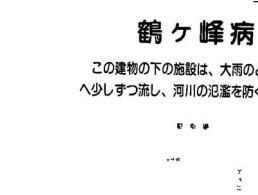

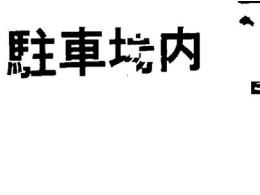

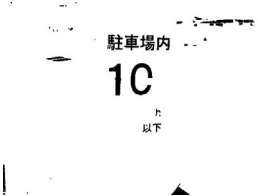

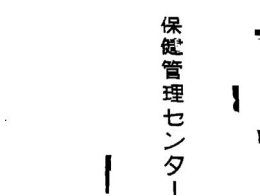

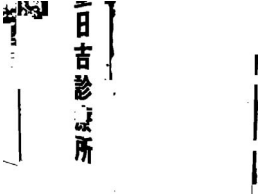



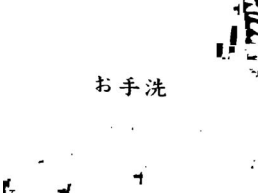

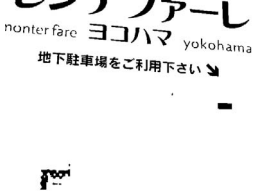

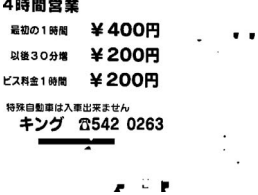

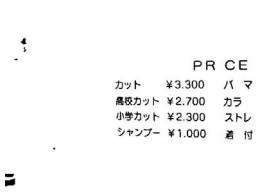
			
in21	out21	in22	out22
			
in23	out23	in24	out24
			
in25	out25	in26	out26
			
in27	out27	in28	out28
			
in29	out29	in30	out30

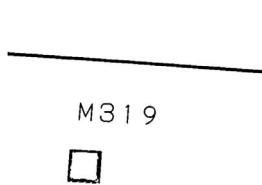
図 B.2 入出力画像 1-3

B.2 難易度の高い画像からの文字抽出

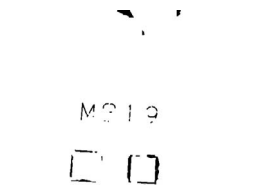
2.3.3 節の、難易度の高い画像からの文字抽出シミュレーションの入出力画像を次ページより掲載する。



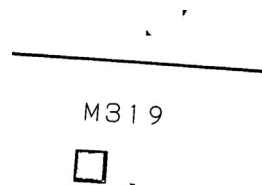
入力画像 06



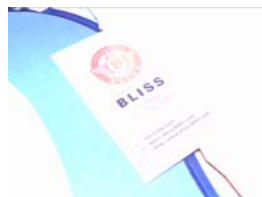
提案法



従来法



従来法+判別分析法



入力画像 07



提案法



従来法



従来法+判別分析法



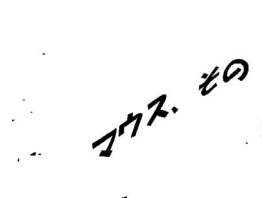
入力画像 08



提案法



従来法



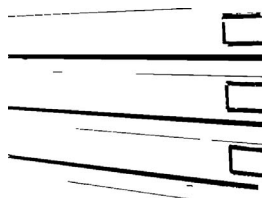
従来法+判別分析法



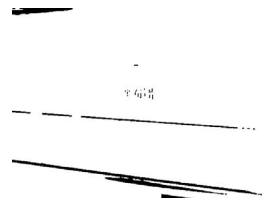
入力画像 09



提案法



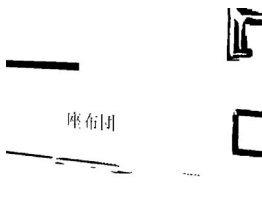
従来法



従来法+判別分析法



入力画像 10



提案法

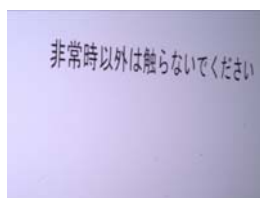


従来法



従来法+判別分析法

図 B.3 入出力画像 2-2



入力画像 11

非常時以外は触らないでください

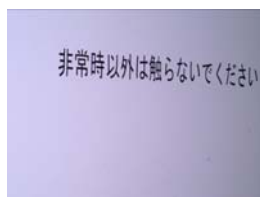
提案法

非常時以外は触らないでください

従来法

非常時以外は触らないでください

従来法+判別分析法



入力画像 12

非常時以外は触らないでください

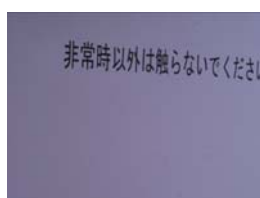
提案法

非常時以外は触らないでください

従来法

非常時以外は触らないでください

従来法+判別分析法



入力画像 13

非常時以外は触らないでください

提案法

非常時以外は触らないでください

従来法

非常時以外は触らないでください

従来法+判別分析法



入力画像 14

歩行禁煙

提案法

歩行禁煙

従来法

歩行禁煙

従来法+判別分析法



入力画像 15

歩行禁煙

提案法

歩行禁煙

従来法

歩行禁煙

従来法+判別分析法

図 B.4 入出力画像 2-3

付録 C

駅構内ピクトグラム検出シミュレーションの入出力画像

3.3 節で掲載しきれなかった入出力画像を次ページより掲載する。



入力画像 1



出力画像 1



入力画像 2



出力画像 2



入力画像 3



出力画像 3



入力画像 4



出力画像 4



入力画像 5



出力画像 5



入力画像 6



出力画像 6



入力画像 7



出力画像 7



入力画像 8



出力画像 8



入力画像 9



出力画像 9



入力画像 10



出力画像 10

図 C.1 入出力画像 5-1



入力画像 11



出力画像 11



入力画像 12



出力画像 12



入力画像 13



出力画像 13



入力画像 14



出力画像 14



入力画像 15



出力画像 15



入力画像 16



出力画像 16



入力画像 17



出力画像 17



入力画像 18



出力画像 18



入力画像 19



出力画像 19



入力画像 20



出力画像 20

図 C.2 入出力画像 5-2



入力画像 21



出力画像 21



入力画像 22



出力画像 22



入力画像 23



出力画像 23



入力画像 24



出力画像 24



入力画像 25



出力画像 25



入力画像 26



出力画像 26



入力画像 27



出力画像 27



入力画像 28



出力画像 28



入力画像 29



出力画像 29



入力画像 30



出力画像 30

図 C.3 入出力画像 5-3



入力画像 31



出力画像 31



入力画像 32



出力画像 32



入力画像 33



出力画像 33



入力画像 34



出力画像 34



入力画像 35



出力画像 35



入力画像 36



出力画像 36



入力画像 37



出力画像 37



入力画像 38



出力画像 38



入力画像 39



出力画像 39



入力画像 40



出力画像 40

図 C.4 入出力画像 5-4