平成 12 年度

学士学位論文

ニューラルネットワークを用いた
食物摂取量計測システムの構築に関する研究

Research on construction of measurement system for food taking amountment using neuralnetwork

学籍番号　1010449　松本拓也

指導教員　竹田史章　教授

2001 年 2 月 5 日

情報システム工学科

要旨

これまで、ニューラルネットワークは紙幣識別に導入され、高い識別性能を発揮してきた。本論文では、病院内の入院患者の食事管理にニューラルネットワークを導入することを提案する。これまで、病院内の入院患者の食事管理は、栄養士が管理してきた。そのため、測定はあいまいで効率に問題があった。そこで、この問題を解決するため、画像処理にニューラルネットワークを応用した食物摂取量計測システムを提案する。

Up to date, the neural network was introduced in the banknote recognition, and a high recognition performance has been demonstrated. In this paper, we propose a meal management system for hospitalization patient by neurarl network. Conventionally, the dietitian has managed meal of patient in hospital. Therefore, the measurement is so vaguel and but so efficient. Then, in this paper, the system applied the neural network to the image processing is proposed to solve the problems.

キーワード：ニューラルネットワーク、画像処理、食事管理、切出し、摂取レベル分類

目次

１章　はじめに

これまでに、ニューラルネットワーク（以後ＮＮと略記する）を用いた研究では、紙幣識別へ適用がなされている。ＮＮの非線形識別能力が、印刷物である多種の劣化ノイズを含む紙幣データに有効であることを実験によって示し、パターンマッチングとの融合を図ることで、識別数の制約の回避と、新規識別パターンの追加の容易性を実現したものである[(1) (2)] 。だが、このシステムでは、紙幣の疲弊、汚れ、破れなどの紙幣状態の変動に対して十分な認識率を維持することは難しかった。そこで、これまでのニューロ識別ボードに学習機能が付加された。識別性能を識別機自体が管理でき、必要に応じて自己学習が可能な自律型ニューロ識別ボードである。本研究は、この紙幣識別用自律型ニューロ識別ボードを応用し、食事摂取量を識別する新たなシステムの開発に携わろうとするものである。

研究の背景として、病院内での入院患者の食事管理がある。その管理の現状は、患者に出す前の状態（食前）と患者が食事を終えた状態（食後）を栄養士が確認し、食品成分表と照らし合わせて、最終的な患者の栄養摂取量を算出するというものである。入院患者数、食事メニュー数、患者ごとに異なる摂取量と、それらを記録として残す作業を考慮すると、膨大な手間と時間が費やされている。研究のはじまりは、それらの作業を人の手に代わり、機械で計測・記録できないか、というところから出発した。

コンセプトは、「食前・食後の画像を比較して、摂取された栄養素を算出する」ことである。そのために基本となるシステムに、紙幣識別用の自律型ニューロ識別ボードを応用する。自律型ニューロ識別ボードが適用された理由は、紙幣画像の汚れや破れなど、状態変動に対応できる点にある[(1)]。患者に与えた食事の食後の画像が、紙幣識別で言うところの状態変動、つまり破れや汚れに相当する。これは、自律型ニューロ識別ボードが、食事摂取量計測システムに応用できる大きな理由の一つに挙げられる。

本論文では、食事摂取システムを構築していく上で、まず、紙幣識別に使用されている自律型ニューロ識別ボードを用いて実験を行なう。ここでは、紙幣ではなく、食事画像を入力画像とする。食前、食中、食後の３レベル分類が正しく行なわれるかを確認し、食事画像識別に対する自律型ニューロ識別ボードの基本的性能を確認する。また、入力画像に対するマスク位置を変えて、識別能力に違いがあるかを検討する。次に、ニューラルネットワーク学習システムのシミュレータを用いて、食事画像の識

別実験をおこなう。ここでは、これまで入力画像としてきた食事の写真ではなく、食物のダミーを作成し、採取した画像を入力画像とする。識別レベルを６つに増やし、詳細な分類が可能かを実験によって検討する。また、USB カメラを搭載した専用撮像装置を組み合わせた機器構成で識別実験を実施する。ここでは、食事摂取量計測システムの提案構成と同様の機器構成で識別実験を行なう。使用するニューロ学習システムは、食事摂取量計測専用に特化したニューロ学習システム（以降 NN－SOFT）を用いる。最初に、数字の２値画像を用いて識別実験を行なう。ここでは、NN－SOFT の基本性能を確認し、入力画像の歪みに対応できるかを実験によって検討する。次に、NN－SOFT のマスク作成に関するモードの実験を行なう。NN－SOFT には２つのマスク作成モードがある。ひとつはユーザがマスクパターンを自由に編集し、入力画像の任意の位置にマスクをかけることができるモードである。これを手動マスク作成モードと呼ぶ。一方は、入力画像全体に自動的にマスクをかけ、縦・横・斜めに走査してスラブ値を作成するモードである。これを自動マスク生成モードと呼ぶ。NN－SOFT を用いた実験では、２つのマスク作成モードで識別分類実験を行ない、食事摂取量計測に適したマスク作成モードを検討する。

食事摂取量計測システムでは、食事の計測に関して、食膳レベルでの識別ではなく、食器単体レベル、食材レベルでの計測を目的としている。そのため、食膳全体から食器と食材を分離する手法を検討する。切出し手法として CUTOUT を提案し、その性能の確認を行なう。ここでは、切出し領域の制限、収縮処理回数を変化させて、食器（食材）切出しに適した条件を検討する。

本研究では、ニューラルネットワーク学習システムの食事摂取量計測応用に際して、各々の実験ごとにもっとも良い条件を検討していく。また、実験によって新たに発生する問題と対策も検討し、論文全体を通して食事摂取量システムの有用性についても考察する。

２章　病院内における食事摂取管理

本研究では、病院内における入院患者の食事管理に関する問題点を解決する手法として、ニューラルネットワークを用いた食事摂取量を自動検出するシステムを提案する。

病院内における入院患者の食事管理に関する問題とは、患者が摂取した食事の量を計測する場合、栄養士が目視で残量から摂取量を決定するという方法が取られているため、測定値はあいまいで、測定基準が栄養士によって異なることである。患者一人ひとりの食事をすべて人の手で計測するため、作業効率も高くはない。また、患者個人の計測データの管理においても、同様の問題がある。

そこで、病院内において、ニューラルネットワークを用いた食事摂取量計測システムについてアンケートを実施した。アンケートの対象は、高知県内の病院に勤務する給食事業に携わっている方々である。以下に、アンケートの内容を記載する。

■実施アンケート

病院内において患者個人のエネルギー及び栄養素摂取量を正確に把握することは、栄養管理上非常に重要であるが、聞き取りによる残食料の調査等は人手の関係で大変です。そこで、短時間に各人の摂取量を検査するシステムがあればと考えます。高知工科大学との共同研究により、人手がかからず、短時間に給食の摂取量を検査し、エネルギーや栄養素の摂取量を正確に算出するシステムの開発に取り組みたいと考えます。これまでの研究では、カメラを内臓した画像認識装置を応用して、食事前後のトレイ上の食物の差からエネルギーや栄養素の摂取量を算出するシステムの開発が可能であると思われます。そこで給食事業に携わっている先生方のご意見をお聞かせ願いたく、アンケート調査をさせていただきます。

質問１．このようなシステムについてどう思いますか？
（1）必要性を感じる
（2）必要性を感じない
（3）考えていなかった

質問２．このようなシステムが開発されれば、導入したいと思いますか？
（1）導入したい

（2）興味がある
（3）検討してみた
（4）興味がない

質問３．導入できる価格はどの範囲ですか？
（1）10～20 万円
（2）20～30 万円
（3）30～50 万円

質問４．その他、疑問、質問等

このアンケートで 57 名からの回答が得られた。回答したのは、栄養士 55 名、医師１名、看護婦 1 名である。以下はその回答の集計結果である。

■アンケート回答

質問１の回答では、
（1）必要性を感じる…26 人（46%）
（2）必要性を感じない…２人（4%）
（3）考えていなかった…29 人（50%）

質問１で必要性を感じると答えた人で、質問２では、
（1）導入したい…5 人（9%）
（2）興味がある…16 人（28%）
（3）検討してみたい…5 人（9%）
（4）興味がない…０人

質問１で考えていなかったと答えた人で、質問２は、
（1）導入したい…０人
（2）興味がある…12 人（21%）
（3）検討してみたい…5 人（9%）
（4）興味がない…4 名（7%）

以上の回答から、このようなシステムを「考えていなかった」人が 50%ともっと

も多いが、「必要性を感じる」人も 46%いる。また、必要性を感じない人と導入に興味がない人を合計しても６名で 11%にすぎない。

このようなシステムを「導入したい」と答えた人は 5 人と 9%であるが、「興味がある」「検討してみたい」と答えた人は、必要性を感じる人のうち 21 人、考えていなかった人のうち 25 人と合わせて 46 名で 80%に達する。

質問３．導入できる価格の範囲は？
（1）10～20 万円…33 人（58%）
（2）20～30 万円…7 人（12%）
（3）30～50 万円…0 人
分からない、無記入…17 人（30%）

この結果から、低価格である方が導入しやすいことを示している。疑問、質問の項目では、「時間がないので人数分できない」「目で見たほうが早い」「メニュー上と調理後の実際の栄養量はかなり違っているのでは？」「病棟ごとに全体として残食料をチェックしている」「訪室やカルテから情報を収集している」などの記入があり、病院、施設の種類や規模によってもシステムへの考え方が異なっているようである。例えば、大病院では各階ごとにこのシステムを設置する必要性も考慮すべき点である。

このアンケート調査結果から、ユーザは食事摂取量計測システムに高い関心をもっていると予想でき、本研究の意義が高まる。

３章　提案システムの構成

＜3.1＞ニューラルネットワーク

本実験の根底にある、ニューラルネットワークとは、もともと人間の脳内に形成される神経回路網のことを指す。本論文でのニューラルネットワークは、神経回路網の構成をモデルとした情報処理手法のことであり、ネットワークの各ユニットの内部状態（あるいは出力）のパターンとして表現された情報を、ネットワークの状態変化規則を使うことによって変換する、といったように表すことができる[(4)]。

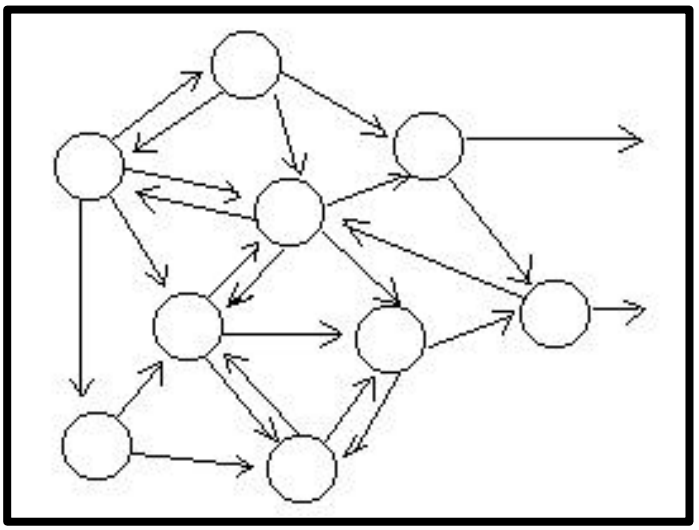


図 3.1　ネットワークメカニズム

また、１．ニューラルネットワークは、個々の要素ユニットはかなり単純で均一な処理しかしない、２．ユニット間には単純な情報しか流れない、３．ユニット同士はの結合の仕方によってメカニズムの動作を本質的に制御する、４．それぞれのユニットは、原則として非同期に勝手に動く…と特徴づけることができる。ここでのユニットとは、ネットワークのノードにあたる処理要素を指し、脳内神経回路網で言うところのニューロンに対応している。普通、多入力・一出力の素子で、入力に応じて内部状態を変え、それに応じた出力を出すものである。また、ニューラルネットワークで重要な意味を持つ「ユニット間の結合」は、神経回路網の信号伝達経路にあたり、それぞれの結合には重みと呼ばれる量が付随している。これは、結合が結んでいるユニット間の相互作用の強さをあらわすパラメータであると言える[(4)]。

こうした特徴を持つニューラルネットワークは、これまでの実験により、パターン識別や非線型識別に高度な能力を発揮することが知られている[(1)(2)]。

＜3.2＞ニューロ識別ボード

実験は、既存の紙幣識別用のニューロ識別ボードを使用する。このニューロ識別ボードは、ボード自体が学習機能をもっており、必要に応じて適切なウェイトを作成できるものである。

ハードウェア構成としては、従来のニューロ識別ボードに、モード設定スイッチ、ＬＥＤ、液晶表示板（ＬＣＤ）、Ｄ／Ａ変換器、パラレル通信ポート、デュアルポートメモリ（ＤＭＰ）、への延長ポートが追加されたものとなっている。ＬＥＤ、液晶表示板には、学習状況、動作モード、識別結果、エラーなどが表示される。ＤＭＰへの延長ポートは、ＰＣの内部バスと専用ケーブルで接続でき、イメージスキャナやデジタルカメラからのデータを使用できるようになっている[(1)]。図 3.2 にニューロボードの外観を示す。

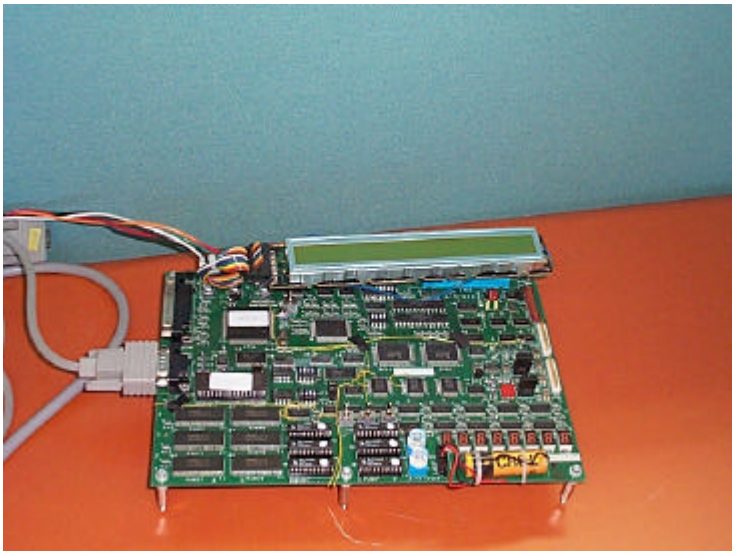


図 3.2　自律型ニューロ識別ボード

＜3.3＞　実験システムの構成

実験は、始めに、ＰＣ（pentium2／400Mhz）にニューロ識別ボードを接続し、イメージスキャナから学習・識別用の画像を取得する形態を取った実験を実施する。これは、ニューロ識別ボードの基本性能と、食物摂取量検出に応用可能か否かを検討するためである。

次に、ニューロ識別ボードのシミュレータプログラムを搭載したＰＣでの実験を行なう。ニューロ識別ボードを使用しないのは、シミュレータプログラムの性能試験のためである。また、実験効率の向上も兼ねている。ニューロ識別ボードは、実験を行

なうための操作が多く、手間がかかる。例えば、学習や識別を行うために、毎回モード切り替えスイッチを操作せねばならず、モード認識にボードのリセットボタンを押す必要がある。また、識別時の未学習画像の入力は、画像を１枚ずつしか入力できない。一方、シミュレータプログラムでは、画面上だけでほぼすべての操作が可能であり、手間のかかる作業も能率化がされている。

専用撮像装置を接続して学習・識別用の画像を取得できるようにしたノートＰＣ（こちらもシミュレータプログラムを搭載）での実験も行なう。

これは、製品化する際の基本システム構成にもっとも近い形態である。市場ではリアルタイム性が求められていると予想されるため、この方式が採用された。

最後に、食物摂取量検出システムの開発を共同で行っている、ヴィスト（株）が試作したシミュレーションプログラム（以降 NN－SOFT と呼称する）での実験を実施する。この NN－SOFT は、Ｃ言語でプログラミングされたニューロ識別ボードを、ＶＢに移植したものである。

４章　食物画像の識別

ここでは、紙幣識別用の自律型ニューロ識別ボードが食物画像の識別に有効であるかを実験によって示す。また、基本性能を確認する。そのために、ニューロ識別ボード、ＰＣ、スキャナを専用ケーブルで接続した機器構成で実験を行ない、サンプルの写真（高知学園短期大学より提供）を用いて正しく識別できるか否かを実験する。

＜4.1＞自律型ニューロ識別ボードによる実験

4.1.1 採取画像

まず最初に、食物画像の識別にニューロ識別ボードが有効であるか否かを検討する実験を行う。その手法として、サンプルの写真をスキャナで取りこみ、ニューロ識別ボードに学習させたのちに識別を行なうという方法をとる。学習・識別のためのサンプル画像は、高知学園短期大学より提供された写真を使う。写真は、入院患者に配膳された食事という設定で、トレイに６種類の異なる器に、お茶、ごはん、豆、漬物、エビチリ、サラダが盛り付けられている。食前（配膳直後）、食中（患者が食事を約半分残した場合）、食後（完食後）の３段階のレベルがあり、食前写真＝9 枚、食中写真＝8 枚、食後写真＝４枚が提供された写真である。各写真の撮影距離、アングル、器の配置、撮影時の明度などにはばらつきがある。撮影条件が近い写真も数枚あるものの、すべてが統一されてはいない。

図 4.1　サンプル写真の例高知学園短期大学提供）

図 4.1 に代表的なサンプルの写真を示す。
実験手順として、まずサンプルの写真をスキャナで取りこみ、ビットマップ画像（256 階調グレイスケール）に変換する。

ビットマップ形式でなければならない理由は、ニューロ識別ボードが認識できる専用画像ファイル形式であるニューロフォーマットを作成するために必要だからである。ニューロフォーマットを作成するには専用アプリケーションを使用する。このアプリケーションではビットマップ以外のファイル形式では、ニューロフォーマットへ変換に対応していない。

次に、グレイスケールにすることの理由は、もともと紙幣識別用ニューロ識別ボードがニューロフォーマットにのみ対応しており、それ以外のファイル形式は受容しない。ニューロフォーマット作成あたって、前述したように、元画像としてのビットマップ画像が必要となり、この段階でのビットマップ画像が、グレイスケールである必要がある。これは、ニューロ識別ボードが、学習あるいは識別時に、対象画像の濃淡を見ており、カラーの画像では正確に認識を行えないからである。ニューロフォーマット作成の際にカラー画像をグレイスケールに変換する機能はないため、ニューロフォーマットに変換する前の段階でグレイスケールにしておく必要がある。

食事画像においても、一度グレイスケールでビットマップ形式の画像を採取する必要がある。グレイスケールへの変換とファイル形式は、スキャナで画像を採取する時に選択できる。画像を取りこむ際には、解像度を 72gps としている。これも、ニューロ識別ボードが紙幣用であることの制約である。これ以上の解像度の画像はシステムが受け付けず、エラーを返す。

また、ニューロフォーマットの変換元となるビットマップの名前は、すべて統一されたものでなければならない。たとえば、ここで使用しているビットマップファイルの名称は、Food0101.bmp といった具合である。「Food」は、すべて共通の名称である。「0101」は、カテゴリ番号で、画像パターンの種類とパターン内でのナンバーを表している。前半の２桁が画像パターンを示し、後半の 2 桁がナンバーを示す。ここでは、サンプル写真の枚数とレベルに対応するので、カテゴリ番号は、0101～0108、0201～0209、0301～0304 となる。

4.1.2 環境設定

最初に行なう実験では、ＳＣＡＮＤＥＮＯと呼ばれるニューロ識別ボードのユーティリティツールを使う。ニューロフォーマット作成からニューロ識別ボードの学習までのプロセスを通して実行する形式となっており、ユーティリティツールであるＳＣＡＮＤＥＮＯとニューロ識別ボード自体の操作によって進行していく。

まず、環境の設定について述べる。ニューロ識別ボードをメンテナンスモード（モード設定スイッチの 1 番・４番を ON、２番３番を OFF）にする。SCANDEMO.BAT

を起動後、メニューから「File Handring」続けて「Create　Weight Header」を選択すると、パラメータを決めるウェイトファイルの作成画面にはいる。
ここでは、学習枚数（num of lrn ptrns）と各画像のパターン数(num of lrn ptrn)の設定を行う。学習枚数を各パターン２枚ずつ、画像パターン数はサンプル画像が食前・食中・食後の３パターンなので３と設定する。作成したウェイトファイルは、ニューロ識別ボードのフラッシュメモリに書き込む必要がある。これは、SCANDEMOの初期メニューから「File Download」を選択し、作成したウェイトファイル名を指定することで、フラッシュメモリへのダウンロードが完了する。

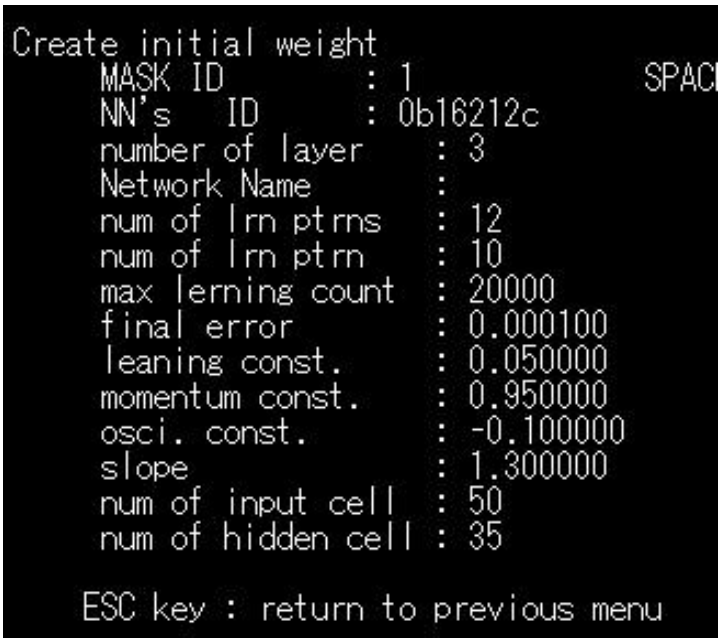


図 4.2　ウェイトファイル編集画面

次に、SCANDEMO 側で学習の設定を行なう。PC 側は、SCANDEMO.CFG という設定ファイルを操作する。ここでは、学習パターン／出力ユニット数、提示枚数、画像切出し領域、BMP_PATH を操作する。学習パターン数／出力ユニット数は、文字通り学習させるデータの種類数と出力ユニット数を指すので、３とする。提示枚数は、各パターンごとに学習させる枚数で、ここでは２としている。画像切出し領域は、切出し開始行・切出し終了行・切出し開始列・切り出し終了列の４つで構成され、対象画像のサイズに合わせて数値を入力する。また、元データとなるビットマップ画像が大きい場合は、必要な領域のみを指定する。画像の一部を指定する場合には問題が生じるのだが、後に行う実験で記述することとする。BMP_PATH は、学習させる元となるビットマップデータを指定する。

ニューロ識別ボード側のウェイトファイルの学習枚数（num of lrn ptrns）と各画像のパターン数(num of lrn ptrn)が、SCANDEMO.CFG の学習パターン／出力ユニット数と提示枚数に対応しているため、各カテゴリの値は同値でなければならない。

図 4.3　CFG ファイル編集画面

DATABASE.CFG、SLABBAE.CFG **という別の設定ファイルがあるが、これらはデータ供給方法が異なる場合に必要になる設定ファイルであって、本実験ではビットマップデータをシリアルポートから供給する方法を取っているため、操作しない。**

4.1.1 **で記述した、ニューロ識別ボード専用フォーマットであるＤＢファイル（データベースファイル）の作成について述べる。各設定ファイルを操作後、ニューロ識別ボードを学習・識別モード（モード設定スイッチ全てを** OFF**）にし、**SCANDEMO **を起動する。初期メニューから「**Leaning**」を実行し、**Online **か** Offline **かの選択をする。ここでは** Offline **を選択する。**Online **は基盤側が未対応となっているため、使用できない。メニューをすべて選択し終えると学習が始まる。このときに自動的にビットマップデータからニューロフォーマットが作成され、学習が始まる。**

4.1.3 学習と識別

ボードと PC **の通信プロトコルはコマンドレスポンス方式がとられている。**PC **上でモニタプログラムを動作させ、モニタプログラムの命令をボードに搭載された** DSP **が割り込みとして受け取り、**DSP **が割り込みを許可した時点でコマンドに応じたレスポンスを** DSP **の中間情報とセットにしてモニタプログラムに返信する。学習状態は、そのときの学習回数、学習誤差、学習定数が棒グラフと折れ線グラフで表示され、学習が収束した時点で終了する。**

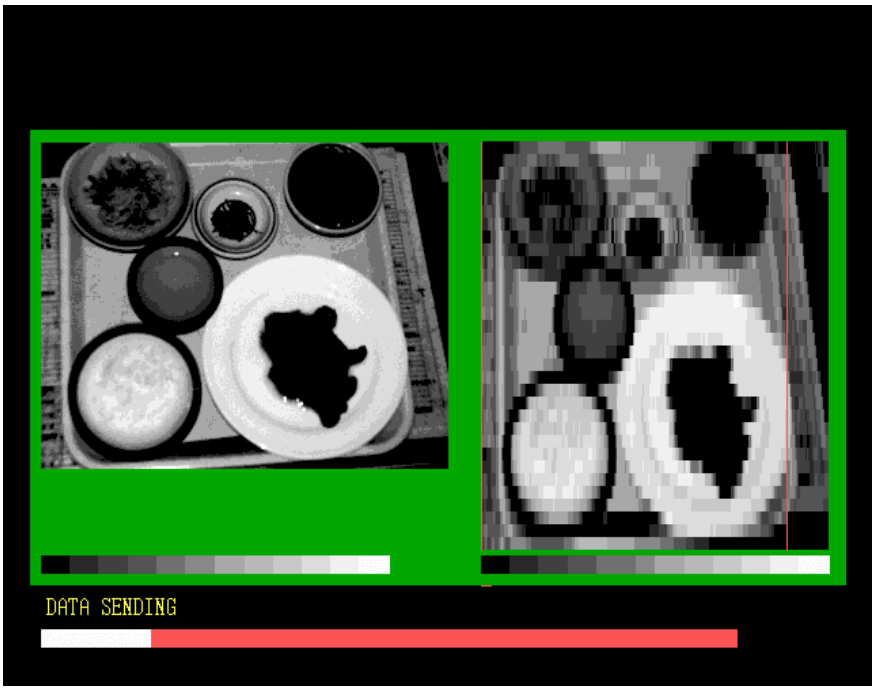


図 4.4　ニューロフォーマット変換時の表示例

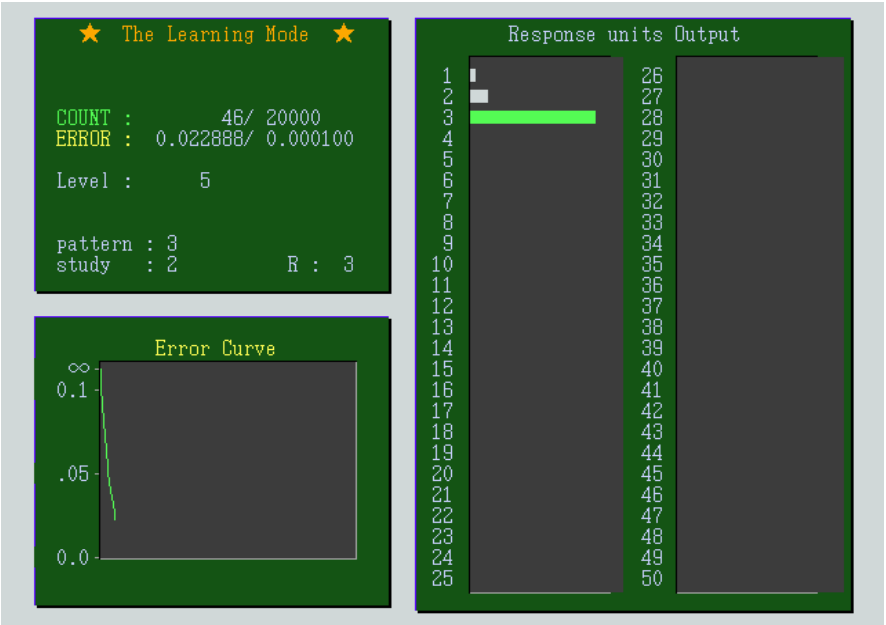


図 4.5　モニタプログラムによる
学習状況の表示例

次に識別を行なう[5]。その手法を以下に述べる。識別は、識別画像のニューロフォーマットへの変換、マスク処理、抽出した特徴量と学習データの比較という流れで展

開される。従来、ニューラルネットワークによる識別手法は、対象の特徴抽出としてマスク処理機構を前処理とし、そのニューラルネットワーク構造は非線型識別が可能な最小構成（入力層、中間層、出力層）の３層構造となっている[(1)~(3)]。マスク処理の具体的手続きは、対象画像を部分的に被覆するマスクで覆い、被覆されない画素の合計値（スラブ値）を対象画像の特徴量として NN への入力としている[(3)]。

具体的には、まず、SCANDEMO の初期メニューから EVALUTE を選択し、認証に使用するビットマップファイル名を入力する。指定されたビットマップファイルはニューロフォーマットに変換され、マスク処理が行われる。ここでは、マスクの編集機能がないため、既存の紙幣用のものを用いる。

このようにして、食物画像の学習を行い、食前・食中・食後の３レベル識別実験の結果を表 4.1～表 4.3 に示す。学習画像は食前＝Food0101、Food0102、食中＝Food0201、Food0202、食後＝Food0301、Food0302 である。

表 4.1　食前画像の識別結果

食前画像	発火率	識別	判定
FOOD0101	0.752106	食前	○
FOOD0102	0.819021	食前	○
FOOD0103	0.753848	食中	×
FOOD0104	0.366374	食後	×
FOOD0105	0.619953	食前	○
FOOD0106	0.760458	食前	○
FOOD0107	0.599887	食前	○
FOOD0108	0.512479	食中	×

表 4.2　食中画像の識別結果

食中画像	発火率	識別	判定
FOOD0201	0.686136	食中	○
FOOD0202	0.761069	食中	○
FOOD0203	0.410528	食中	○
FOOD0204	0.427011	食中	○
FOOD0205	0.488928	食後	×
FOOD0206	0.494312	食中	○
FOOD0207	0.397546	食前	×
FOOD0208	0.548283	食後	×
FOOD0209	0.470547	食中	○

表 4.3　食後画像の識別結果

食後画像	発火率	識別	判定
FOOD0301	0.814852	食後	○
FOOD0302	0.862586	食後	○
FOOD0303	0.896273	食中	×
FOOD0304	0.442438	食後	○

アングル(a)

アングル(b)

図 4.6　撮影アングルによる画像の差違

この結果から、約 67%で正しく識別できる能力があることがうかがえる。ところが、この実験で使用したサンプル画像には、対象食物との撮影距離、器の配置、撮影アングルなどにばらつきがあり、特に撮影アングルではその差が顕著に見られ、同じレベルに属する画像でも、まったく異なる画像として認識する可能性が大きいと考えられた。そこで、アングルが大きく異なるサンプル画像＝Food0104、Food0207、Food0208、Food0209、Food0302、Food0303 を排除したところ、識別率は 80%に達した。紙幣識別用の自律型ニューロ識別ボードでの実験結果から、食物摂取量検出システムに、NN は十分応用できると判断できる。

＜4.2＞ NN シミュレータによる実験

これまでの実験は、自律型ニューロ識別ボードを使用して実施してきた。ここでは、そのシミュレーションプログラムで実験を行なう。

自律型ニューロ識別ボードを接続した実験システムは、PC 側の操作とボード側の両方の操作が不可欠であった。また、マスクの編集が自由にできず、識別においても、画像ファイルを１つずつしか選択できないなど、非常に手間のかかるものであった。しかし、このシミューレータプログラムは、それらの手間が省略され、使い勝手の良いものとなっている。具体的には、各種の設定ファイルの編集が容易であること、識別時に識別させたい画像ファイルをまとめて扱えること、結果を LOG ファイルとして作成できること、それらの作業がすべて PC 上だけで行える、などの点で実験効率が向上している。また、このシミュレータプログラム自体にニューロフォーマットへの変換機能は付属されていないため、SCANDEMO で別個に作成しなければならない。

4.2.1 採取画像

この実験では、＜4.1＞の、食物の写真から作成したビットマップファイルをそのまま使用する。ニューロフォーマットへの変換は、前述したように SCANDEMO 上で行なう。

まず、SCANDEMO を起動するのだが、SCANDEMO．EXE からコマンドオプション「offline」と入力し、オフラインモードで起動する。オンラインモードは、ニューロ識別ボードが接続されている場合で、この実験では使用しない。SCANDEMO 起動後、初期メニューから「DATA CONVERT」を選択すると、SCANDEMO．CFG で指定したビットマップファイル群が、カテゴリナンバー順にニューロフォーマットに変換されていく。このニューロフォーマットは、レベルごと（ここでは食前・食中・食後）に DB（データベース）ファイルに格納される。各 DB ファイルごとに同レベルの画像データが順に格納されるため、ビットマップファイルの数と同数の DB ファ

イルが生成されるわけではない。BD ファイル内のニューロフォーマットデータは通常では確認できないが、コマンドオプション「view_img」で閲覧することができる。また、ニューロ識別ボードを使用した実験で、のちに追加された機能で DB ファイルを作成することは可能だが、こちらで作成した DB ファイルはサイズが異なるため、シミュレータプログラムでは使用できないことが判明している。

4.2.2　環境設定

シミュレータプログラムの環境設定を行う。実際に操作するファイルは DATABASE.CFG、DATABASE.TBL、EVAL.BAT、MASK.TXT、NEURO.CFG の５つである。DATABASE.CFG、DATABASE.CFG は、学習データ抽出に関するファイルで、ここに記載された DB ファイルを、記載された順番に読み込む。EVAL.BAT は識別に関する設定ファイルで、識別元となる DB ファイル名、追加データ名、リジェクトデータ名、識別結果である LOG ファイル名を記述する。MASK.TXT はマスク処理に関する設定ファイルで、マスクサイズやマスクの位置などを決定する。NEURO.CFG は、学習システムの環境設定ファイルである。ここでは、学習パターン数＝3、１パターンの定時枚数＝２とする。学習定数、振動定数などは最適な数値が設定されている。

4.2.3 学習と識別

学習から識別までは、学習画像の特徴抽出、学習、マスク処理、スラブ値作成、識別という流れで展開する。シミュレータでは学習から識別までを継続的に行い、プログラム実行中でも各処理の実行前に設定ファイルの内容変更が可能である。

図 4.7　特徴抽出処理の例

特徴抽出処理が終了すると、続けてマスク設定ファイル変更の選択が表示されるので、任意に選択する。良ければ、マスク処理が開始される。ここでは紙幣用のマスクではなく、マスク範囲を変えて食物メニュー全体を対象として処理を行う。図 4.8 では、マスクの届いていない部分があるが、これはマスクサイズの制限によるもので、最大マスクサイズで処理しているため、これ以上は範囲を拡大できない。また、マスクをかける範囲をさらに変えて、ひとつの器にマスクをかけて識別する実験も行なった。図 3.8、図 3.9 に全体マスク・部分マスクでの範囲の違いと識別結果を示す。学習画像には、Food0101、Food0102、Food0202、Food0301、Food0302 を使用し、識別画像には Food0103、Food0105、Food0203、Food0204、Food0303、Food0304 を使用した。

図 4.8　対象画像全体マスク

図 4.9　対象画像部分マスク

表 4.4　全体マスク識別結果（食前）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.870852	0.092274	0.011670	食前	○
0.815138	0.011114	0.166111	食前	○

表 4.5　全体マスク識別結果（食中）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.169468	0.583439	0.026636	食中	○
0.845513	0.120347	0.010863	食前	×

表 4.6　全体マスク識別結果（食後）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.004080	0.090068	0.962706	食後	○
0.000567	0.013655	0.998847	食後	○

表 4.7　部分マスク識別結果（食前）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.458140	0.373229	0.012802	×	×
0.543969	0.057833	0.066378	食前	○

表 4.8　部分マスク識別結果（食中）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.018463	0.889169	0.036440	食中	○
0.351123	0.329683	0.021903	×	×

表 4.9　部分マスク識別結果（食後）

発火率（食前）	発火率（食中）	発火率（食後）	識別	判定
0.004753	0.025411	0.978279	食後	○
0.000567	0.039217	0.996191	食後	○

識別には学習データを除く未学習データを使用し、撮影アングルが大きく異なるものについては除外した。全体にマスクをかけて識別した場合、約 83%の識別率を示し、ひとつの器にマスクをかけて識別した場合は、約 67%を示した。ここで、一つの疑問が生じる。全体マスクで処理した場合、部分マスク処理に比べて対象画像が複雑な分、あきらかに識別が困難なはずである。それにもかかわらず、部分マスクでの識別よりも全体マスクでの識別は識別率が高い。これは、マスクをかける処理の部分に問題があると推測される。シミュレータプログラムでマスクをかける場合、ユーザが視認できるように図 3.10 のように表示されるのだが、このマスクは DB ファイル内の先頭の画像データのみに対して行なわれる。つまり、DB ファイル内の先頭以降の画像データは、先頭の画像データにかけたマスクと同じマスクで同じ座標を被覆されるため、画像データによっては、まったく意図しない部分にマスクがかかることになる。全体にマスクをかける場合はそれほど問題にはならないが、画像の一部にマスクをかける場合は、対象とする部分（この実験では器）が同じ位置になくてはならない。それを踏まえて、部分マスクで識別を行なった画像のうち、器の位置の違いでマスクがずれたものを除くと、実に 100%の識別率を示す結果となった。

＜4.3＞　食物ダミーを用いた実験

ここまで、ニューロ識別ボードを用いて、ＮＮが食物摂取量計測に応用できるか否かを実験し、シミュレータプログラム上でマスクの対象を変更することで識別率にどのような変化が生じるのかを実験した。いずれも、条件がそろえば、十分な識別率を示し、NN が食物摂取量測定に応用できる結果を出している。

しかし、これらの実験では、画像に統一性があるとはいえなかった。実験に用いた画像は、すべて提供された写真を画像データに変換して行なった。それらには撮像距離やアングル、レベルごとの枚数に差違があり、同一条件で撮影されたものではなかった。そこで、実験に使用する画像に同一条件を与え、さらに詳細な識別分類実験を実施する。

4.3.1　食物ダミー画像採取法

これまでは、提供されたサンプル画像のみで実験を行ってきた。しかし、これでは、分類レベルの限界、画像のばらつきなど、学習・識別において多くの問題があった。そこで、これらの問題を解決するために、発泡スチロールを加工した食物のダミーを作製した。ダミーの組み合わせで、ある程度メニューにバリエーションを持たせることができる。また、画像を採取する際にダミーの分量を調整することで、食前・食中・食後の３レベルだけでなく、その中間も再現できるようになっている。

図 4.11　食物ダミー／キャベツ・とんかつ

図 4.12　食物ダミー／魚・とんかつ

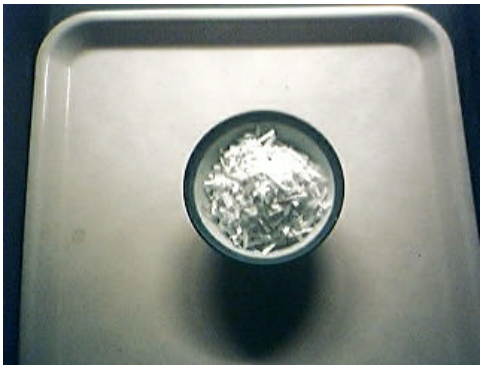


図 4.13 食物ダミーごはん

食物ダミーの製作後、学習・識別用の画像データを採取する。これまでは、写真をスキャナで取りこむことで、ビットマップファイルとして保存していたが、立体である食物ダミーでは、同様の手法ではビットマップデータが作成できなかった。そこで、ＵＳＢカメラを使用した専用の撮像装置を設計した。この撮像装置は、ダンボールを加工したもので、天井部分に設置したＵＳＢカメラを、シミュレータプログラムを実装したノートＰＣに接続したものである。画像を採取するときは、撮像台低部に食物ダミーをセットし、ノート PC からＵＳＢカメラのシャッターを操作して画像を採取する。内装は、光を反射しないように黒くし、光源は室内蛍光灯で補う。図 4.13、図 4.14 に撮像システムの外観と装置天井部に設置したカメラの様子を掲載する。

図 4.13　撮像システム

図 4.14　撮像装置カメラ部

このＵＳＢカメラを使用した撮像装置によって、これまでの実験で統一されなかった撮像時の条件をそろえることができるようになった。ひとつは撮像距離である。ＵＳＢカメラを固定することで、撮像距離が一定になった。この実験での撮像距離は約 60cm となっており、これは、食事のトレイ全体がカメラのフレームに収まる限界距離によって決まったものである。また、カメラを固定することによって、写真を用いた実験で問題となった撮影アングルのばらつきもなくなった。この実験では、撮影対象に対して真上からではなく、若干の角度を付けて撮影している。そして照明だが、こちらは室内の蛍光灯を光源としている。画像採取時には、特定の場所に設置位置を決め、撮像台を固定して行なっている。

4.3.2　食物ダミーで構成したメニューでの識別

これまでの実験では、写真での食事画像を識別してきたが、より実物に近いダミーによる食事画像での識別時における識別率はどうなるか検証する。

食物ダミーを組み合わせて食事のメニューを作成し、専用撮像装置で画像を採取する。レベルはこれまでと同様に、食前、食中、食後の３段階で行なう。各レベルの画像ごとに、ある程度の変化を持たせるため、食材の位置を少しずつ変えることとする。図 4.15 は、専用撮像装置で採取した実験用の画像である。

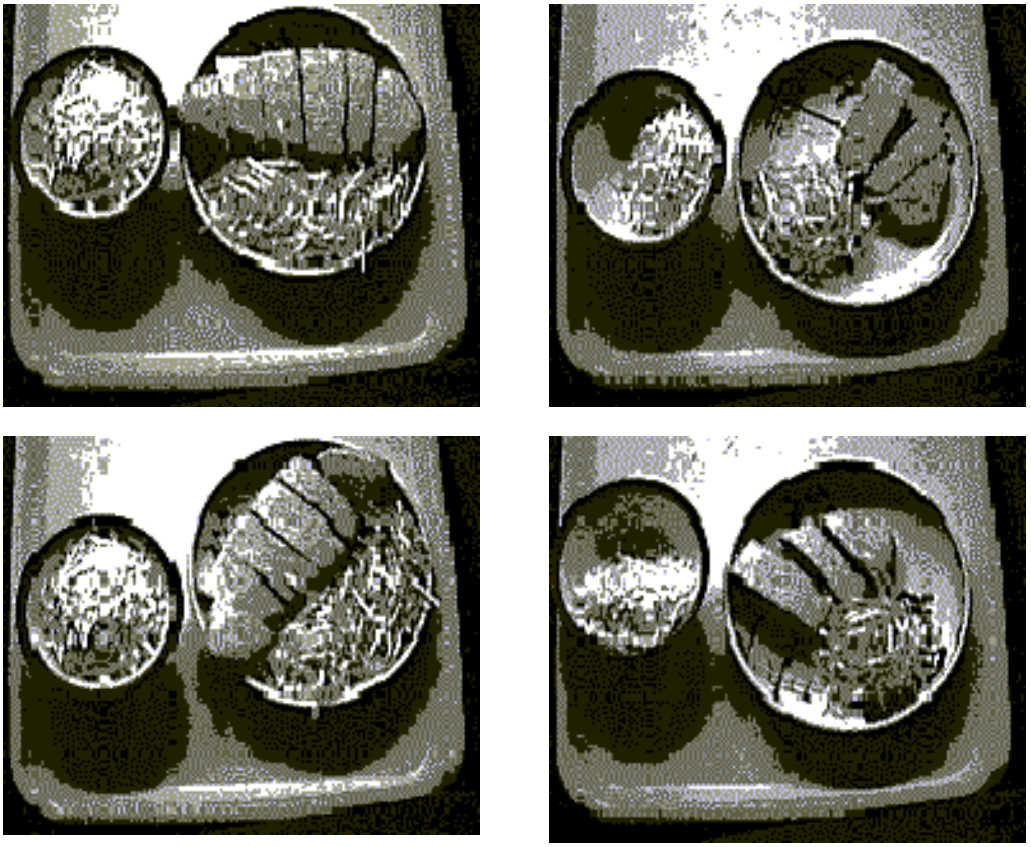


図 4.15　食事ダミーで構成したメニューの例

採取した食事画像は、これまでの実験のようにグレイスケールのビットマップ画像に変換したものを実験に使用する。次に、＜4.2＞のシミュレータプログラムでの実験と同様に、SCANDEMO を使用してニューロフォーマットデータに変換する。このとき、作成する DB ファイルはこれまでどおりの３つだが、格納するニューロフォーマットデータは各レベルごとに９個とする。学習・識別をおこなうシミュレータプログラムの各種設定ファイルの変更は、MASK.TXT である。マスクをかける位置を変更し、図 4.13 の大きな器をマスクで被覆する。また、器全体を覆うようにマスクのサイズを調整した。NEURO.CFG の学習パターン数／出力細胞数などは＜4.2＞の実験と同値とする。

以下に識別結果を提示する。なお、食後の画像の識別結果は割愛し、食前と食後の識別結果のみを掲載している。食後の画像は完食時で器上にまったく食材がない状態のため、識別でほぼ誤認しないからである。これは、これまでの実験結果から判明している。

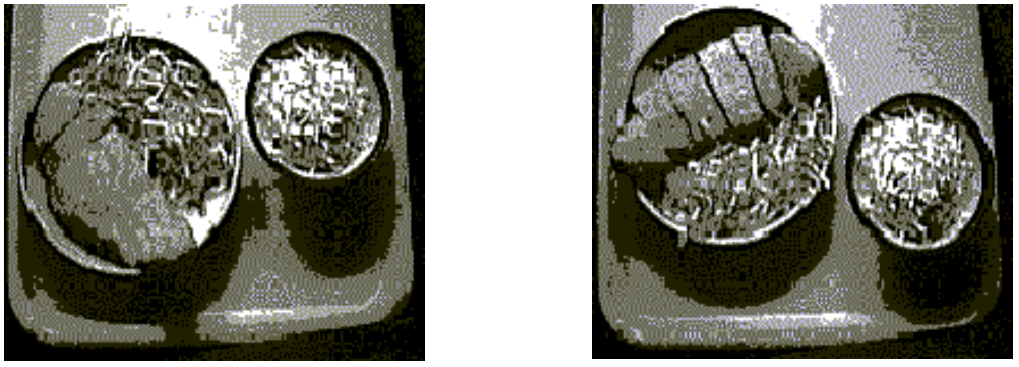
表 4.10　食物ダミーによる識別結果（食前画像）

食前画像	発火率	識別	判定
KATU01 01	0.980598	食前	○
KATU0102	0.995323	食前	○
KATU0103	0.657425	食前	○
KATU0104	0.667788	食前	○
KATU0105	0.172119	食中	×
KATU0106	0.080816	食中	×
KATU0107	0.069249	食中	×
KATU0108	0.03092	食中	×
KATU0109	0.715227	食前	○

表 4.11　食物ダミーによる識別結果（食中画像）

食中画像	発火率	識別	判定
KATU0201	0.993943	食中	○
KATU0202	0.980873	食中	○
KATU0203	0.984375	食中	○
KATU0204	0.95545	食中	○
KATU0205	0.962956	食中	○
KATU0206	0.104589	食前	×
KATU0207	0.953478	食中	○
KATU0208	0.972794	食中	○
KATU0209	0.91294	食中	○

この結果では、食物ダミーによる食前画像は約 56%、食中画像は約 89%の識別率を示した。ところが、写真での実験で問題となった、画像のアングル、撮影距離、明



(a)　(b)

図 4.16　器の位置ずれ

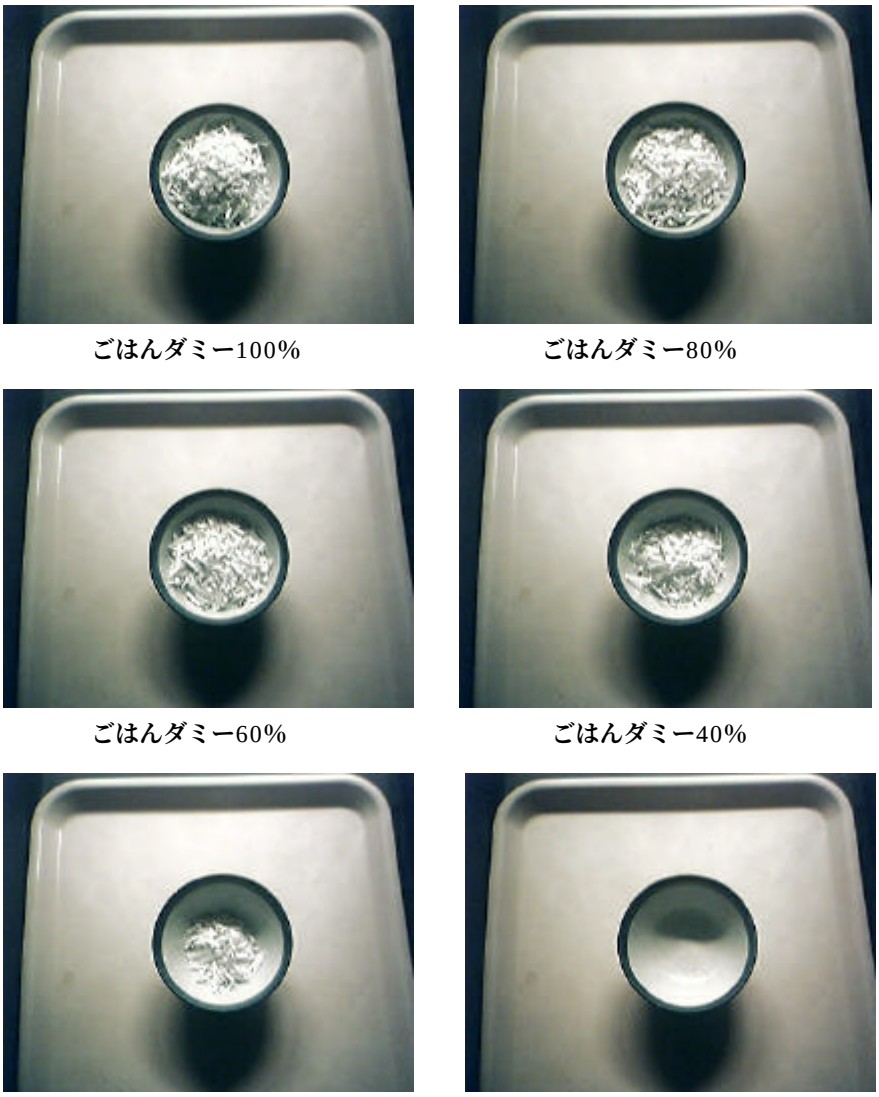
度の３点は、撮像装置の導入によってクリアしているが、ダミーを乗せた器は手作業で配置しているため、位置のずれまでは修正しきれなかった。これは、＜4.2＞での部分マスクの実験と同様の問題で、マスクの被覆が対象に対して正確におこなわれていないことによって生じた結果であると考えられた。図 4.16 は、手作業での配膳によって生じた器の位置ずれの画像である。

画像(a)、(b)の右側の器のような位置ずれを考慮して、器の位置ずれによって対象にマスクの被覆が及ばなかった画像を排除する。食前画像では、KATU0105～KATU0109、食中画像では KATU0207～KATU0209 が、器の位置が大きく異なっており、先頭画像に合わせたマスクでの被覆位置がずれていたため、排除する。それらの画像を除いた場合の識別率は、食前画像・食中画像ともに 100%であった。より実物に近い食物ダミーでも、写真の場合と同等の識別率を発揮した結果から、実物の食事摂取量計測への対応度は高いものと言える。

4.3.3 食物ダミー単体での識別実験

これまでは、サンプルの写真や食物ダミーを組み合わせた画像を用いて実験してきた。ここで、最終的な食物摂取量測定システムの完成品（詳細は 5 章で記述する）では、食事のメニュー全体から栄養の摂取量を計測するのではなく、食材単位で摂取量を算出し、各器の食材ごとの計測値を合計することで、メニュー全体の栄養摂取量を測定する方式を採用する。そこで、より最終形態の食物摂取量計測システムに近いかたちでのシミュレートが必要になってきた。ここでは、数種類作成した食物ダミーのうち、ごはんのみを使用した食材単品・部分マスク被覆での識別実験を行なう。

実験用のデータ作成には、食物ダミーのごはんのみを使用する。使用画像データは、専用撮像装置にて画像を採取し、グレイスケールのビットマップ画像に変換したものを使う。また、これまで食前、食中、食後の３レベルの画像に分類していたものを、２倍の６レベルに増やした。実験画像は、ダミーの内容量を実際に食べたのと同じように減量させながら撮像し、食前画像を 100%として、80%、60%、40%、20%、０%と、20%きざみで変化させたものとした。また、器の位置ずれを解決する手段として、トレイにマーキングをすることで器を完全に同じ場所に配置できるようになり、問題を解決している。データはレベルごとに６個の DB ファイルを作成する。格納されるニューロフォーマットデータとして各レベルごとに 5 個のニューロフォーマットを用意し、そのうち２個を学習に使用する。図 4.17 は実験用の食物ダミーの画像データ例である。



ごはんダミー100%　　ごはんダミー80%

ごはんダミー60%　　ごはんダミー40%

ごはんダミー20%　　ごはんダミー0%

図 4.17　食物ダミー（ごはん 100%画像～０%画像の例）

これらの画像を使用し、シミュレータプログラムで学習・識別を行う。実験形態の変化によって、シミュレータプログラムの各種設定ファイルを変更ずる必要がある。NEURO.CFG では、学習パターン数＝６、出力細胞数＝６とし、１パターンの提示数＝２とする。DATABASE.CFG では、参照する DB ファイルの増加に合わせて６つの DB ファイルを順に記述する。DATABASE.TBL も同様に記述する。
EVAL.BAT にも参照する DB ファイルを追加変更し、識別結果を出力する LOG ファイル名をそれぞれ R100～R0 と変更しておく。MASK.TXT では、マスクをかける位置を器の位置に修正し、サイズを調節する。ここでは、中心からのオフセット X=－39、中心からのオフセット Y=－10、マスク単位の X サイズ＝７、マスク単位の Y サイズ＝５である。 表 4.12～表 4.17 は、この設定条件で学習し、識別した結果を LOG ファイルを参照して記述したものである。

表 4.12　食物ダミー・100%画像の識別結果

100%画像	発火率	識別	判定
RICE0101	0.997158	100%	○
RICE0102	0.994220	100%	○
RICE0103	0.984375	100%	○
RICE0104	0.010369	80%	×
RICE0105	0.128354	80%	×

表 4.13　食物ダミー・80%画像の識別結果

80%画像	発火率	識別	判定
RICE0201	0.982960	80%	○
RICE0202	0.995252	80%	○
RICE0203	0.927061	80%	○
RICE0204	0.981747	80%	○
RICE0205	0.976680	80%	○

表 4.14　食物ダミー・60%画像の識別結果

60%画像	発火率	識別	判定
RICE0301	0.993513	60%	○
RICE0302	0.985013	60%	○
RICE0303	0.811029	60%	○
RICE0304	0.845922	60%	○
RICE0305	0.780776	60%	○

表 4.15　食物ダミー・40%画像の識別結果

40%画像	発火率	識別	判定
RICE0401	0.988986	40%	○
RICE0402	0.985946	40%	○
RICE0403	0.041255	×	×
RICE0404	0.401150	×	×
RICE0405	0.000060	60%	×

表 4.16　食物ダミー・20%画像の識別結果

20%画像	発火率	識別	判定
RICE0501	0.991634	20%	○
RICE0502	0.987421	20%	○
RICE0503	0.772098	20%	○
RICE0504	0.922858	20%	○
RICE0505	0.984993	20%	○

表 4.17　食物ダミー・0%画像の識別結果

0%画像	発火率	識別	判定
RICE0601	0.996074	0%	○
RICE0602	0.993463	0%	○
RICE0603	0.996900	0%	○
RICE0604	0.994095	0%	○
RICE0605	0.967085	0%	○

識別を行なった結果、上記の表のようになった。残量 100%画像（食前）の識別では識別率＝60%、残量 80%画像の識別では識別率＝100%、残量 60%画像の識別では識別率＝100%、残量 40%画像の識別では識別率＝40%、残量 20%画像の識別では識別率＝100%、残量 0%画像（完食後）の識別では識別率＝100%を、それぞれ示す結果となり、全体的な識別率では約 83%であった。

画像的な条件は統一し、マスクによる被覆位置も固定されたが、やはり微細な実験条件は整えることは難しい。この実験でも、実験画像の採取の際、レベル分類に器の食物ダミーの内容量を手作業で減らしているため、厳密には 20%きざみの画像ではないはずである。さらに、現在の採光手法では食物ダミーの撮像時に影ができるため、誤認識の原因となっている可能性もある。また、分類レベルが詳細になれば、その画像的な差違は見分けがつかなくなり、正しい認識はさらに難しいものとなると予測される。

実験に使用する画像の条件を整え、より認識しやすい画像とするためには、現在使用している専用撮像装置の精度を上げるほかない。また、分類レベルの詳細化に対応していくには、画像だけを NN にかけて識別する手法では限界があり、別の手法を考察する必要がある。

4.3.4 多段階分類の識別手法の考察

食物摂取量計測システムの最終形態では、実際に病院の入院患者の食事を想定している。実際の食事では、分類レベル数は無限といってよい。分類レベル数に制約を付けて限定することも可能ではあるが、その分、摂取した栄養量の計測値と実際に摂取した栄養量に差が生じ、正確な食事摂取量は計測できなくなる。そこで、これまで行ってきた画像だけによる識別とは異なる手法を考察した。

画像を NN によって識別する実験では、識別結果のみを重視してきた。しかし、実験を重ねると、NN による識別結果だけでは多レベル分類に対応できないのではないかという予測ができた。そこで、識別時に発生する発火率を利用できないかと考えた。発火率とは、反応値のことである。識別では、入力された画像が学習した画像のどれにもっとも近いのか、数値化して識別結果を表示する。NN は、発火率がより高い数値のものへと分類するのだが、他の分類カテゴリに対しても発火する。表 4.18 は、識別の LOG ファイルである。

<ライス８０%>

(100%)	(80%)	(60%)	(40%)	(20%)	(0%)
0.006682	0.982960	0.008053	0.000114	0.000024	0.000000
0.001413	0.995252	0.003094	0.007414	0.000003	0.000000
0.004236	0.927061	0.005793	0.000652	0.000004	0.000000
0.001926	0.981747	0.001611	0.012784	0.000003	0.000000
0.001775	0.976680	0.009167	0.002166	0.000009	0.000000

表 4.18　80%画像の LOG ファイル

この表は、80%の食物ダミーを識別した結果の LOG ファイルである。識別結果はすべて正常に 80%の画像として認識しているのため、80%の行に 0.9…という高い数値が並んでいる。ところが、100%の行にも、大きな数値ではないにしろ、多少の発火が認められ、別のレベルにも発火が見て取れる。新しい識別手法は、発火率の大きさをレベル分類に利用するものである。そのままではレベル分類には使えないため、発火率×定数＝レベルといった形態となる。

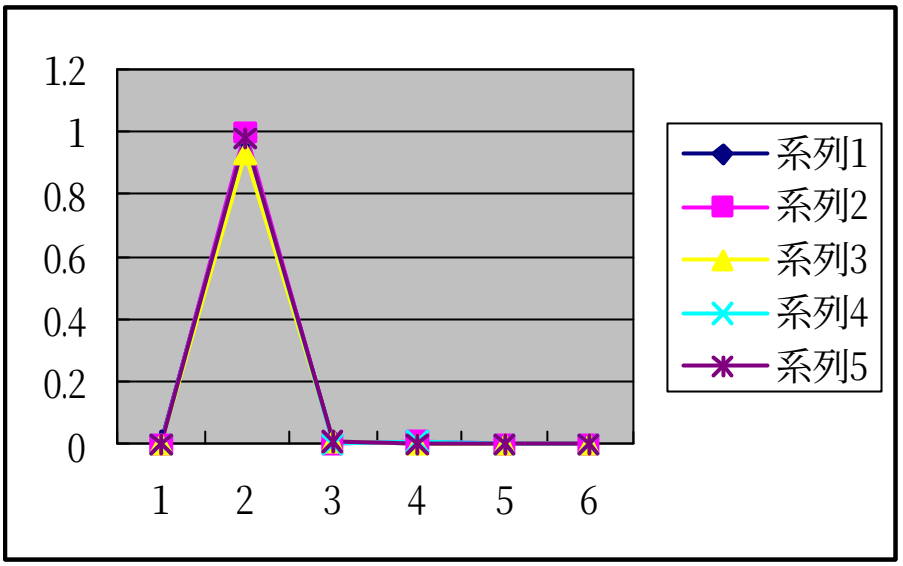


図 4.18　発火率の変

これは、レベル分類に使用する発火率を、食物ダミー単体での識別実験の識別結果からグラフ化したものであるが、どれも正常な認識レベルの部分が高く、その他は緩やかな発火であるといえる。しかし、グラフでは、正常な認識レベル以外の発火が低すぎること、低い数値どうしが近い値をとることなどの理由から、レベル分類の目安にはならないため、発火率を利用した多レベル分類手法は実現しなかった。多段階分類には、また別の手法の検討が必要とされる。

5 章　食物摂取量計測システムのプロトタイプ

この章では、4 章で記述した食物摂取量計測システムの最終形態およびプロトタイプについて述べる。また、プロトタイプに搭載する NN による学習・識別システムである NN―SOFT の動作実験も行なう。

＜5.1＞ システム案

当初研究の発案段階では、病院の入院患者に出される食事の摂取量を検出するシステムを構築するというものであった。イメージするシステムの外観などは未決定であったが、その段階でおおまかなシステム構成は決まっていた。以下は研究初期における食物摂取量計測システムのシステム構成である。

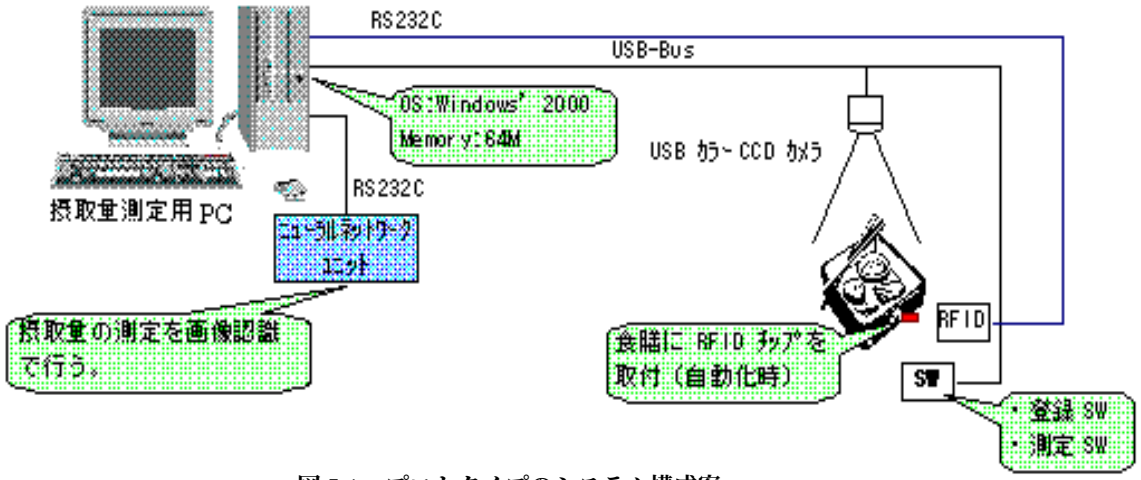


図 5.1　プロトタイプのシステム構成案

構成しステムの USB カメラ部

このシステム構成では、摂取量測定用の PC に、ニューラルネットワークユニットと画像採取用の USB カメラを、それぞれ RS232C と USB―BUS で接続したものを本体として、測定用 USB カメラと測定スイッチの操作のみで計測する、というものである。また、食事トレイ全体で識別するのではなく、食器ごとに識別を行う方式を提案していた。このとき、トレイには RFID チップを取りつけて、患者個人の食膳として自動判定するという案を採用する。

システム実用化を考えた場合、基本部分で満たさなければならない項目は、以下の通りである。

1．USB　I／F 対応のカラーCCD カメラの画像を取りこみ、食膳画像周囲の切出しをおこなう処理の開発。
2．トレイ上の食器の種類を自動判別する機能の開発（NN 処理）。
3．食器内の摂取量の測定アルゴリズムの開発と、食器内の摂取量を 20%程度の誤差で測定するアルゴリズムの開発。
4．測定用 NN の学習アルゴリズムとデータベース化処理の開発。
5．NN システムの VB6.0 への移植。

また、いくつかの制限事項を設ける必要もあり、食物摂取量計測システム１台につき画像採取用の USB カメラは１台であること、食膳（トレイ）の回転の補正は無しで固定とすること、食膳上の食器の位置は固定であることなどが挙げられる。

具体的な処理として、個人情報の登録では、対象患者の名前、性別、年齢、RFID コード、病名、体格（身長・体重・標準体重）、職業、治療職種１～５、単位が登録項目とされた。そして、個人データを格納するファイルの形式は、ファイルサイズ＝256 キロバイトとし、格納 ID 数で格納レコード数をしめす。各レコードは、個人情報を格納しており、256 キロバイトに固定されている。食膳の個人識別に使用される RFID チップは外注品で、仕様は、耐環境性、書き込み回数（書き込みは１回で問題ない）、書き込み容量（35 ビット）、アクセスタイム（0.5 秒程度）、読み取り可能距離（数 cm）、放射電界強度（病院内で使用するため、厚生省の基準を満足する必要がある）などの項目を想定している。

記録形式は、個人 ID、システムが設置されている場所を示す場所コード、システムが設置されている場所内の部署を示す部署コードの３項目で記録される。また、RFID の登録の流れは、まず RFID の出荷時点で書き込みを行う。書き込んだ RFID を食器業者に供給し、名札加工を施す。最後に、病院は、納入された RFID がついた名札に個人名をつけ、RFID コードを個人情報登録データベースファイルに登録する。図 5.2 は、RFID の登録の流れである。

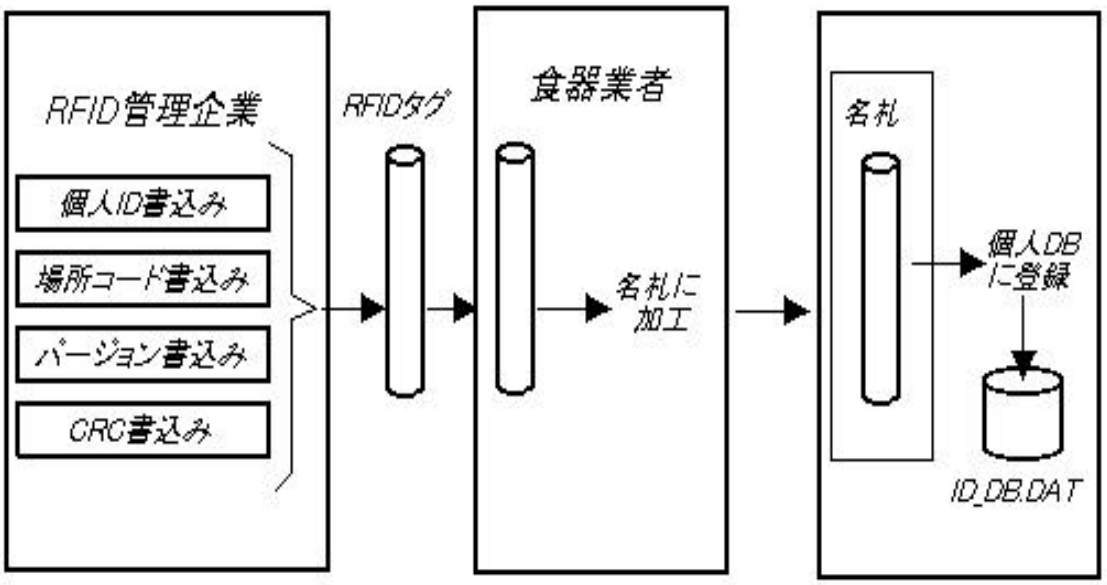


図 5.2　RFID 登録の流れ

食事摂取量検出システムで仕様されるニューラルネットワークの構造はニューロ識別ボードと同様の３層の階層型ネットワークとし、入力層の細胞数は最大 50 ユニット、中間層の細胞数は最大 50 ユニット、出力層の細胞数は最大 50 ユニットとなっている。学習方式には誤差逆伝搬（Back Propagation）アルゴリズムを使用し、(5.1) で算出される。W(k) i j(t)は、第ｋ－１層の第 i ユニットから第ｋ層の第ｊユニットへの結合重み係数、εは学習定数、αは慣性定数、βは振動定数である。また、ｔは学習のタイムステップである。

$$\Delta W(k)ij(t) = -\varepsilon \frac{\delta E(t)}{\delta W(k)ij(t)} + \alpha W(k)ij(t-1) + \beta W(k)ij(t-2) \qquad (5.1)$$

NEURO.CFG はシミュレータのものとほぼ同じもので、学習パターン数や１パターンの学習枚数をプログラム動作中に設定できる。食器データベースは、食膳上の食器を切出すために必要なデータベースである。同一食器であっても、食膳の画像でその内容物で差が判断できるものは別の食器として登録できるようなものとする。また、このデータベースは、食器の画像（切出すのに必要な対象画像)、摂取量を判断するためのレベルごとの画像、食器の名前、食物の名前、100%時の栄養素量（栄養素の

項目は未決定）を格納している。このデータベースのファイル形式であるが、食器の分解能は 64×64 ドット以内、画像ファイル上は 24 ビットフルカラーで保管しニューラルネットワークでの処理時に減色を行なうものとする。システムの運用を仮定したときの食前処理の流れは、図 5.3 のフローチャートで表される。

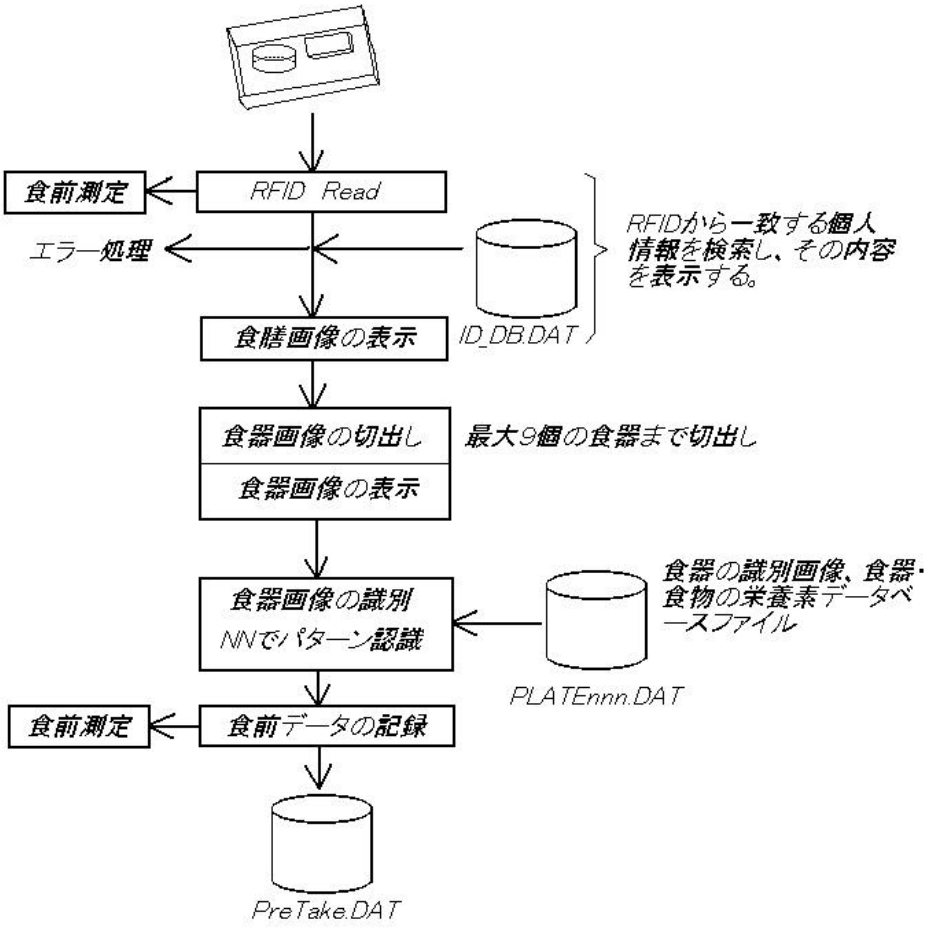


図 5.3　食前処理の流れ

図 5.1 の ID__DB.DAT は､RFID チップに登録された個人情報を格納するデータベースファイルである。PLATE.DAT 食器の識別に用いるデータベースファイルである。PraTake. DAT は、最終的な識別で参照する食前データのデータベースファイルである。また、食前処理の食器切出し処理で、切り出す食器の最大数が９となっているが、ひとつのトレイ上に 9 つ以上の食器が乗ることはないからである。これは、食物摂取量計測システムのプロトタイプ運用試験を実施する予定の病院内での調査で明らかになっている。図 5.4 に食後処理の流れを図示する。

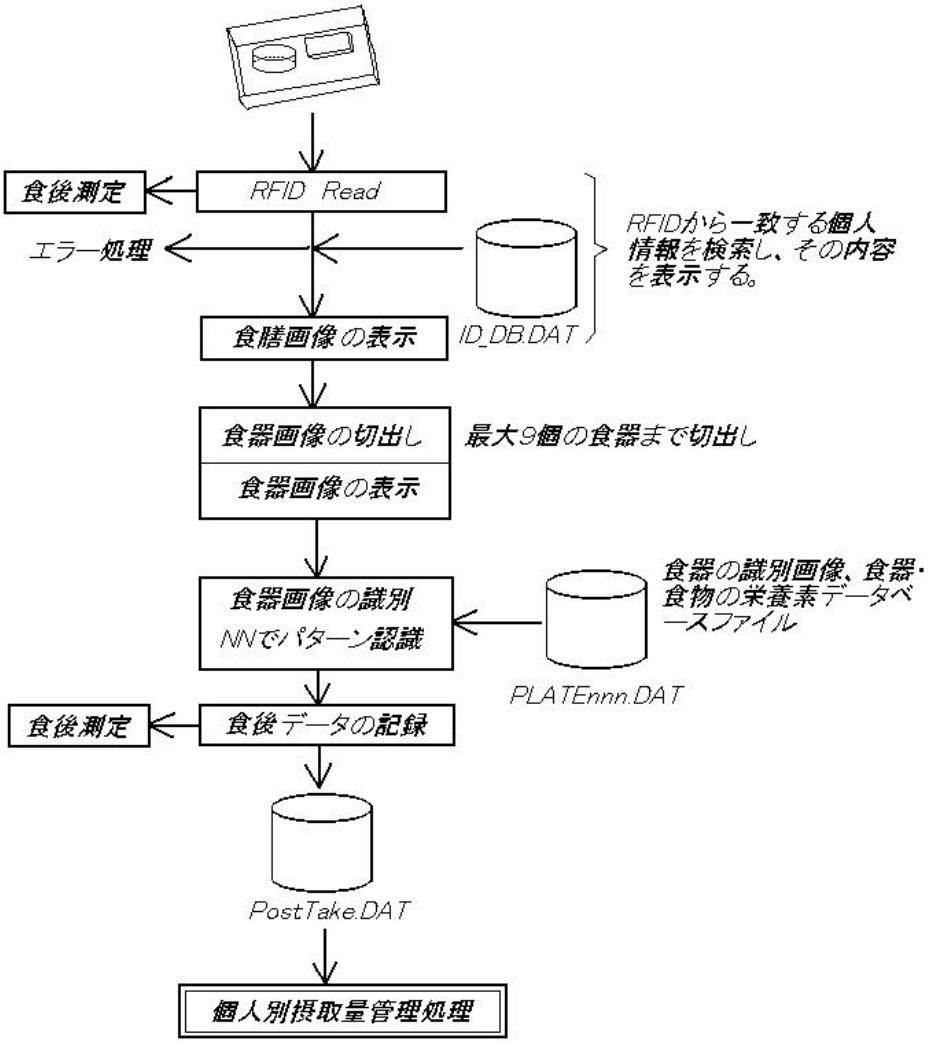


図 5.4　食後処理の流れ

次に、システムのソフトウェア構成を述べる。システムを統括するソフトウェアは、以下の機能ごとのプログラムで構成される。

１．個人情報プログラム

・個人情報（名前、病名、治療食種等）を登録、変更、印字を行う。

２．栄養素データベース作成プログラム

・病名ごと治療職種（栄養素）のデータベース作成する。

３．食器データベース登録プログラム

・食器と、その中にある食材を含めたデータベースを作成する。
・このデータベースにより食膳内の食器と食器内の食物を識別する。

4．摂取量自動測定プログラム

・画像処理により食事摂取量を自動で測定する。
・RFID タグの個人コードにより測定した摂取量で個人別摂取量データベースを更新する。

5．個人識別摂取量印字プログラム

・個人別の食事摂取量の来歴を印字する。

図 4.5 にソフトウェア構成を図示する。

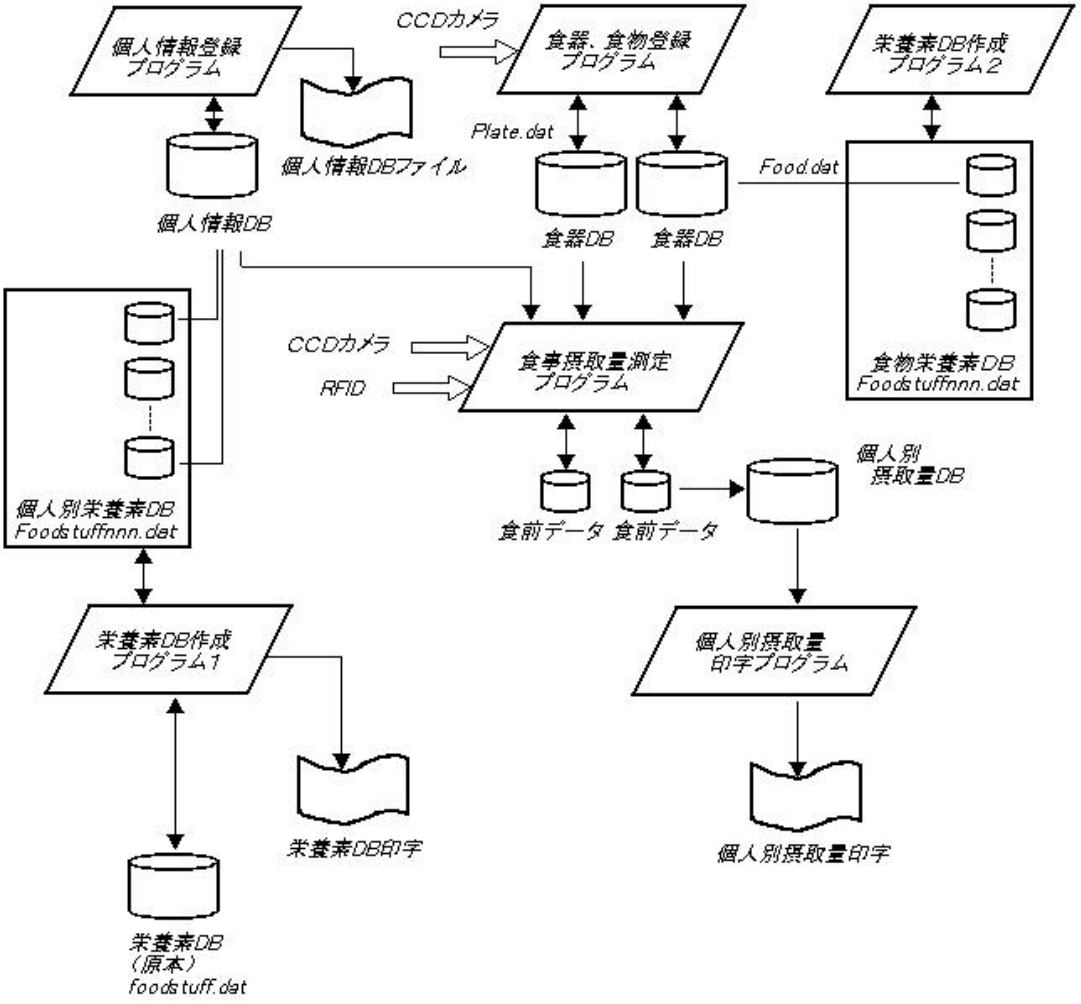


図 5.5　ソフトウェア構成

＜5.2＞ プロトタイプ用 NN シミュレータ

＜5.1＞で述べた食物摂取量計測システムの案のうち、システムの中枢と言える、NN を用いた学習・識別ソフトウェアについて述べる。

食物摂取量計測システムに搭載するソフトウェア（NN－SOFT）は、これまでの実験で使用した自律型ニューロ識別ボードとシミュレータプログラムの機能を引き継ぐものでありながら、食物摂取量の計測に適したシステムへと特化している。そのため、自律型ニューロ識別ボードやシミュレータプログラムにはなかった機能が追加され、表示項目も大幅に変わっている。

まず、NN－SOFT の概要を記述する。NN－SOFT は、食物摂取量計測システム最終バージョンに搭載するシステムの原型であるため、現状ではもっとも完成体に近い実行形式を持つ。専用撮像台にセットされた食膳から USB カメラで画像を採取し、ニューロフォーマットデータへの変換、マスクの設定、データベースの更新、学習データの抽出、スラブ値ファイルの作成、学習ファイル作成、学習、判定の各処理を画面上でユーザ側が選択できる。

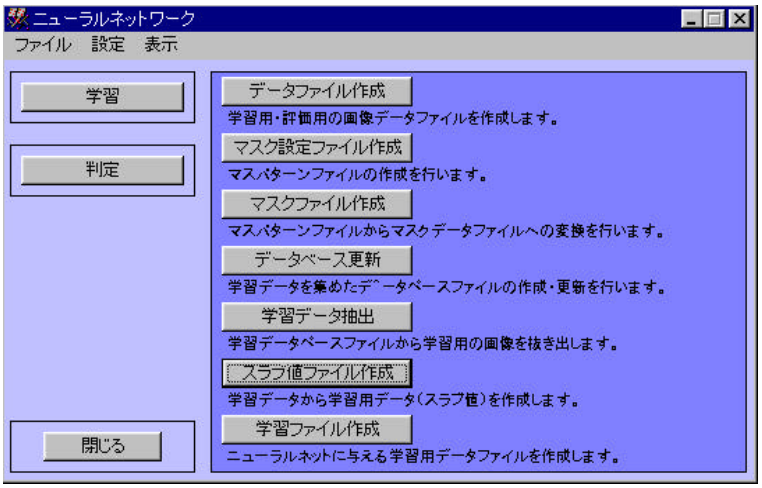


図 5.6　NN-soft によるメニュー画面

最初に、NEURO.CFG で学習設定をおこない、NN 構造ファイルと教師ファイルを更新したのち、データファイル作成のボタンを押す。ここでは、ニューロフォーマットデータの元となる BMP ファイルを選択し、データ作成ができる。作成したデータファイルには、後から画像を追加することも可能となっている。現状では、USB カメラから直接画像を取り込んでニューロフォーマットデータを作成する機能があるが、未完成である。

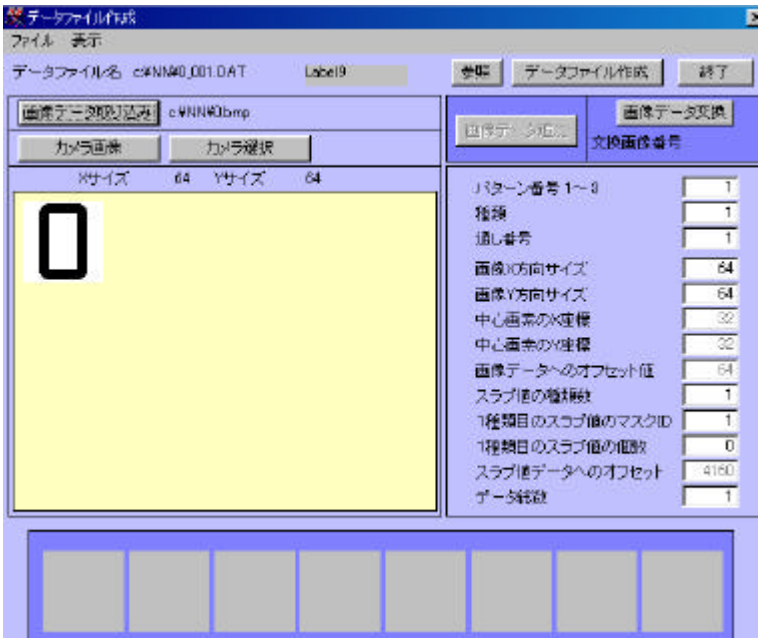


図 5.7　NN-soft によるデータ作成の例

次に、マスク設定ファイル作成を行なうのだが、NEURO.CFG での設定によって２種類のモードが存在する。ひとつは、手動でマスクを作成するモードで、そのモード時に限り、NN－SOFT のメニュー画面で、「マスク設定ファイル作成」「マスクファイル作成」が使用可能である。手動マスク作成モードでは、マスクの編集を行う。マウス操作でメッシュ状のマスクパターンを自由に作成し、任意のマスクパターン数を登録できる。また、全マスクや全スリット、反転などのマスク作成操作が、ボタンひとつで実行できるようになっており、編集した内容は、マスクパターン表示部に表示されるイメージ画像にリアルタイムで反映されるため、編集効率は高いものとなっている。そして、このときのマスクパターン数が NEURO.CFG での入力層数となる。

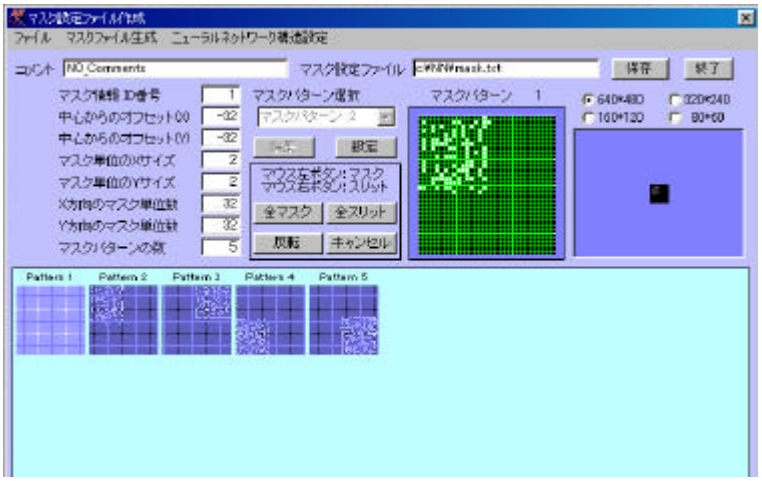


図 5.8　NN-soft による
マスク設定ファイル編集の例

マスクファイル作成では、マスク設定ファイル作成で作成したマスクパターンを、対象となる画像に重ね合わせて、位置ずれやマスクサイズの確認を行うことができる。ここで問題がなければ、マスクファイルを作成し、MASK.TXT が出力マスク情報ファイルである MASK.M に出力される。

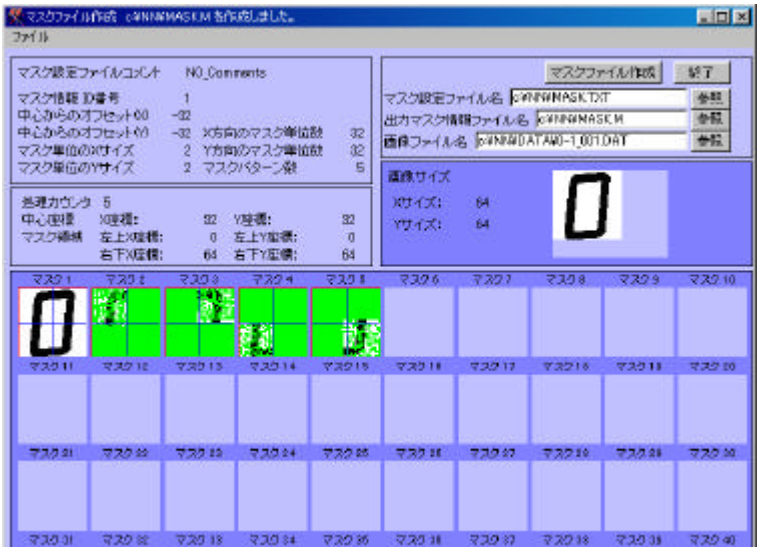


図 5.9　NN-soft 上でのマスクファイル作成の例

マスク設定ファイル作成とマスクファイル作成は、NEURO.CFG で手動マスク作成モードの場合にのみ実行できる。もうひとつのモードである、自動マスク作成モードでは、対象画像に対してメッシュをかけ、メッシュの合計値をスラブ値に反映させる手法を取っている。対象画像にかけたメッシュの最小単位（４ドット）の枠を縦／横／斜めの列で走査し、枠内の濃淡情報から１と０を判別、その合計値をスラブ値算出に反映するものである。そのため、このモードでは通常のグレイスケール画像ではなく、色反転処理をしたグレイスケール画像を使用しなければならない。現状では、その処理アルゴリズムから使用できる画像サイズは 64×64 ドットのものに限定されている。また、同様の理由で、自動マスク生成モードの場合、NEURO.CFG の設定項目である入力層細胞数・中間層細胞数は 46 に固定される。手動マスク作成モードと自動マスク生成モードの切り替えは、NEURO.CFG 内のスラブ作成モードの項目で１か０を選択することで決まり、１は自動マスク作成モード、０は手動マスク作成モードとなっている。

次に、データベース更新を行う。ここでの作業内容は、4 章でのシミュレータプログラムの実験時に記述した DATABASE.CFG の編集と同様のものであるが、こちらは選択式になっている。前述したデータファイル作成で作成した DAT ファイル（これまでの DB ファイル）を任意にデータベースファイルとして設定でき、更新ボタンを押すことで更新が完了する。また、再更新処理を行うときには、一度登録した DAT ファイルの削除・追加も自由に行うことができる。データベース更新画面は、データ

ファイル作成で作成した選択可能な DAT ファイルの一覧とディレクトリ、実際に追加される DAT ファイルの表示部で構成されている。

図 5.10　NN-soft によるデータベース作成の例

ここまでの設定が完了したら、学習データを抽出する。この処理で、学習用データベースファイルから学習用の画像を抜き出すことができる。学習データ抽出のボタンを押すと抽出処理がはじまり、データベース内の画像が表示部に表示されていく。画像が大きな場合、表示に時間がかかるため、チェックボックスの画像表示の有無を画像表示無しにすればよい。次に、スラブ値ファイルを作成する。ここでは、学習データから学習データ（スラブ値）を作成する。手動マスク作成モードの場合は、マスク後の濃淡データを加算してスラブ値としている。操作手順は、使用するマスクファイル名を入力し、作成ボタンを押すだけである。自動マスク生成モードでは、縦／横／斜め方向の濃淡積加算を行い、スラブ値を作成する。また、閾値も任意に設定でき、NEUORO.CFG 内で編集できる。このモード時にはマスクファイル名の入力は必要ない。スラブ値作成画面は、現在のモードを表すスラブ作成モードをのチェックボックス、学習画像の表示部、マスクによる対象画像の被覆画面表示部、スラブ値の表示、生スラブ値のグラフ表示部で構成される。

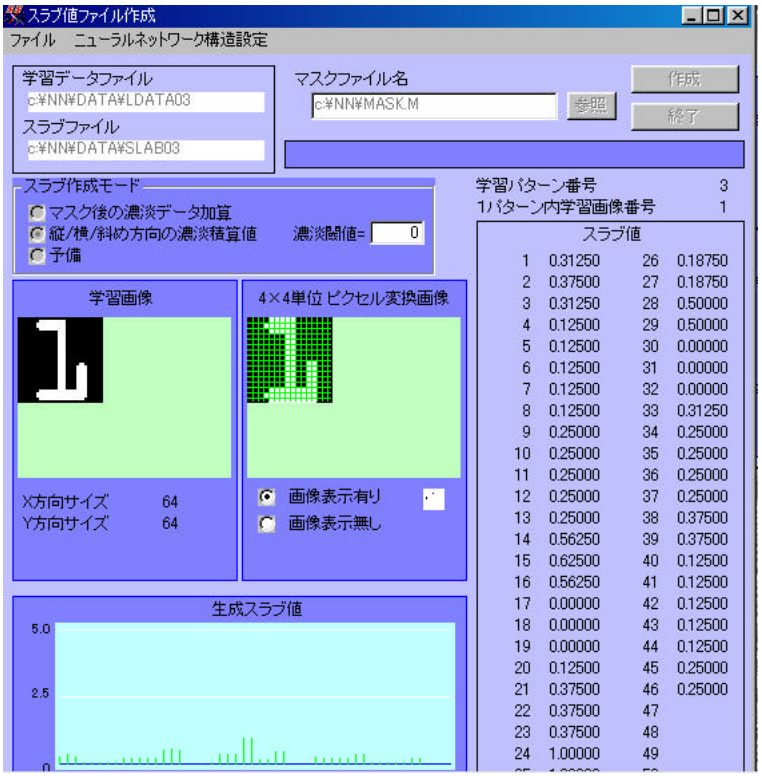


図 5.11　NN-soft によるスラブ値ファイル作成の例

学習・判定までの最後の準備として、学習ファイル作成を行なう。ここでは、ニューラルネットワークに与える学習用データファイルの作成を行なう。学習ファイル作成画面は、学習パターン数、1 パターンの学習枚数、読み込みスラブファイル名、処理学習パターン数、学習数カウント、スラブ数、スラブ作成モードなどの各情報の表示と、出力されるスラブ値がグラフ表示される表示部で構成されている。操作手順は、作成学習ファイル名（LEARN.S）を入力した後、作成ボタンを押す。グラフが表示しきれない場合は、グラフスケールのチェックボックスを Max5.0 に変更すれば良い。以下は学習ファイル作成画面である。

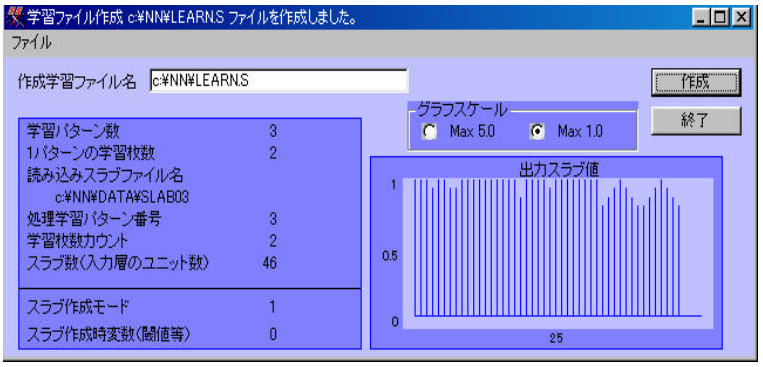


図 5.12　NN-soft による学習ファイル作成の例

最後に、学習を行なう。ここでは、これまでの手順で作成したデータファイルを使用し、ニューラルネットワークに学習をさせる。実際の操作手順として、最初に学習モードの選択しなければならない。通常は初期学習を選択するが、続けて学習させたい場合には継続学習を選択できる。次に、学習に使用する学習スラブ値ファイルと教師ファイルを選択する。通常、学習スラブ値ファイルはスラブ値ファイル作成で作成したものを、教師ファイルは NEURO.CFG の編集時に作成する教師ファイルを使う。出力ウェイトファイルは、NEURO.CFG の内容の一部と、振動数や慣性定数、総合誤差、温度勾配などの項目があり、現時点では直接ウェイトファイルを編集する機能がないため、デフォルト設定のものを使っている。

ここまでの設定が完了したら開始ボタンを押すと自動的に学習がはじまる。学習時には、学習回数がカウントされ（上限は NEURO.CFG で設定した最大学習回数まで）、誤差値が収束範囲内に収まるまで繰り返し学習が行われる。同時に、誤差値と学習回数を X 軸・Y 軸に持つグラフと、入力層の入力値グラフ、出力層の出力値グラフを表示する。図 5.13 は学習時の画面の例である。

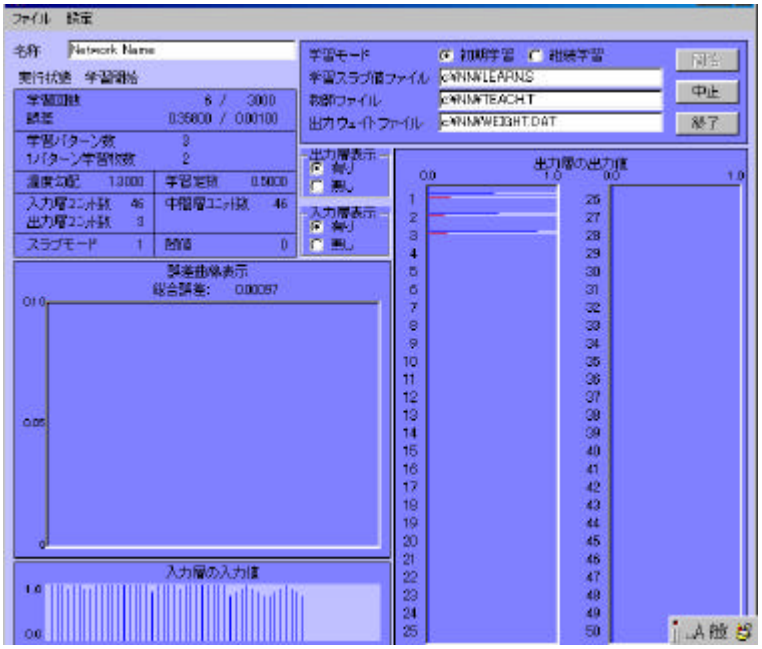


図 5.13　NN-soft による学習の例

学習が終了したら、判定を行なう。判定には、データファイル作成で学習用ＤＡＴファイルを作成したのと同様の手順で、判定用の未学習データを作成しておかなければならない。ここでの操作手順は、認識画像ファイルの選択と判定実行のみである。画面上には認識宇画像ファイル、ウェイトファイル、ＮＮ定義ファイル、マスクファイルの選択表示部があるが、実際に選択できるのは、手動マスク作成モードでは認識画像ファイルとマスクファイル、自動マスク生成モードの場合は認識画像ファイルのみとなっている。判定を実行すると、判定に使用された認識画像が表示され、入力ス

ラブ値がグラフで表される。出力層の出力値には、各パターンごとの発火値が表示され、認識画像は、もっとも発火値の高いパターンに判定される。

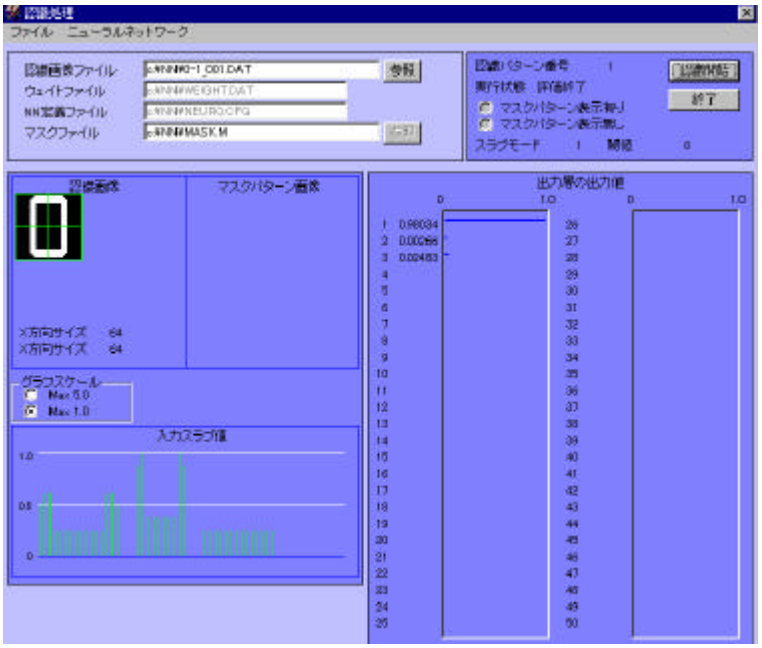


図 5.14　NN-soft による判定の例

＜5.3＞　NN－SOFT による 2 階調画像の識別

ここでは、NN－SOFT の動作試験用に作成したデータを使用して、学習・識別実験を実施する。また、手動マスク作成モードと自動マスク生成モードで識別率にどのような差が生じるのかも検討する。

5.3.1 採取画像

これまで使用した画像は、サンプルとして提供された写真から採取したビットマップデータや、食物ダミーを元にしたビットマップデータで、いずれも食物に関連したものであった。しかし、ここでは、食物とは関連のない画像データを使用する。これは、もともと C プログラムであったシミュレータプログラムを VB に書き換えた NN－SOFT が、正常に機能するかを確認するためである。

使用する画像は、１～９の各数字をビットマップ画像として作成したもので、色情報は白と黒の２値しかない。１つの数字画像のサイズは、自動マスク生成モードに合わせて 64×64 ドットとする。手動マスク作成モードで使用するビットマップ画像は、数字部分を黒、背景を白としている。自動マスク生成モードで使用する画像は、手動

マスク作成モードで使用するビットマップ画像に色反転処理を施したものを使用する。図 5.15、図 5.16 に学習に使用する数字画像の例を掲載する。

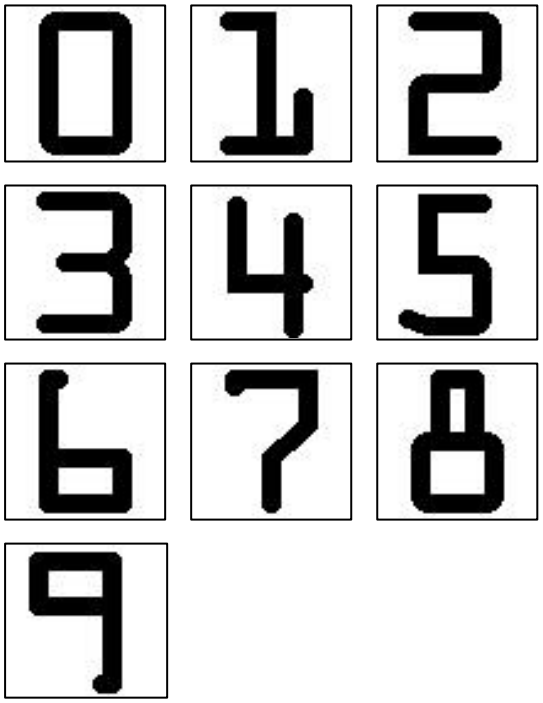


図 5.15　手動マスク作成モードに使用する学習画像の例

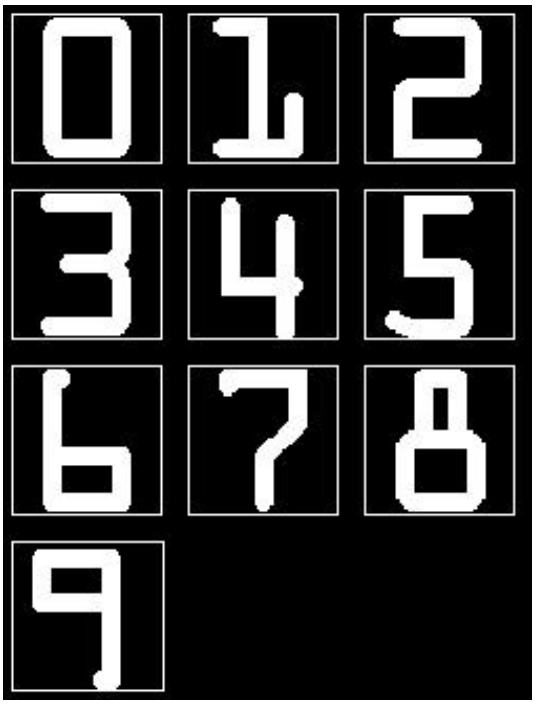


図 5.16　自動マスク生成モードで使用する学習画像の例

次に、これらの画像のうち、「０」「１」「２」を学習画像として使用し、各２枚ずつをニューラルネットワークに学習させる。識別には、これら学習画像に変化をつけたものを使用する。この変化をつけた画像の識別で、ニューロ識別ボードでは対応できた、対象物（例えば紙幣）の回転、位置ずれなどの幾何学的挙動に、この NN‐SOFT が対応できるのかを確認する。

具体的な識別用画像の差違は以下のとおりである。

１．学習画像
２．学習画像を水平方向に＋15 度傾斜後、サイズを 64×64 に調整
３．学習画像を水平方向に‐15 度傾斜後、サイズを 64×64 に調整
４．学習画像を垂直方向に＋15 度傾斜後、サイズを 64×64 に調整
５．垂直方向に‐15 度傾斜後、サイズを
　64×64 に調整

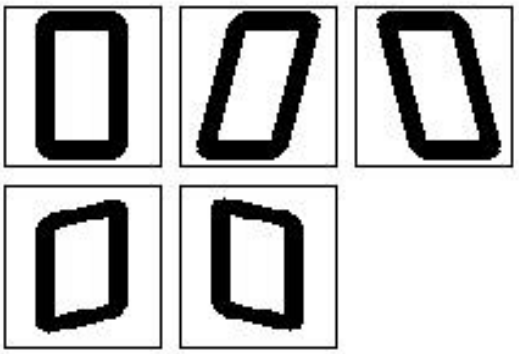


図 5.17　手動マスク作成モードでの識別画像

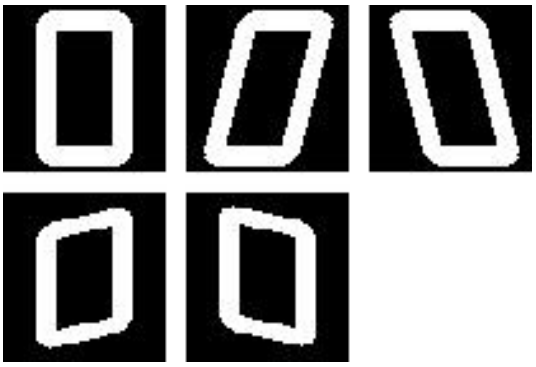


図 5.18　自動マスク生成モードでの識別画像

5.3.2 数字画像での識別結果

このような画像を識別に作成し、NN－SOFT で認識をおこなった。学習には図 4.15、図 5.16 に示した数字画像のうち「0」「1」「2」を使用する。識別には、学習した数字画像に加えて、傾斜処理によって変化を与えた 4 枚の画像を使用した。表中左列の識別画像番号は、‐1 が習画像、‐2 が平方向＋15 度、‐3 が平方向－15 度、‐4 が直方向＋15 度、‐5 が垂直方向－15 度の傾斜画像を表す。

表 5.1　自動マスク作成モード識別結果

数字画像：0

識別画像	発火率	識別
0-1	0.97728	0
0-2	0.49547	0
0-3	0.56789	0
0-4	0.95587	0
0-5	0.96838	0

表 5.2　自動マスク作成モード識別結果

数字画像：1

識別画像	発火率	識別
1-1	0.99571	1
1-2	0.97849	1
1-3	0.98593	1
1-4	0.9099	1
1-5	0.98813	1

表 5.3　自動マスク作成モード識別結果

数字画像：2

識別画像	発火率	識別
2-1	0.97036	2
2-2	0.9749	2
2-3	0.98735	2
2-4	0.46477	2
2-5	0.65235	2

表 5.4　手動マスク生成モード識別結果

数字画像：0

識別画像	発火率	識別
0-1	0.96364	0
0-2	0.19544	2
0-3	0.95634	0
0-4	0.62505	0
0-5	0.95053	0

表 5.5　手動マスク生成モード識別結果

数字画像：1

識別画像	発火率	識別
1-1	0.99672	0
1-2	0.08494	2
1-3	0.99894	0
1-4	0.98449	0
1-5	0.95053	0

表 5.6　手動マスク生成モード識別結果

数字画像：2

識別画像	発火率	識別
2-1	0.99488	2
2-2	0.9998	2
2-3	0.64291	0
2-4	0.99697	2
2-5	0.96462	0

手動マスク作成モードでは、数字画像「0」に対して識別率＝80%、数字画像「1」に対して識別率＝80%、数字画像「2」に対して識別率＝40%であった。自動マスク生成モードでは、数字画像「0」に対して識別率＝100%、数字画像「1」に対して識別率＝100%、数字画像「2」に対して識別率＝100%であった。この結果では、手動マスク作成モードよりも自動マスク生成モードのほうは識別率が高く、より有効であるといえる。しかし、マスク作成モードで作成したマスクが最適なものでない場合が考えられるため、一概に自動マスク生成モードのほうが優れているとは断言できない。また、1この実験で用いた識別画像が、コンピュータがもっとも扱い易い2値画像を用いたことから、食物画像などの多階調グレイスケール画像の場合も検討してみる必要がある[7]。だが少なくとも、このレベルでの画像の識別には、対象画像の変形や回転にある程度対応できることは実験によって示された。この結果から、正しく認識する能力があることが実証できたといえる。

＜5.4＞　NN-SOFTによるグレイスケール画像識別実験

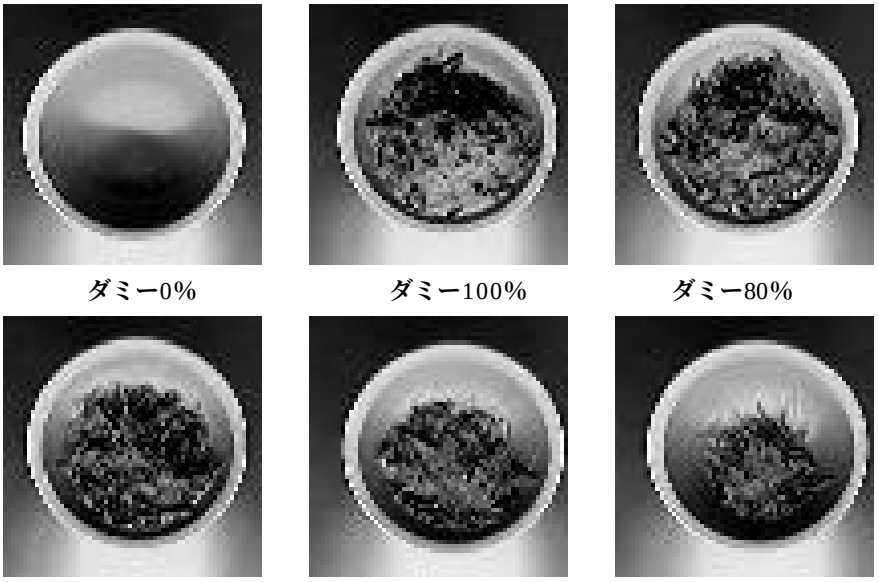
ここまで、NN-SOFTの基本性能の試験を行ってきた。使用画像も単純な2階調のものを使用し、識別結果を分析した。その結果、2階調画像では識別は正常にできること、手動マスク作成モードと自動マスク生成モードでは識別率が異なることなどが実証された。

そこで、次に、プロトタイプで使用することになると思われる、グレイスケールの食事画像（ここでは食物ダミーを使用）で識別実験を行なう。

5.4.1 採取画像

使用する画像は、シミュレータプログラムでの実験時に使用した食事ダミーのごはんの画像を使用し、NEURO.CFG の設定を学習パターン数＝6、各パターンの学習枚数＝2として実験する。また、マスクモードは自動マスク生成モードに固定する。

そのため、シミュレータプログラムでの実験で用いた画像をそのまま使用せず、NN‐SOFT での制限に従い、サイズは 64×64 に修正し、色反転処理を施した。図 5.19 は実験に用いた食事ダミー（ごはん）である。



ダミー0%　ダミー100%　ダミー80%

ダミー60%　ダミー40%　ダミー20%

図 5.19　学習・識別画像の例食事ダミーごはん（100%画像～0%画像）

5.4.2 食物ダミーの画像による識別結果

5.4.1 で提示したような画像を学習に用いて、識別を行なう。評価は、100%～0% まで 20%刻みに減量させた画像で 6 段階、各段階ごとに学習画像を含めた 5 パターン、計 30 枚で実施した。表 5.7～表 5.12 はその識別結果である。

表 5.7　NN-SOFT による識別結果（100%画像）

識別画像	発火率	識別	判定
100%_1	0.99885	100%	○
100%_2	0.1267	60%	×
100%_3	0.68537	100%	○
100%_4	0.48859	80%	×
100%_5	0.41802	80%	×

表 5.8　NN-SOFT による識別結果（80%画像）

識別画像	発火率	識別	判定
80%_1	0.99966	80%	○
80%_2	0.9399	80%	○
80%_3	0.008	100%	×
80%_4	0.27211	100%	×
80%_5	0.41802	100%	×

表 5.9　NN-SOFT による識別結果（60%画像）

識別画像	発火率	識別	判定
60%_1	0.99984	60%	○
60%_2	0.94218	60%	○
60%_3	0.05106	20%	×
60%_4	0.36708	60%	○
60%_5	0.9966	60%	○

表 5.10　NN-SOFT による識別結果（40%）

識別画像	発火率	識別	判定
40%_1	0.99966	40%	○
40%_2	0.00009	20%	×
40%_3	0.00255	20%	×
40%_4	0.00052	20%	×
40%_5	0	20%	×

表 5.11　NN-SOFT による識別結果（20%）

識別画像	発火率	識別	判定
20%_1	0.99889	20%	○
20%_2	0.9138	20%	○
20%_3	0.16173	20%	○
20%_4	0.38886	0%	×
20%_5	0.65737	20%	○

表 5.12　NN-SOFT による識別結果（0%）

識別画像	発火率	識別	判定
0%_1	0.99989	0%	○
0%_1	0.95587	0%	○
0%_1	0.98384	0%	○
0%_1	0.9884	0%	○
0%_1	0.89445	0%	○

以上の結果では、100%画像では識別率＝40%、80%画像では識別率＝40%、60％画像では識別率＝80%、40%画像では識別率＝20%、20%画像では識別率＝100%であった。この結果は、シミュレータプログラムでの実験結果に比べると、非常に識別率が低く、誤認識していることがわかる。シミュレータプログラムでの識別実験では、完璧に近い識別率を示したが、同じ画像を使用したにも関わらず識別率が低い。その原因としては、マスク処理の問題が考えられる。＜5.2＞で識別率が高かった自動マスク生成モードを採用したが、このモードは入力画面全体に処理が及んでしまうため、入力画像の余白までスラブ値に反映されてしまう。その結果、レベルの異なる画像でも、余白部分が値として加算され、スラブ値が近似してしまい、学習が上手くできなかったと推測される。また、この実験で学習・識別に用いた画像はグレイスケール画像であるため、スラブ値作成時の濃度積加算をコントロールする閾値にも問題があると予測される。

そこで、以上の考察を踏まえて新たな実験をおこなう。使用する画像をさらに修正し、余白を限りなく削減したものを使用する。また、濃淡積加算で閾値を変えて実験を実施する。

5.4.3 採取画像としきい値の変化による差違

5.4.1、5.4.2 では、画像は３章のシミュレータプログラムでの実験で用いたものをそのまま使用した（NN－SOFT では色階調は 24 ビットフルカラー）。そのため、非常に識別率の低い結果となってしまった。そこで、5.4.2 での考察を踏まえた実験

を行う。

まず、使用する画像の余白部分を限りなく削除する。切出しソフト（後述する）による食器切出し処理を行った画像と同じ状態に修正する。以下は修正誤の食物ダミーの画像である。

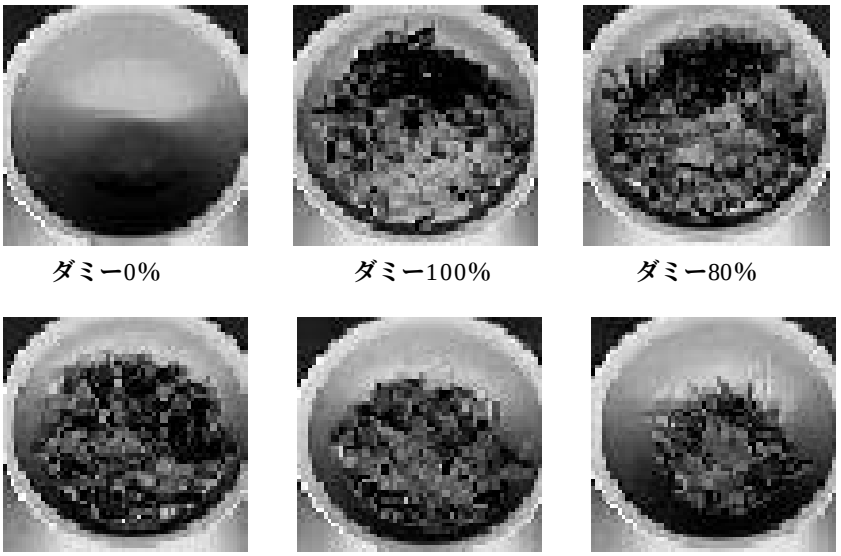


図 5.20　食物ダミー・ごはん／余白カット（100%~0%）

入力画像については、図 5.20 のものとする。また、スラブ値作成にあたって、閾値を変化させる。これまでの自動マスク作成モードでは、閾値は常にデフォルト設定の０でおこなってきた。ところが、このままではスラブ値を作成した場合にどのデータも近似してしまい、食物ダミーの減量がデータとして反映されにくくなっていることがわかった。そこで、スラブ値作成時に、閾値をデフォルト値でスラブを作成した場合と、閾値を段階的に変化させていき、適切であると思われる閾値に設定しなおしてスラブを作成した場合の違いを記述する。図 5.21、図 5.22 は閾値の違いによるスラブの差違を、NN－SOFT のスラブ作成画面で掲載したものである。

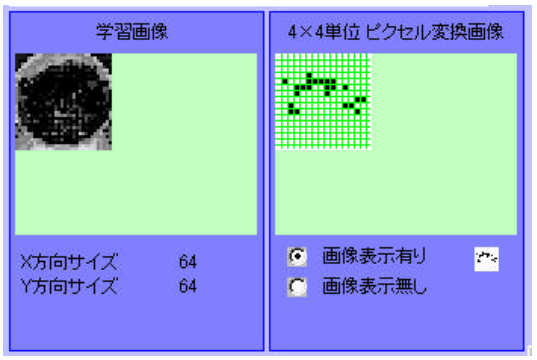


図 5.21　スラブ作成画面：閾値＝0

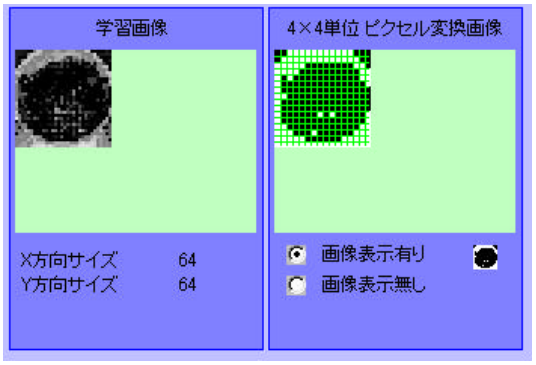


図 5.22　スラブ作成画面：閾値＝60

これは、どちらも食物ダミー100%の画像でスラブ値を作成した画面の一部である。図右側で、黒くなっている部分が食器内のごはんの量を表している。閾値が0である図は、食器の内容量が 100%であるにもかかわらず、黒い部分がほとんどない。逆に、閾値が 60 である図 5.22 は、その内容量が画像的にも反映されている。

5.4.4 食物ダミー・切出し画像による学習識別

学習には、図 5.18 に示した食物ダミーの画像を使用する。食器に外接する四角形に画像をカットし、サイズを 64×64 に調整したものである。NEURO.CFG の設定は 5.4.1、5.4.2 の実験と同様に、学習パターン数＝6、各パターンの学習枚数＝2とし、新たに閾値＝を設定した。マスクの作成モードは、引き続き自動マスク生成モードで実験をおこなった。表 5.13～表 5.18 は食物ダミーでの識別結果である。

表 5.13　食物ダミー100%画像の識別結果

識別画像	発火率	識別	判定
100%_1	0.99795	100%	○
100%_2	0.98714	100%	○
100%_3	0.89312	100%	○
100%_4	0.97783	100%	○
100%_5	0.96758	100%	○

表 5.14　食物ダミー80%画像の識別結果

識別画像	発火率	識別	判定
80%_1	0.9937	80%	○
80%_2	0.40239	80%	○
80%_3	0.5066	80%	○
80%_4	0.27981	80%	○
80%_5	0.21103	80%	○

表 5.15　食物ダミー60%画像の識別結果

識別画像	発火率	識別	判定
60%_1	0.98552	60%	○
60%_2	0.19732	60%	○
60%_3	0.00003	0%	×
60%_4	0.55408	60%	○
60%_5	0.02428	40%	×

表 5.16　食物ダミー40%画像の識別結果

識別画像	発火率	識別	判定
40%_1	0.9616	40%	○
40%_2	0.37064	40%	○
40%_3	0.3439	40%	○
40%_4	0.13786	60%	×
40%_5	0.22622	20%	×

表 5.17　食物ダミー20%画像の識別結果

識別画像	発火率	識別	判定
20%_1	0.9709	20%	○
20%_2	0.86897	20%	○
20%_3	0.88988	20%	○
20%_4	0.79062	20%	○
20%_5	0.95038	20%	○

表 5.18　食物ダミー0%画像の識別結果

識別画像	発火率	識別	判定
0%_1	0.99451	0%	○
0%_1	0.9982	0%	○
0%_1	0.98384	0%	○
0%_1	0.94333	0%	○
0%_1	0.89445	0%	○

この実験結果では、100%画像の識別では識別率＝100%、80%画像の識別では識別率＝100%、60%画像の識別では識別率＝60%、40%画像の識別では識別率＝60%、20%画像の識別では識別率＝100%、0%画像の識別では識別率＝100%であった。総合的な識別率は、約 87%であった。(1)(2)の実験結果と比較しても、識別率の向上はあきらかである。

この結果から、NN－SOFT の自動マスク生成モードでは、入力画像を対象の食器のみを切出したものとし、対象食物の色に応じて適切な閾値を設定することにより識別は十分に行なえると言える。また、さらなる識別率の向上のためには、使用画像を採取する段階で食物の影ができないようにするなど、入力画像を明確に撮影しデータ化する工夫も必要である。

＜5.5＞ 画像切出しの手法

＜5.1＞でも述べたように、食物摂取量計測システムでは、画像処理で、患者に配膳された食事の乗ったトレイから皿単位で切出し、次に皿上の食物を切出すという２段階の切出し処理を行うものとしている。この切出しの処理は、切出しプログラム（以降 CUTOUT と呼称する）で処理をする。ここでは、CUTOUT の概要について記述し、実際に画像を切出す実験を行なう。

5.5.1　CUTOUT のアルゴリズム

プロトタイプに搭載予定の切出し処理をおこなう CUTOUT は、C 言語で書かれたプログラムである。CUTOUT は、256 階調のグレイスケール画像から、対象物に外接する長方形の枠をかけ、その中心座標を算出するものである。

CUTOUT では、入力画像を閾値処理によって２値画像に変換する[(6)(7)]。画像の切出しの処理手法として収縮処理、ラベリング処理、8 近傍処理を採用し、切出しを実現している[(6)(7)]。８近傍処理は、画像全体をドット単位で走査して対象物があればその座標に“１”をつけ、なければ“０”をつける。“１”があればその座標を中心にして８方向をさらに走査し、次々に“１”を探していく。最終的に“１”の塊ができると、それがひとつの対象物の形をなし、ここで“１”“０”をつける基準は、対象の色情報である。CUTOUT では、RGB のうち、B の色情報を基準とし、その基準は閾値によって制御される。ラベル処理は、8 傍処理でできた塊ごとにラベルを張りつける処理である。これによって、塊ごとに独立した対象物であると区別できるようにしている。収縮処理は“1”“0”で分けた塊を文字通り収縮させる処理である。１と０の境界部分を少しずつ削除して、塊を際立たせることができる。これによって、8 近傍処理でひとつの塊として認識された２つの対象物を分離することが可能となっ

ている。これらの処理手法を組み合わせて、画像から対象物を切出すプログラムが、CUTOUT である。

PC 上の具体的な操作は、CUTOUT の起動後、処理したい画像ファイル名を入力し、収縮処理回数、切出し領域の X、Y それぞれの上限値と下限値を入力するだけである。これらを設定することでかなりのノイズを除去することができ、望ましい画像を採取できる。図 5.23～図 5.25 は、条件の違いによる切出し画像の違いを掲載したものである。

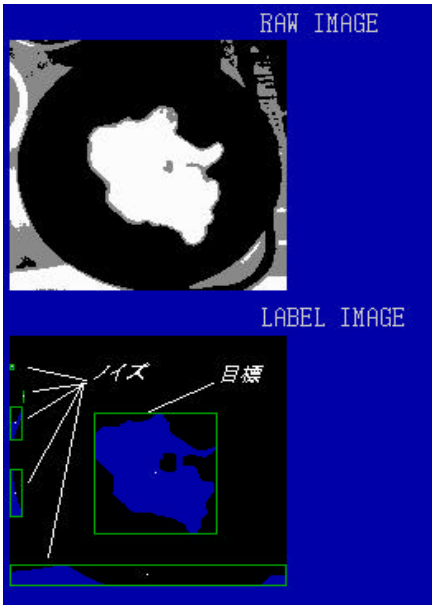


図 5.23　CUTOUT による切出し画面 1
（失敗例）

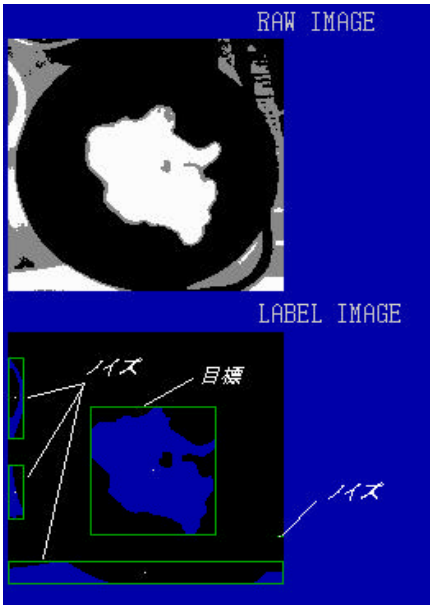


図 5.24　CUTOUT による切出し画面 2
（失敗例）

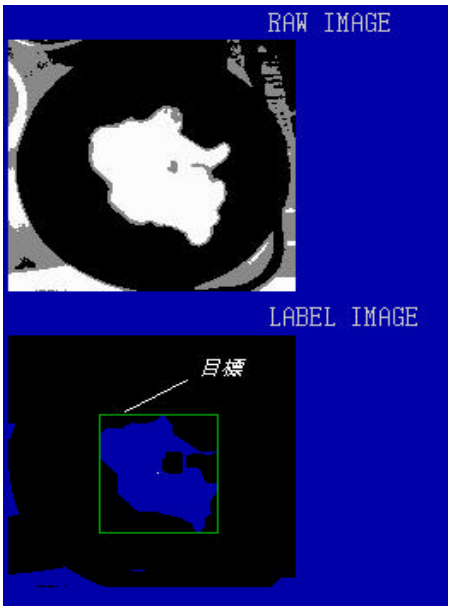


図 5.25　CUTOUT による切出し画面 3
（成功例）

使用した画像は、4 章での実験に用いた食事の写真から採取した画像データで、この図では、上半分が入力画像、下半分は切出した領域をあらわしている。入力画像は色反転処理を施したものだが、これは、現状では色判定処理の部分で白に近い色に“１”をつけるようになっているからである。図 5.21 は、収縮回数＝１、切出し領域制限なしで処理したものである。食器内の食物は正確に切出しているが、食器の影にまで切出しがおよび、ノイズが多い切出し結果となっている。図 5.22 は、収縮回数＝２、切出し領域制限なしで処理したものである。収縮回数を増やしたことで、小さいノイズは完全に消滅している。図 5.23 は、収縮回数＝２、切出し領域制限を X の上限＝200、X の下限＝20、Y の上限＝200、Y の下限＝20 で処理したものである。ノイズも完全に排除し、食器上の食物だけを切出すことに成功している。これまでの実験で、CUTOUT は実用段階に入っているといえるが、まだ問題も多い。現時点では、入力画像の色階調の具合によっては、収縮処理や切出し領域制限では対応できないことがある。例えば、食材と食材の影の色が近い場合、両方を 1 個の物体として切出しをしてしまう、といった事態も起こっている。また、食材表面に大きな凹凸が多いと、凹凸ごとに切出してしまう現象も起こっている。プログラムを改定して、8 近傍処理時に使われる色判定の閾値を調節したが、逆に不要な部分にラベリングが行われ、解決には至らなかった。解決には、入力画像に何らかの処理を施すか、色情報だけでなく別の情報も判定基準に取り入れるなど、工夫が必要である。

＜5.6＞プロトタイプの筐体

これまでの実験は、ニューロ識別ボード、PC 上でのシミュレータプログラム、専用撮像装置での実験と、きわめて部分的な実験を行なってきた。特に、専用撮像装置はすべて手作りであり、材質や強度などに多くの問題を抱えたまま実験に投入している。そこで、現在もっとも完成形態に近い、専用撮像装置と USB カメラ、ノート PC の実験システムを、プロトタイプ化する。その際、専用撮像装置を外部に発注し、スチール製の強固で安定した筐体を製作する。

1. スチール製であること。
2. 食事のトレイ全体を画像として採取できること。
3. ノート PC を設置できる棚を付ける。
4. 照明は 4 方向で、ミニクリア灯を使用候補とする。
5. 運用試験時には、移動式のワゴンに設置するため、重さ等は考慮しなくてよいが、PC を支える強度を持つこと。

以上の点を考慮にいれて、図面をおこし、プロトタイプに搭載する USB カメラのサイズ等、必要と思われる図面も作成した。以後デザインや、細部の方式に多少の変更があると思われるが、図 5.26、図 5.27、図 5.28 に現時点のものを提示しておく。

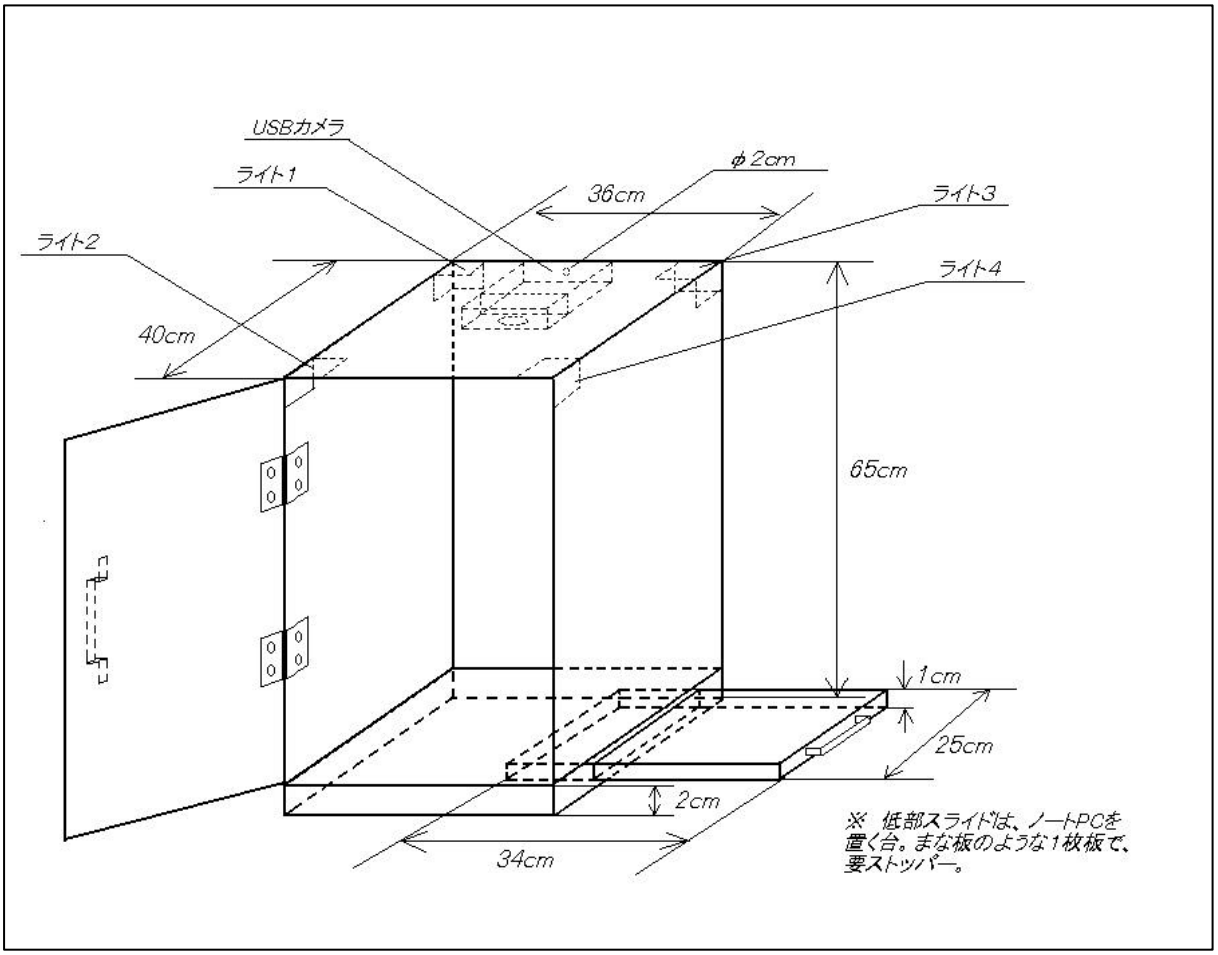


図 5.25　プロトタイプ筐体設計図

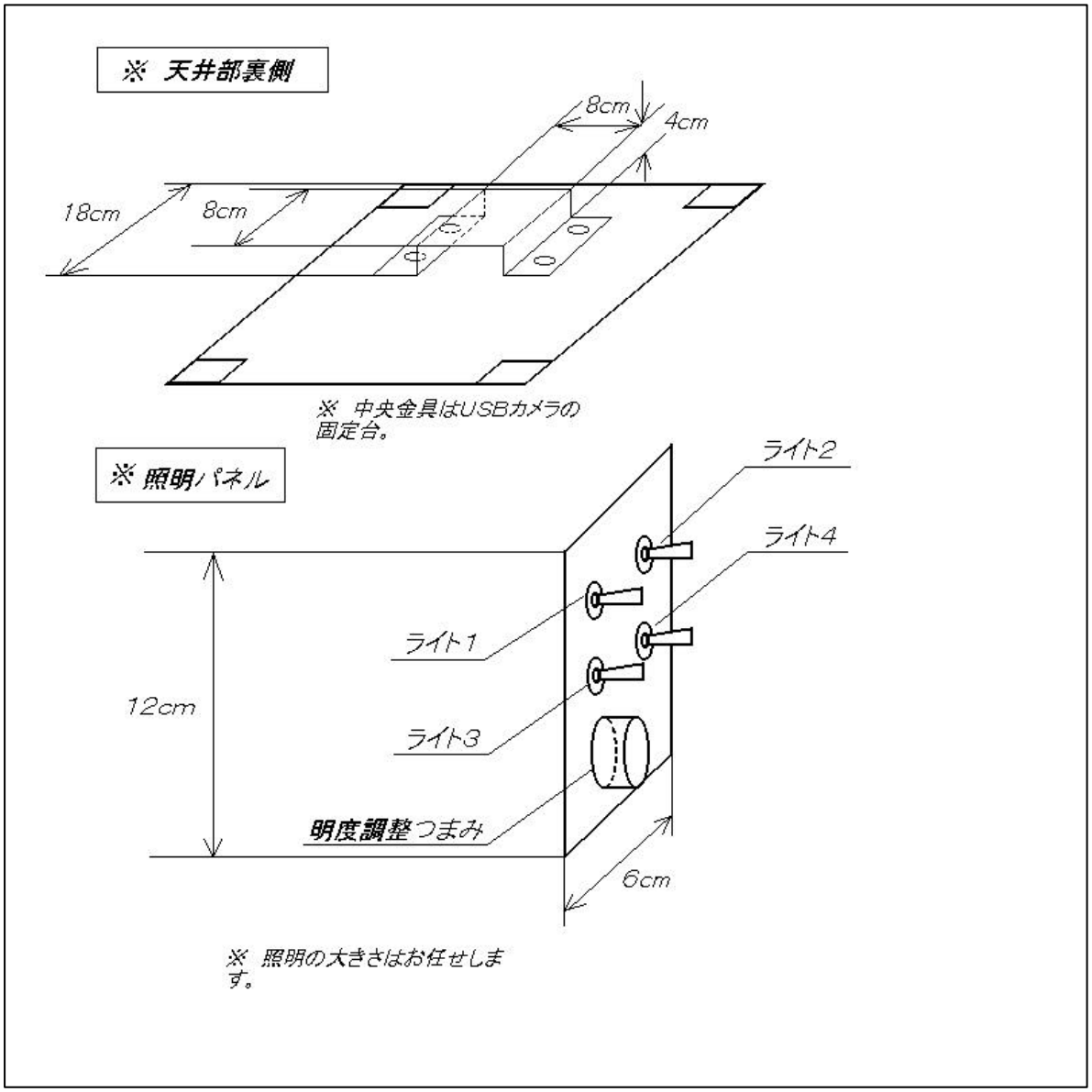


図 5.26　プロトタイプ筐体天井部／スッチ部設計図

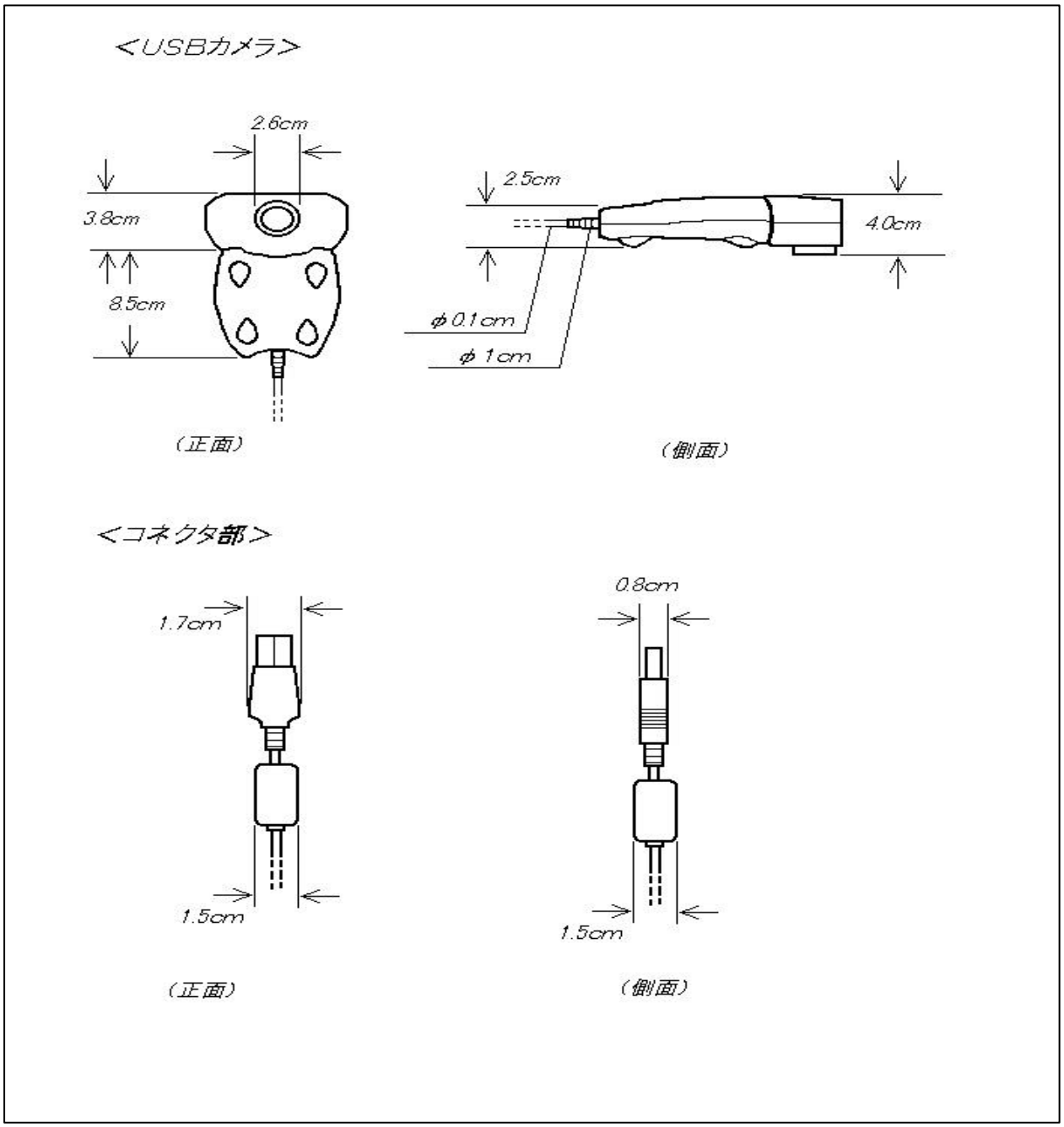


図 5.27　プロトタイプ搭載の USB カメラ

6章　まとめ

本論文では、病院内の入院患者の食事を対象とした、ニューラルネットワークを応用した食物摂取量を計測するシステムの研究を行なった。そして、紙幣識別用自律型ニューロ識別ボードの実験に始まり、食物摂取量計測に特化したNN－SOFT の提案を行ない、その基本性能を食物画像で確認した。また、識別システムだけでなく、システムに付随する画像切出しの手法では、問題点を残しながらも、ラベル数の増加や切出し領域の制限、収縮回数設定などの機能を追加し、その性能を向上させた。実用段階で必要となる画像採取用の専用装置の試作、設計も行った。この時点でも高い識別能力があることが実験によって示された。

しかしながら、実験では最大6レベルの識別分類実験を行い、高い識別結果を出したが、現実では食物の減量パターン数は無限であり、そのレベルを分類するにあたって画像単位で差違を判別できるのかという問題がある。また、固形の食物でなく、スープなどの液体は、上から撮像した画像では減量がわかりにくく、ニューラルネットワークでその差違が判別するのは困難であることなどの問題が残されている。

現在、無限レベルの分類問題については、その解決手法として、識別時の発火率をレベル分類に応用できないか、という案が提案されている。これは、画像識別時に、目的以外のパターンにも若干の発火率が発生することを利用し、100%画像に対する発火の割合を数式化してレベル分類を行おうという試みである。そのためには、学習画像の枚数を100枚程度に増やし、100%時に対する発火率をより明確に示しさなければならない。学習画像作成にあたっては、1枚の学習用画像に正規乱数で作成した100種類のマスクをかけて100パターンの学習画像を用意する案が有力である。

また、画像単位で減量が判別困難な液体の食物の判定には、食物の重量も判別用のデータとして利用する手法や、食器を逆円錐状のものに限定して、その減量を画像単位で明確にするなどの案が提案されている。

今後、更なる性能の向上と問題解決の新しい手法の考案が待たれる。完成した食物摂取量計測システムが病院内患者の食事・栄養管理におおいに貢献することが期待できる。

7 章　謝辞

日々熱心にご指導していただいた高知工科大学竹田史章教授に感謝いたいします。また、量計測システムの研究開発に協力していただいたグローリー工業（株）の吉田様、長田様、およびヴィスト（株）の小林茂樹代表取締役、形部麗子取締役、エヌビーサイト（株）の尾江俊昭代表取締役、高知学園短期大学の安房田先生、高知潮江高橋病院の山本様、以下ご協力していただいた皆様に深く感謝いたします。

8 章　文献

(1) 竹田史章・大松繁、「ニューロ紙幣識別ボードの開発」、電気学会論文誌 C、pp1～6、(平成 8 年 3 月号抜刷)
(2) 竹田史章・西蔭紀洋、内田久也、中原昌樹、「紙幣用ニューロテンプレートマッチング識別手法の開発」、電気学会論文誌 C、pp1～7、（2001）
(3) 竹田史章、大松繁、「マスク方式によるニューロ紙幣識別機の開発」、システム制御情報学会、Vol.6、pp31～36、(1993)
(4) 麻生英樹、「ニューラルネットワーク情報処理」、産業図書、pp4～14、33～37、(1988)
(5) 竹田史章、西蔭紀洋、藤田靖、「自己学習型ニューロ紙幣識別ボードの開発とその汎用展開」、電気学会論文誌 C、pp1～8、(平成 13 年 1 月号掲載予定)
(6) 船久保登、「パターン認識」、共立出版社、pp114～120、(1991)
(7) 岡崎彰夫、「はじめての画像処理」、工業調査会、pp39～41、（2000）

≪付録≫

切出しプログラム（CUTOUT）

```
/*********************************************************
******
***　ファイル名　　：　MKSLAB.c
***
***　概要　　：　画像から対象を切出す。
*********************************************************
******/
#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include    <graph.h>
#include    <conio.h>
#include    <time.h>
#include    <windows.h>

#define             IMG_STY             5
#define             IMG_STX             5

#define             IMG_STY_RAW
        35          // 生画像表示左上端Y座標表示位置
#define             IMG_STX_RAW         280         //
生画像表示左上端X座標表示位置
#define             IMG_STY_LABEL       260         //
ラベル画像表示左上端Y座標表示位置
#define             IMG_STX_LABEL       280         //
ラベル画像表示左上端X座標表示位置

#define             OK                  0           //
正常終了
#define             NG                  1           //
異常終了
#define             BMP256              256         //
256 色 BMP
#define             BIN_TH    130       // ２値化のし
きい値

#define             LABEL_START_NO              2
                    /* ラベル開始番号 */
#define             BIN_1
        255                 /* ２値化後の'1'画素 */
#define             BIN_0
        0                   /* ２値化後の'0'画素 */
#define             IMAGE_HEIGHT
        168                 /* 入力画像の高さ　単位
画素 */
#define             IMAGE_WIDTH
        260                 /* 入力画像の幅　単位
画素 */
#define             LABEL_MAX_NO
        230                 /* ラベルの最大番号
                    */
#define             LABEL_MAX_NUM
        250                 /* ラベルの最大個数
                    */
#define             LABEL_IMAGE_COLOR
        0x00        /* ラベル画像の色
                    */
#define             LABEL_RECT_COLOR
        0x0a        /* ラベル画像の外接長方形の色 */
#define             LABEL_RECT_CENT_COLOR 0x0f
        /* ラベル画像の外接長方形の中心の色 */

#define             BMP_SIZE_MAX_X
        320                 /* BMP ファイルの幅の最
大値 */
#define             BMP_SIZE_MAX_Y
        200                 /* BMP ファイルの高さの
最大値 */

void        disp_information(void);             /* 画像情報
表示 */
void        disp_label_img( void );
void        disp_raw_img( void );
void        disp_rect( int );                   /* 外接長方
形の表示 */
void wait( double ) ;
void set_palette(       unsigned char ,         unsigned
char ,      unsigned char ,

            unsigned char );
void        dsp_menu(int , int ) ;
void        dsp_end() ;
int         bmp_read( char * );
int         labeling( int               * );
void exe_label( int       , int         , int );
void        mks_prm( void );
void error_end( char * );               /* エラー表示 */
void image_contraction();               /* 収縮処理 */
void image_bin2();                      /* 入力画像の２値化 */

int                 xsize ;
                    /*　入力画像の幅 */
int                 ysize ;
                    /*　入力画像の高さ */

unsigned char       *image ;            /* 入力画像
領域(２値化後) */
unsigned char       *image_io ;         /* 入力画像
領域 */
unsigned char       *image_label ;      /* ラベル画
像領域 */
int         px=1 , py=1 ;
            /* 画素ごとの表示幅と高さ　*/
int         label_xmin[LABEL_MAX_NUM] ;/* ラベル領
域の左上端X座標 */
int         label_xmax[LABEL_MAX_NUM] ;/* ラベル領
域の右下端X座標 */
int         label_ymin[LABEL_MAX_NUM] ;/* ラベル領
域の左上端Y座標 */
int         label_ymax[LABEL_MAX_NUM] ;/* ラベル領
域の右下端Y座標 */
int         label_center_y[LABEL_MAX_NUM] ;     /*
ラベル領域の中心Y座標 */
int         label_center_x[LABEL_MAX_NUM] ;     /*
ラベル領域の中心X座標 */
int         label_num ;
                        /* ラベル数 */

char        fname_i[100] ;                      /*
入力画像ファイル名 */
// 2000.11.22 収縮回数の指定の追加
int         contraction_num ;
            /* 収縮回数 */
#define     TRUE    1
                    /* 真 */
#define FALSE       0
                    /* 偽 */

// 2000.11.27 切出しエリア制約条件の入力
int                 cutflag ;     // 1:切出しエリア制約条
件の使用　0:未使用
int                 cut_x_max ;  // X 方向上限
int                 cut_x_min ;  // X 方向下限
int                 cut_y_max ;  // Y 方向上限
int                 cut_y_min ;  // Y 方向下限
#define     LABEL_INF_DISP  0           // 1:ラベル情
報表示　0:ラベル情報非表示
int         label_num_new ;                     //
切出し制約時のラベル数 */

/*********************************************************/
/*

                        */
/*          image_bin2

                        */
/*
                        */
/*                      入力画像の２値化処理
                                    */
/*

                        */
/*********************************************************/
void image_bin2()
{
unsigned char *imagew ;
long                                    i , j ;

        imagew = image_io ;
        for ( i=0 ; i<ysize ; i++ )
        {
                for ( j=0 ; j<xsize ; j++ , imagew++ )
                {
                        if    (    *imagew    >=
BIN_TH )
                        {
                        /* しきい値より大きいと
き */
                                *imagew    =
BIN_1 ;
                        }
                        else
                        {
                                *imagew    =
BIN_0 ;
                        }
                }
```

```
        }
}

/*******************************************************/
/*                                                     */
/*          image_contraction                          */
/*                                                     */
/*          入力画像の収縮処理                         */
/*                                                     */
/* 2000.11.22 改造                                     */
/*******************************************************/
void image_contraction()
{
long        i , j ;
unsigned char *pw1 , *pw2 ;

        pw1 = image ;
        pw2 = image_io ;
        for ( i=0 ; i<ysize ; i++ )
        {
                for ( j=0 ; j<xsize ; j++ )
                {
                        *pw1++ = *pw2++ ;
                }
        }

        for ( i=1 ; i<ysize-1 ; i++ )
        {
                for ( j=1 ; j<xsize-1 ; j++ )
                {
                        pw1 = image + i*xsize + j ;
                        pw2 = image_io + i*xsize + j ;
                        if ( *pw2 == BIN_0 )
                        {
                                *(pw1 - (long)xsize - 1L) = BIN_0 ;
                                *(pw1 - (long)xsize     ) = BIN_0 ;
                                *(pw1 - (long)xsize + 1L) = BIN_0 ;
                                *(pw1 - 1L) = BIN_0 ;
                                *pw1 = BIN_0 ;
                                *(pw1 + 1L) = BIN_0 ;
                                *(pw1 + (long)xsize - 1L) = BIN_0 ;
                                *(pw1 + (long)xsize     ) = BIN_0 ;
                                *(pw1 + (long)xsize + 1L) = BIN_0 ;
                        }
                }
        }
}

/*******************************************************/
/*                                                     */
/*          labeling                                   */
/*                                                     */
/*          ラベル処理                                 */
/*                                                     */
/*                                                     */
/*  label_count                    ラベル数            */
/*          戻り値      0:正常終了 -1:ラベルの個数が多すぎた */
/*                                                     */
/*******************************************************/
int         labeling( int           *label_count )
{
long                    i , j ;
unsigned char           labelno ;
unsigned char *imagew , *image_labelw ;
long                    lw ;
                                                        /*
ラベル領域に初期値となる入力画像をコピー */
        imagew = image ;
        image_labelw = image_label ;
        for ( i=0 ; i<ysize ; i++ )
        {
                for ( j=0 ; j<xsize ; j++ , imagew++ , image_labelw++ )
                {
                        *image_labelw           = *imagew ;
                }
        }
        labelno = LABEL_START_NO ;      /* 開始ラベル番号の設定 */
        for ( i=0 ; i<ysize ; i++ )
        {
                for ( j=0 ; j<xsize ; j++ )
                {
                        lw = (long)i*xsize + j ;
                        if ( *(image_label+lw) == BIN_1 )
                        {
                        /*   "1"  画素のあるところ */
                                if  (   labelno >= LABEL_MAX_NO )
                                {               /* ラベル数が多すぎるとき */
                        *label_count = labelno - LABEL_START_NO ;
                        return(-1) ;
                                }
                        /*   ラベリングの実行 */
                        exe_label( (int)j , (int)i , labelno );
                                labelno++ ;
                        }
                }
                                                /*   ラベル領域の個数 */
        *label_count = labelno - LABEL_START_NO ;
        return(0);
}

/*******************************************************/
/*                                                     */
/*          exe_label                                  */
/*                                                     */
/*          ラベル処理                                 */
/*                                                     */
/*                                                     */
/*  xpos    : ラベル処理を行うX座標                    */
/*  ypos    : ラベル処理を行うY座標                    */
/*  labelno: 処理中のラベル番号                        */
/*                                                     */
/*******************************************************/
void exe_label( int     xpos , int   ypos , int   labelno )
{
long                    i ;
long                    j ;
int                     num ;
unsigned char           *pw , *pww ;

        *(image_label+(long)ypos*xsize+xpos)            = labelno ;
        for ( ;; )
        {
                num = 0 ;
                for ( i = 0 ; i < ysize ; i++ )
                {
                        for ( j = 0 ; j < xsize ; j++ )
                        {
                                pw              = image_label + i*xsize + j ;
                                if  ( *pw  ==
```

```
labelno )
                                        {
        /* 指定したラベル番号あり */
        /* ８近傍に"1"画素があるならば､ラベル番号をセット */
        pww = pw - xsize - 1 ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw - xsize ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw -xsize + 1 ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw - 1 ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw + 1 ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw + xsize - 1 ;
                                        if
( *pww == BIN_1 )
                                        {
        *pww = labelno ;
        num++ ;
                                        }
        pww = pw + xsize ;
                                        if
( *pww == BIN_1)
                                        {
        *pww  = labelno ;
        num++ ;
                                        }
        pww = pw + xsize + 1 ;
                                        if
( *pww == BIN_1)
                                        {
        *pww  = labelno ;
        num++ ;
                                        }
                                }
                        }
                if( num == 0 )
                        return ;
        }
}
/*****************************************************/
/*
                                */
/*      calc_rect
                                */
/*              外接長方形の計算
                                */
/*
                                */
/*
                                */
/*****************************************************/
void calc_rect()
{
int     i , j , iw ;
unsigned char *image_labelw ;
                                /*  最小値、最大値の初期値の設定 */
        for ( i=0 ; i<label_num ; i++ )
        {
                label_xmin[i] = xsize ;
                label_xmax[i] = 0 ;
                label_ymin[i] = ysize ;
                label_ymax[i] = 0 ;
        }
                                /* ラベルごとの領域位置の最大値と最小値を求める */
        image_labelw = image_label ;
        for ( i=0 ; i<ysize ; i++ )
        {
                for ( j=0 ; j<xsize ; j++ , image_labelw++)
                {
                        if ( *image_labelw >= LABEL_START_NO )
                        {
                                iw = *image_labelw - LABEL_START_NO ;
                                if ( i < label_ymin[iw] )
        label_ymin[iw] = (int)i ;
                                if ( i > label_ymax[iw] )
        label_ymax[iw] = (int)i ;
                                if ( j < label_xmin[iw] )
        label_xmin[iw] = (int)j ;
                                if ( j > label_xmax[iw] )
        label_xmax[iw] = (int)j ;
                        }
                }
        }
                                /* ラベル領域の中心位置を求める */
        for ( i=0 ; i<label_num ; i++ )
        {
                label_center_x[i] = ( label_xmax[i] + label_xmin[i] ) / 2 ;
                label_center_y[i] = ( label_ymax[i] + label_ymin[i] ) / 2 ;
        }
}
/*****************************************************/
/*
                                */
/*      error_end
                                */
/*              エラー表示
                                */
/*
                                */
/*
                                */
/*****************************************************/
void error_end( char *charbuf )
{
```

```
                        _setvideomode( _DEFAULTMODE );
/*  画面モード＝ "03H"  */
                        printf("¥x1B[30;19H ERROR %s !!!
¥n",charbuf );
                        getch();
}

/*******************************************************/
/*
                                                */
/*          main
                                                */
/*                      メイン処理
          */
/*
                                                */
/*          引数： argc             - パラメータ数
                                                        */
/*                      *argv[] - パラメータ文字列
                                                */
/*
                                                */
/*******************************************************/
void main( int argc, char *argv[] )
{
int                     errw ;
// 2000.11.22
unsigned char           *pw1 , *pw2 ;
int                     i , j , k ;
char        buf[10] ;

          if( argc == 1 )           /* コマンド引き数のチェック
          */
                        mks_prm();              /* コマンド
引き数入力 */
// 2000.11.22 収縮回数の指定の追加
//        else if( argc != 2 )
//        {
//                      printf("¥n Usage: CUTOUT [in_file]
");
//                      printf("¥n        ( in_file    : 入力フ
ァイル                ) ");
//                      printf("¥n¥n    Push    any    key!");
getch(); exit(1);
//        }
// 2000.11.27 切出しエリア制約条件の入力
//        else if( argc != 3 )
          else if( argc != 8 )
          {
                        printf("¥n Usage: CUTOUT [in_file]
[num] [cutflag] [xmax] [xmin] [ymax] [ymin]");
                        printf("¥n        ( in_file    : 入力フ
ァイル                ) ");
                        printf("¥n        ( num        : 収縮
回数     max 100      ) ");
                        printf("¥n        ( cutflag    : 切出し
制約条件   Y/N        ) ");
                        printf("¥n        ( xmax       : ラベ
ル   X 方向最大値     ) ");
                        printf("¥n        ( xmin       : ラベ
ル   X 方向最小値     ) ");
                        printf("¥n        ( ymax       : ラベ
ル   Y 方向最大値     ) ");
                        printf("¥n        ( ymin       : ラベ
ル   Y 方向最小値     ) ");
                        printf("¥n¥n    Push    any    key!");
getch(); exit(1);
          }
          else
          {


                        strcpy( fname_i , argv[1] );
                        contraction_num = atoi(argv[2]);
                        strcpy( buf , argv[3] );
                        if ( ( buf[0] == 'Y' ) || ( buf[0] ==
'y' ) )
                        {
                                cutflag = TRUE ;
                        }
                        else
                        {
                                cutflag = FALSE ;
                        }
                        cut_x_max = atoi(argv[4]);
                        cut_x_min = atoi(argv[5]);
                        cut_y_max = atoi(argv[6]);
                        cut_y_min = atoi(argv[7]);
          }
// 2000.11.27 引数チェック
          if    (    (    contraction_num    <    1    )    ||
( contraction_num > 100 ) )
          {
                        printf("  収縮回数エラー
1<=max<=100 ¥n");
                        printf("¥n¥n    Push    any    key!");
getch(); exit(1);
          }
          if ( cutflag )
          {
                        if ( cut_x_min < 0 )
                        {
                                printf(" X方向最小値エラ
ー  ¥n");
                                printf("¥n¥n  Push  any
key!"); getch(); exit(1);
                        }
                        if ( cut_y_min < 0 )
                        {
                                printf(" Y 方向最小値エラ
ー  ¥n");
                                printf("¥n¥n  Push  any
key!"); getch(); exit(1);
                        }
                        if ( cut_x_min >= cut_x_max )
                        {
                                printf(" X方向最大値・最
小値エラー  ¥n");
                                printf("¥n¥n  Push  any
key!"); getch(); exit(1);
                        }
                        if ( cut_y_min >= cut_y_max )
                        {
                                printf(" Y 方向最大値・最
小値エラー  ¥n");
                                printf("¥n¥n  Push  any
key!"); getch(); exit(1);
                        }
          }

          dsp_menu(xsize , ysize);
                                                        /*
BMP ファイルのリード */
          if ( ( errw = bmp_read(fname_i )) != OK )
          {
                        if ( errw == 2 )
                                error_end("BMP    FILE
SIZE OVER ");
                        else if ( errw == 3 )
                                error_end("MEMORY
ALLOC ");
                        else
                                error_end("BMP    FILE
READ ");
                        dsp_end();
                        exit(1);
          }

          disp_raw_img();                   /* 入力画像
の表示 */

          image_bin2();                     /* 入力画像
の２値化 */
                        /* ノイズ除去 */
// 2000.11.22 収縮回数の指定の追加
//        image_contraction();    /* 収縮処理 */
          for ( i=0 ; i<contraction_num ; i++ )
          {
                        image_contraction();    /* 収縮処理
*/

                                /* データのコピー */
                        pw1 = image ;
                        pw2 = image_io ;
                        for ( j=0 ; j<ysize ; j++ )
                        {
                                for ( k=0 ; k<xsize ; k++ )
                                {
                                        *pw2++      =
*pw1++ ;
                                }
                        }
          }


          /* ラベル処理の実行 */
          if ( labeling( &label_num ) != 0 )
          {
```

```
		/* ラベル数が多すぎるという表示をする */
			error_end("LABEL OVER");
			dsp_end();
			exit(1);
		}

		calc_rect();			/* 外接長方形の計算 */

// 2000.11.27 切出しエリア制約条件の入力
#if LABEL_INF_DISP
		disp_information() ;	/* 画像情報表示 */
#else
						/* ラベル数、ファイル名の表示 */
//		printf("¥x1B[5;18H%3d", label_num ) ;
		printf("¥x1B[6;18H%s", fname_i ) ;
#endif
		disp_label_img();		/* ラベル画像の表示 */

		disp_rect( label_num ) ;	/* 外接長方形の表示 */

		dsp_end();
}

/*******************************************************/
/*
				*/
/*		mks_prm
			*/
/*			コマンドパラメータ入力
								*/
/*
				*/
/*
				*/
/*******************************************************/
void	mks_prm( void )
{
		int			yesno;
// 2000.11.22
		int			iflag ;

		printf("¥x1B[2J");
		/* クリアスクリーン		*/
		printf("¥x1B[6;10H < 米粒画像ファイルから米粒を切出す >");

// 2000.11.27 切出しエリア制約条件の入力
		printf("¥x1B[16;10H 画像ファイル名を指定して下さい : ");
		scanf("%s", fname_i);

// 2000.11.22 収縮回数の指定の追加
//		printf("¥x1B[22;10H		OK? (Y/N)  ");
		iflag = TRUE ;
		while(iflag)
		{
// 2000.11.27 切出しエリア制約条件の入力
			printf("¥x1B[17;30H                    ");
			printf("¥x1B[17;10H 収縮回数を指定して下さい 1<=max<=100 : ");
			scanf("%d", &contraction_num);
			if ( ( contraction_num >= 1 ) && ( contraction_num <= 100 ) )
			{
				iflag = FALSE ;
			}
		}
// 2000.11.27 切出しエリア制約条件の入力
		printf("¥x1B[18;10H 切出し制約条件の入力? (Y/N)  ");
		yesno = getch();

		if( yesno == 'Y' || yesno == 'y' )
		{
			cutflag = TRUE ;
		}
		else
		{
			cutflag = FALSE;
		}
		if ( cutflag )
		{
			printf("¥x1B[19;10H X方向上限 : ");
			scanf("%d", &cut_x_max);
			printf("¥x1B[20;10H X方向下限 : ");
			scanf("%d", &cut_x_min);
			printf("¥x1B[21;10H Y方向上限 : ");
			scanf("%d", &cut_y_max);
			printf("¥x1B[22;10H Y方向下限 : ");
			scanf("%d", &cut_y_min);
		}

		printf("¥x1B[23;10H		OK? (Y/N)  ");

		while(1){
			yesno = getch();
			if( yesno == 'Y' || yesno == 'y' )
				break;
			else if( yesno == 'N' || yesno == 'n' )
				exit(1);
			else
	putchar(0x07);
		}
}

/*******************************************************/
/*
				*/
/*		disp_information
					*/
/*			画像情報表示処理
					*/
/*
				*/
/*
				*/
/*******************************************************/
void disp_information( )
{
int		i ;
char	buf[50] ;
						/* ラベル数、ファイル名の表示 */
		printf("¥x1B[5;18H%3d", label_num ) ;
		printf("¥x1B[6;18H%s", fname_i ) ;
						/* ラベル領域の中心位置を表示 */
		if ( label_num <= LABEL_MAX_NUM/2 )
		{
			for ( i=0 ; i<label_num ; i++ )
			{
	_settextposition( 12+i,4 );
	sprintf(buf,"%4d %4d %4d ", i+1 , label_center_x[i] , label_center_y[i]);
				_outtext(buf);
			}
		}
		else
		{
			for ( i=0 ; i<LABEL_MAX_NUM/2 ; i++ )
			{
	_settextposition( 12+i,4 );
	sprintf(buf,"%4d %4d %4d ", i+1 , label_center_x[i] , label_center_y[i]);
				_outtext(buf);
			}
			for ( i=LABEL_MAX_NUM/2 ; i<label_num ; i++ )
			{
				_settextposition( 12+i-
```

```
LABEL_MAX_NUM/2,19 );
                sprintf(buf,"%4d %4d %4d ", i+1 , label_center_x[i] , label_center_y[i]);
                                        _outtext(buf);
                        }
                }
}
/*******************************************************/
/*
                                        */
/*              disp_raw_img
                */
/*                      生画像を表示
                */
/*
                                        */
/*
                                        */
/*******************************************************/
void            disp_raw_img( void )
{
                long                    i, j, k, t;
                int                     xs, ys;
                unsigned char           *datap;

                xs = IMG_STX_RAW;
                ys = IMG_STY_RAW;

                datap = image_io;

/*              _setcolor( 4 );*/               /* 画像表示のバック */
/*              _rectangle(                     _GFILLINTERIOR, xs,ys,xs+xsize*px,ys+ysize*py );*/

                for( i=0; i<ysize ; i++ )
                {
                        for( j=0; j<xsize; j++, datap++ )
                        {
                                if( *datap > 256*8/10 )
                _setcolor( (unsigned char)(0x0F));
                                else if( *datap > 256*7/10 )
                _setcolor( (unsigned char)(0x0D));
                                else if( *datap > 256*6.5/10 )
                _setcolor( (unsigned char)(0x0B));
                                else if( *datap > 256*6/10 )
                _setcolor( (unsigned char)(0x09));
                                else if( *datap > 256*5.5/10 )
                _setcolor( (unsigned char)(0x06));
                                else if( *datap > 256*5/10 )
                _setcolor( (unsigned char)(0x07));
                                else if( *datap > 256*4.5/10 )
                _setcolor( (unsigned char)(0x05));
                                else if( *datap > 256*4/10 )
                _setcolor( (unsigned char)(0x04));
                                else if( *datap > 256*3.5/10 )
                _setcolor( (unsigned char)(0x08));
                                else if( *datap > 256*3/10 )
                _setcolor( (unsigned char)(0x02));
                                else if( *datap > 256*2.5/10 )
                _setcolor( (unsigned char)(0x01));
                                else
                _setcolor( (unsigned char)(0x03));
                                for ( k=0 ; k<py ; k++ ){
                                        for( t=0 ; t<px ; t++ ){
                _moveto( (int)(xs+j*px+t), (int)(ys+i*py+k) );
                _lineto( (int)(xs+j*px+t), (int)(ys+i*py+k) );
                                        }
                                }
                        }
                }
}
/*******************************************************/
/*
                                        */
/*              disp_label_img
                */
/*
                                        */
/*                      ラベル画像を表示                                */
/*
                                        */
/*******************************************************/
void            disp_label_img( void )
{
                long                    i, j, k, t;
                int                     xs, ys;
                unsigned char           *datap;

                xs = IMG_STX_LABEL ;
                ys = IMG_STY_LABEL ;

                datap = image_label ;

                for( i=0; i<ysize ; i++ )
                {
                        for( j=0; j<xsize; j++, datap++ )
                        {
                                if( ( *datap >= LABEL_START_NO ) && ( *datap <= LABEL_MAX_NO ) )
                _setcolor( (unsigned char)(LABEL_IMAGE_COLOR));
                                else if( *datap == 255 )
                _setcolor( (unsigned char)(0x0A));
                                else if( *datap > 256*8/10 )
                _setcolor( (unsigned char)(0x0F));
                                else if( *datap > 256*7/10 )
                _setcolor( (unsigned char)(0x0D));
                                else if( *datap > 256*6.5/10 )
                _setcolor( (unsigned char)(0x0B));
                                else if( *datap > 256*6/10 )
                _setcolor( (unsigned char)(0x09));
                                else if( *datap > 256*5.5/10 )
                _setcolor( (unsigned char)(0x06));
                                else if( *datap > 256*5/10 )
                _setcolor( (unsigned char)(0x07));
                                else if( *datap > 256*4.5/10 )
                _setcolor( (unsigned char)(0x05));
                                else if( *datap > 256*4/10 )
                _setcolor( (unsigned char)(0x04));
                                else if( *datap > 256*3.5/10 )
                _setcolor( (unsigned char)(0x08));
                                else if( *datap > 256*3/10 )
```

```
                _setcolor( (unsigned char)(0x02));
                                else    if(     *datap     >
256*2.5/10 )
                _setcolor( (unsigned char)(0x01));
                                else
                _setcolor( (unsigned char)(0x03));
                                for ( k=0 ; k<py ; k++ )
                                {
                                        for(    t=0    ;
t<px ; t++ )
                                        {
                _moveto( (int)(xs+j*px+t), (int)(ys+i*py+k) );
                _lineto( (int)(xs+j*px+t), (int)(ys+i*py+k) );
                                        }
                                }
                        }
                }
}

/*****************************************************
***
***   名称：          disp_rect
***
***   機能：          外接長方形の表示
***
***  int labelnum :  ラベルの個数
*****************************************************/

                void    disp_rect( int labelnum )
{
                int             i ;
// 2000.11.27 切出しエリア制約条件の入力
                int             xw , yw ;
                label_num_new = 0 ;

                _setcolor(0x0a);
                for ( i=0 ; i<labelnum ; i++ )
                {
                        if ( cutflag )
                        {
// 2000.11.27 切出しエリア制約条件の入力
                                xw = label_xmax[i] -
label_xmin[i] ;
                                yw = label_ymax[i] -
label_ymin[i] ;
                                if (        ( xw <=
cut_x_max ) &&
                ( xw >= cut_x_min ) &&
                ( yw <= cut_y_max ) &&
                ( yw >= cut_y_min )     )
                                {
                label_num_new++ ;
                                                /* 外接長方
形の表示 */
                _setcolor(LABEL_RECT_COLOR);
                _rectangle(             _GBORDER        ,
IMG_STX_LABEL+label_xmin[i] ,

                IMG_STY_LABEL+label_ymin[i] ,

                IMG_STX_LABEL+label_xmax[i] ,

                IMG_STY_LABEL+label_ymax[i] );
                                                /* 外接長方
形の中心の表示 */
                _setcolor(LABEL_RECT_CENT_COLOR);
                _setpixel( IMG_STX_LABEL+label_center_x[i] ,

                IMG_STY_LABEL+label_center_y[i] ) ;
                                }
                                printf("¥x1B[5;18H%3d",
label_num_new ) ;
                        }
                        else
                        {
#if LABEL_INF_DISP
#else
                                        printf("¥x1B[5;18H%3d",
label_num ) ;
#endif
                                        /* 外接長方形の表示 */
                _setcolor(LABEL_RECT_COLOR);
                                        _rectangle( _GBORDER ,
IMG_STX_LABEL+label_xmin[i] ,
                        IMG_STY_LABEL+label_ymin[i] ,
                        IMG_STX_LABEL+label_xmax[i] ,
                        IMG_STY_LABEL+label_ymax[i] );
                                        /* 外接長方形の中心の表
示 */
                _setcolor(LABEL_RECT_CENT_COLOR);
                _setpixel( IMG_STX_LABEL+label_center_x[i] ,
                IMG_STY_LABEL+label_center_y[i] ) ;
                        }
                }
}

/*****************************************************
***
***   名称：          dsp_menu
***
***   機能：          グラフィック画面の初期表示
***
***   引数：          IN:     int xsize BMP 画像の幅
***                           int ysize BMP 画像の高
さ
***
***                           OUT:
***
***   戻値：
***
***   備考：
***
*****************************************************/
void    dsp_menu(int xsize , int ysize )
{

  int xs;

                /* ビデオモードセット */
                if( ! _setvideomode( _VRES16COLOR ) )
                        exit(1);

                /* バックグラウンド（青） */
                set_palette( 0x00, 0x00,0x00,0x2a );
                _clearscreen( _GCLEARSCREEN );

                /** カラーパレット設定 **/
                /*              0x0A */
                set_palette( 0x3A, 0x00,0x2a,0x00 );

                /* 紙幣画像表示のために１２カラー使用する */
                set_palette(  0x03,   0,   0,   0 ) ;
                set_palette( 0x01, 10, 10, 10 ) ;
                set_palette( 0x02, 16, 16, 16 ) ;
                /*                                      0x08 */
                set_palette( 0x04, 28, 28, 28 ) ;
                set_palette( 0x05, 34, 34, 34 ) ;
                /*                                      0x07 */
                set_palette( 0x14, 46, 46, 46 ) ;       /* 0x06 */
                set_palette( 0x39, 50, 50, 50 ) ;       /* 0x09 */
                set_palette( 0x3B, 53, 55, 55 ) ;       /* 0x0B */
                set_palette( 0x3D, 60, 60, 60 ) ;       /* 0x0D */
                /*                                      0x0F */

                /* 画面全体塗り潰し（白） */
                _setcolor( 7 );
                _rectangle( _GFILLINTERIOR, 0,0,640,480 );

                /* 全体情報パネル表示 */
                _setcolor( 3 );
                _rectangle( _GFILLINTERIOR, 17,14,272,130 );
                _setcolor( 0 );
                _rectangle( _GFILLINTERIOR, 15,12,270,128 );

                /* 全体情報パネル内容表示 */
                _settextcolor( 9 );
                _settextposition( 3,6 );
                _outtext("X SIZE    :" );
                _settextposition( 4,6 );
```

```
            _outtext("Y SIZE     :" );
            _settextposition( 5,6 );
            _outtext("LABEL NUM :" );
            _settextposition( 6,6 );
            _outtext("FILE NAME :" );
            /* ラベル情報パネル表示 */
            _setcolor( 3 );
            _rectangle(                 _GFILLINTERIOR,
17,142,272,466 );
            _setcolor( 0 );
            _rectangle(                 _GFILLINTERIOR,
15,140,270,464 );

            _setcolor( 3 );                 /* バック   */
            _rectangle(                 _GFILLINTERIOR,
xs,IMG_STY,xs+224,IMG_STY+32*4 );
            _rectangle(                 _GFILLINTERIOR,
xs,IMG_STY+32*4+10,xs+224,IMG_STY+32*4*2+10 );
            _setcolor( 9 );                 /* 枠
                                            */
            _rectangle(     _GBORDER,   xs-1,IMG_STY-
1,xs+224+1,IMG_STY+32*4+1 );
            _rectangle(         _GBORDER,       xs-
1,IMG_STY+32*4+10-
1,xs+224+1,IMG_STY+32*4*2+10+1 );
            /* 階調バー表示                         */
            _setcolor( 3 );
            _rectangle( _GFILLINTERIOR, 36,435,51,445 );
            _setcolor( 1 );
            _rectangle( _GFILLINTERIOR, 52,435,67,445 );
            _setcolor( 2 );
            _rectangle( _GFILLINTERIOR, 68,435,83,445 );
            _setcolor( 8 );
            _rectangle( _GFILLINTERIOR, 84,435,99,445 );
            _setcolor( 4 );
            _rectangle(                 _GFILLINTERIOR,
100,435,115,445 );
            _setcolor( 5 );
            _rectangle(                 _GFILLINTERIOR,
116,435,131,445 );
            _setcolor( 7 );
            _rectangle(                 _GFILLINTERIOR,
132,435,147,445 );
            _setcolor( 6 );
            _rectangle(                 _GFILLINTERIOR,
148,435,163,445 );
            _setcolor( 9 );
            _rectangle(                 _GFILLINTERIOR,
164,435,179,445 );
            _setcolor( 11 );
            _rectangle(                 _GFILLINTERIOR,
180,435,195,445 );
            _setcolor( 13 );
            _rectangle(                 _GFILLINTERIOR,
196,435,211,445 );
            _setcolor( 15 );
            _rectangle(                 _GFILLINTERIOR,
212,435,227,445 );
            _setcolor( 9 );                 /* 枠
                                            */
            _rectangle( _GBORDER, 35,434,228,446 );

            /* 画像パネル表示 */
    _setcolor( 3 );
    _rectangle( _GFILLINTERIOR, 278,14,624,466 );
    _setcolor( 0 );
    _rectangle( _GFILLINTERIOR, 276,12,622,464 );

                        /* プログラムタイトル */
            _settextcolor( 0x0a );
            _settextposition( 2,6 );
            _outtext("***   RICE   CUTOUT   PROGRAM
1.02" );

                        /* 画像タイトル出力 */
            _settextcolor( 9 );
            _settextposition( 2,55 );
            _outtext("RAW IMAGE" );
            _settextposition( 16,55 );
            _outtext("LABEL IMAGE" );

// 2000.11.27 切出しエリア制約条件の入力
#if LABEL_INF_DISP
                        /* ラベル情報 */
            _settextcolor( 0x0a );
            _settextposition(10,6 );
            _outtext("LABEL AREA CENTER" );
            _settextposition(11,6 );
            _outtext("NO     C_X     C_Y       NO     C_X
C_Y" );
#endif
            _settextcolor( 9 );
}

/*****************************************************
***
***   名称：          dsp_end
***
***   機能：          終了時の表示及び処理
***
***   引数：          IN:
***                           OUT:
***
***   戻値：
***
***   備考：
***
*****************************************************/


void        dsp_end()
{
                                /* メモリの解放 */
            if ( image != NULL )
                        free(image);
            if ( image_label != NULL )
                        free(image_label);
            if ( image_io != NULL )
                        free(image_io);

            printf("¥x1B[30;19H Push any key ! " );
            getch();

/*          printf("¥x1B[30;2H Complete ! " );*/
/*          wait(1); */
            _setvideomode( _DEFAULTMODE );  /*  画面
モード＝ "03H"   */
            exit(0);
}

/*****************************************************
**

   名称：set_palette

   機能：パレットのＲＧＢ情報（濃淡）を設定する.

   引数：unsigned char Color_No  パレット番号
         unsigned char red       パレットのＲＧＢ情報
         unsigned char green      パレットのＲＧＢ情報
         unsigned char blue      パレットのＲＧＢ情報

*****************************************************
**/
void set_palette(       unsigned char Color_No
                                                ,
            unsigned char redi
                                                ,
            unsigned char greeni
                                                ,
            unsigned char bluei )
{
  _asm
  {
        cli
        mov dx,03c8h
        mov al,Color_No
        out dx,al
        mov dx,03c9h
        mov al,redi
        out dx,al
        mov al,greeni
        out dx,al
        mov al,bluei
        out dx,al
        sti


  }
}

/*****************************************************
**

   名称：wait

   機能：時間稼ぎ用

   引数：秒
```

```
**********************************************************
**/
void wait( double ss1 )
{
        time_t      start, finish;
        double      ss2;

        time( &start );

        while(1){
                time( &finish );
                ss2 = difftime( finish, start );
                if( ss2 > ss1 )             break;
        }
}

/**********************************************************
**

   名称      bmp_read

   機能：BMP ファイルを読む。

   引数：char *fname    BMP ファイル名
   戻値:OK：正常終了   1：読込みエラー 2:サイズオーバー 3:
メモリ獲得エラー
**********************************************************
**/
int bmp_read( char *fname  )
{

        BITMAPFILEHEADER i_bmFH  ; /* ファイル
ヘッダ */
        BITMAPINFOHEADER i_bmIH  ; /* インフォ
ヘッダ */
        FILE        *fp ;
        int         iWidth , iHeight , iWidtho ;
        unsigned char *imagei , *imageiw ;
        RGBQUAD     *rgbtriplei , *pw ;
        size_t filesize;
        long        li , lj ;

                                        /* BMP ファ
イルのオープン */
        if ( ( fp = fopen( fname , "rb" ) ) == NULL )
        {
                return(1) ;
        }
                                /* ファイルヘッダ */
        if      (       fread(      &i_bmFH      ,
sizeof( BITMAPFILEHEADER ) , 1 , fp ) < 1 )
        {
                return(1) ;
        }
                                /* インフォヘッダ */
        if      (       fread(      &i_bmIH      ,
sizeof( BITMAPINFOHEADER ) , 1 , fp ) < 1 )
        {
                return(1) ;
        }

        iWidth = (int)i_bmIH.biWidth  ;
        iHeight = (int)i_bmIH.biHeight;
        _settextposition( 3,18 );
        printf("%3d ",iWidth);
        _settextposition( 4,18 );
        printf("%3d ",iHeight);

        iWidtho = iWidth ;
        if ( ( iWidth - iWidth/4*4 ) != 0 )
        {
                iWidth = ( iWidth / 4 + 1 ) * 4 ;
        }
                                /* BMP ファイルが処理で
きるサイズかをチェックする */
        if ( iWidth > BMP_SIZE_MAX_X )          /*
幅 */
        {
                printf("BMP SIZE OVER X ¥n");
                fclose(fp);
                return(2);
        }
        if ( iHeight > BMP_SIZE_MAX_Y )
        /* 高さ */
        {
                printf("BMP SIZE OVER Y ¥n");
                fclose(fp);
                return(2);
        }

        filesize = iWidth * iHeight;

                                /* 色のインデックス領域
の獲得 */
        rgbtriplei = (RGBQUAD *)malloc( BMP256 *
sizeof(RGBQUAD));
        if( rgbtriplei == NULL )
        {
                printf("Cannot Malloc image ¥n");
                fclose(fp);
                return(3);
        }
                                /* 色のインデックス領域
の読込み */
        if ( fread((unsigned char *)rgbtriplei,BMP256 *
sizeof(RGBQUAD),1,fp)  < 1)
        {
                printf("fread error ¥n") ;
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                return(1);
        }
                /* 入力画像領域の獲得 */
        image = (unsigned char *)malloc( filesize );
        if( image == NULL )
        {
                printf("Cannot Malloc image ¥n");
                fclose(fp);
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                return(3);
        }
                /* 入力画像領域の獲得 */
        image_io = (unsigned char *)malloc( filesize );
        if( image_io == NULL )
        {
                printf("Cannot Malloc image ¥n");
                fclose(fp);
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                if ( image != NULL )
                        free(image);
                return(3);
        }
                                /*  画像読込み領域の獲得
*/
        imagei = (unsigned char *)malloc( filesize );
        if( imagei == NULL )
        {
                printf("Cannot Malloc image ¥n");
                fclose(fp);
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                if ( image != NULL )
                        free(image);
                if ( image_io != NULL )
                        free(image_io);
                return(3);
        }
                                /* ラベル領域の獲得 */
        image_label     =     (unsigned     char
*)malloc( filesize );
        if( image_label == NULL )
        {
                printf("Cannot  Malloc  image_label
¥n");
                fclose(fp);
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                if ( image != NULL )
                        free(image);
                if ( image_io != NULL )
                        free(image_io);
                if ( imagei != NULL )
                        free(imagei);
                return(3);
        }
                                                /*
RGB 画像データの読込み */
        if ( fread( (unsigned char *)imagei , filesize ,1,fp)
< 1 )
        {
                printf("fread error ¥n") ;
                if ( rgbtriplei != NULL )
                        free(rgbtriplei);
                if ( image != NULL )
                        free(image);
                if ( image_io != NULL )
                        free(image_io);
                if ( imagei != NULL )
                        free(imagei);
                if ( image_label != NULL )
                        free(image_label);
                return(1);
```

```
		}
		fclose(fp);
					/* 画像として取り出す
*/
		imageiw = imagei ;
		for ( li=0 ; li<iHeight ; li++ )
		{
			for ( lj=0 ; lj<iWidth ; lj++ ,
imageiw++ )
			{
				pw = rgbtriplei +
*imageiw ;
					/* Blue だけ
で '1' になっていることを判断する */
				*imageiw = (int)pw-
>rgbBlue ;
			}
		}
					/* 上下逆転
*/
		for ( li=0 ; li<iHeight ; li++ )
		{
			for ( lj=0 ; lj<iWidth ; lj++ )
			{
				*(image_io + (iHeight-li-
1)*iWidth + lj ) =
					*( imagei +
li*iWidth + lj) ;
			}
		}

		xsize = iWidth ;
		ysize = iHeight ;

					/* メモリの解放 */
		if ( rgbtriplei != NULL )
			free(rgbtriplei);
		if ( imagei != NULL )
			free(imagei);

		return(OK);
}
```