-

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米用色彩選別機開発に関する研究

Research on development of rice sorter method using neural network by the color information

1010372 内田久也

所属：高知工科大学情報システム工学科

指導教員：竹田史章教授

2001．2．5

要旨

現在、精米工場などで使用されている米選別用色彩選別機は、ある一定のしきい値を用いて米の選別を行う。この選別機は取り扱う米の量によって米の選別率が変化するといった問題点がある。こうした問題点を解決するために、本研究ではニューロテンプレートマッチング選別手法を新たに色彩選別機に搭載するアルゴリズムとして提案する。本研究では、提案する選別手法での米選別の有効性をシミュレーションで検討する。

Rice grading machine used in the rice cleaning factory grades rice by using constant threshold. This machine can not increace grade rate of rice according to increace of the amount of rice. To solve this problem, we propose to adopt the neuro-templates matting technique to a rice grading machine .In this paper, we make experiments of rice grading by the simulation and show the effectiveness of the proposed method

キーワード：ニューロテンプレート、ニューラルネットワーク、テンプレートマッチング、米選別、標準米、被害米

目次

1．はじめに
2．現状の色彩選別機
　＜2．1＞現状の色彩選別機
　　（2．1．1）性能実験 1
　　（2．1．2）性能実験 2
　＜2．2＞問題点
3．NN で構成する色彩選別機
　＜3．1＞新しい色彩選別機の基本構成
　＜3．2＞色彩選別機全体の制御フロー
　＜3．3＞各パーツ概要
　　（3．3．1）米の切り出し
　　（3．3．2）NN の構造
　　（3．3．3）ニューロテンプレートマッチング選別手法
4．シミュレーション
　＜4．1＞実験手順
　　（4．1．1）データ採取手順
　　（4．1．2）シミュレーション手順
　＜4．2＞NN1 による実験
　　（4．2．1）初期学習
　　（4．2．2）追加学習
　＜4．3＞NN2 による実験
　　（4．3．1）初期学習
　　（4．3．2）追加学習（1）
　　（4．3．3）追加学習（2）
　＜4．4＞実システムセンサ画像での実験
　　（4．4．1）実システム装置の構成
　　（4．4．2）データ採取
　　（4．4．3）シミュレーション
　＜4．5＞考察
5．まとめ
6．今後の課題
7．謝辞
8．参考文献
≪付録≫

1．はじめに

米や麦などの穀物類や野菜などを市場に出荷する前に、ほとんどの食物は選別工場や人の目を通して、その善し悪しの選別を行なう。その選別を行なう上で食物内部の色情報や形状などを使用する。その内、米の選別については、米１粒が大変小さいため形状による選別が難しいとされている。そこで、米選別では米内部の色情報を使用して選別作業を行なっている。

米内部の色情報を基に米を選別する米用色彩選別機は、ある選別感度を規定するしきい値を用いて米選別を行なう。この米用色彩選別機の仕組みは、機構内のシュータを落下してくる米をラインセンサなどの撮影装置で撮影する。実際の米選別は、米に電位を与え、米内部の白色部分と選別感度のしきい値を比較して被害粒の除去を行なう。この選別感度のしきい値を大きくすると、取り扱う米の落下流量が数百ｋ／ｈであれば、被害粒の除去率が 90%以上である。しかしながら、選別感度のしきい値を変更せず、米の落下流量を数千ｋ／ｈに増加すると被害粒の除去率は 90%を割る恐れがある。この点のみを考慮すると、常に米の落下流量を数百ｋ／ｈに設定し、選別感度のしきい値も高くしておくことも考えられる。しかしながら、実際の精米工場にて取り扱う米の量は、最大で数十万ｋに及ぶこともある。このため、取り扱う米の量を数百ｋ／ｈに設定しておくと、すべての米の監査終了に膨大な時間を費やすことになる。

つまり、高い被害粒除去率を維持するために落下流量を減らせば米の監査時間が長くなり、作業効率が悪くなる。逆に、落下流量を増やし、米の監査時間を短くすれば、被害粒の除去率が低下し良品である標準米の中に被害粒が混入する問題が生じる。

このような落下流量に関する問題点を解決するために、本研究では、この米用色彩選別機に非線形識別能力を持つニューラルネットワーク（以下 NN と略記）による選別手法を使用し、高性能な選別結果を期待する。この NN による選別手法は、紙幣識別や筆跡による個人認証など、選別分野において幅広い応用力を有する。さらに、NN による選別手法と、現状の米用色彩選別機の選別方法であるテンプレートマッチング識別手法を融合させたニューロテンプレートマッチング選別手法[(1)、(2)]を搭載することにより、米の選別を効率よく行うことを提案する。また、現状の米用色彩選別機では処理できなかった米を 1 粒ずつ切り出すアルゴリズムをラベリングを用いて行なうことを提案する。さらに、提案手法と切り出しアルゴリズムをＤＳＰを核とする自律型ニューロボードに搭載し、このニューロボードを色彩選別機に搭載することも提案する。

以上より、この選別手法の有効性を実データを用いてシミュレーションシステムを用いて定量的に示す。

具体的には、共同研究企業から提供された研究室用米撮影用筐体とインド米、日本

米の各良品粒と被害粒を使用し、米の画像データを採取する。この装置を用いることにより、撮影時に研究室内の照明の光を遮断し米に影が映らない。このデータ採取の際、各データ数は米の種類ごとに 100～200 枚採取する。また、実システムの撮影装置においても共同研究企業から提供された。この装置を使用してのデータ採取も行なう。

米のデータ採取が完了すると、つぎに選別シミュレーションシステムを起動する。このシミュレーションシステムについては他の共同研究企業から提供されたシステムを使用する。

シミュレーションシステムの流れは、まず、米の種類数や学習用サンプル画像の枚数などを設定する。つぎに、撮影装置で採取した画像データをシミュレーションシステム用データに変換する。最後に、提案手法による学習、選別シミュレーションを実施する。

この米選別用に新たに提案するニューロテンプレートマッチング選別手法は、NN 部分の出力層のユニット数などの構成の違いにより、出力ユニット数が 1 つの構成（NN1）と出力ユニット数が 2 つの（NN2）の構成がある。本研究では、NN1 と NN2 を使用した米選別のシミュレーションを実施する。以上により、NN1 あるいは NN2 のうち、どちらが米の選別用として優れた選別性能を示すか検証を実施する。

また、米を 1 粒ずつ切り出すアルゴリズムについては、8‐近傍収縮処理とラベリング処理を基にアルゴリズムの検討を実施する。さらに、このアルゴリズムを基に米を 1 粒ずつ切り出すシステムを作成する。この切り出しシステムの性能を確認する上で、機構内を落下する米の状態を擬似的に再現する。その画像に対して切り出しシステムを起動させ、米が正確に切り出せるか確認も実施する。

以上のような方法で、提案した新しい米の選別手法であるニューロテンプレートマッチング選別手法の有効性を示す。

2．現状の色彩選別機

この章では、精米工場などの市場に存在する現状の米用色彩選別機について機能や仕様などを説明し、その問題点なども説明する。

＜2．1＞現状の色彩選別機

現状の米用色彩選別機は、シュータ内を落下してくる米に対してラインセンサなどの撮影装置を用いて米を撮影する。撮影した米の画像データに電位を与え、米の白色部分と、選別感度と呼ばれるしきい値とを比較する。比較した結果、評価した米が選別感度のしきい値より低ければ、その米を被害粒とみなしその米に対し空圧を与え除去する仕組みである。

詳細に米用色彩選別機について説明すると、米はラインセンサで撮影する際と、空圧で除去する際の 2 回シュータ内を落下する。選別感度のしきい値は、20～80 までの 7 段階ある。

この米用色彩選別機の現段階の性能を確認する上で性能実験を実施する。なお、この実験は共同研究先企業（以下（株）セイレイ工業と表記）にて実施する。投入した米は、白度 31．3 度のジャポニカ種標準白米と白度 14 度の赤色着色もみの 2 種類を投入する。赤色着色もみの混入率を 3%、2%、1%、0.7%、0.4%、0.1%とし、性能を確認する。

（2．1．1）性能実験 1

性能実験 1 では、米の供給量を均等にし、選別感度ともみ混入率を変化させての除去率の変化を検証する。結果は、表 2．1 に示す。

この実験結果より、選別感度が低いと、もみ混入率が少ない場合においても、もみ除去率は低い。しかしながら、選別感度が高く設定すると、もみ除去率は向上する。また、もみ混入率が低くなると、もみ除去率は高くなる。

＜実験条件＞

米の供給量：349Ｋｇ／Ｈ、感度：選別感度

表 2．1 着色米除去率 1

もみ混入率	3%	2%	1%	0.40%	0.10%
感度					
20	53.5%	47.7%	47.9%	62.8%	58%
30	80.3%	75%	77.3%	85.2%	84.7%
40	90.8%	89.3%	90.5%	96.6%	96%
50	96.4%	95.9%	96.8%	99.6%	99.3%
60	98.2%	97.9%	96.5%	99.3%	99.3%
70	97.2%	98.8%	97.3%	99.3%	100%
80	96.5%	97.8%	96.6%	98.9%	100%

（2．1．2）性能実験 2

性能実験 2 では、選別感度を一定に設しておき、米の供給量を変化させてのもみ除去率を検証する。実験結果は、表 2．2 に示す。

＜実験条件＞

供給量　：　①40：105Kg　/　H
②50：144Kg　/　H
③60：224Kg　/　H
④70：261Kg　/　H
⑤80：349Kg　/　H

選別感度：20、40、もみ混入率：3%

表 2．2　着色米除去率 2

供給量	40	50	60	70	80
感度					
20	76.5%	84%	73%	57%	53.5%
40	99%	98.5%	97.4%	94.8%	90.8%

この実験結果より、米供給量が多くなると選別感度を高く設定しておいても、もみ除去率は低い。

＜2．2＞ 問題点

性能確認の実験の結果から現状の米用色彩選別機の問題点は、選別感度 30 以下に設定した場合に、もみ除去率が 40%から 70%と非常に低い結果になり、製品として成立しない。逆に選別感度が 40 以上になるとどのもみ混入率においても、もみ除去率は 90%以上になる。しかしながら、米の供給量が増加すると、もみ除去率はどの選別感度においても、もみ除去率は低い。

この性能実験において、米の供給量は実際の選別機で取り扱われる供給量の半分以下に設定している。したがって、実際の米用色彩選別機では、取り扱う米の供給量がこの性能実験時と比較して倍以上になるため、選別感度を 40 以上に設定した場合においても、もみ除去率は 90%を割る恐れがある。

つまり、高い除去率を維持するためには、米の供給量を少なくしなけらばならないが、供給量を少なくすれば米すべての監査終了までの時間が多く費やされる。逆に、米の供給量を増やし監査終了時間の短縮を図れば、もみ除去率が下がってしまう問題点が生じる。

また、米も 1 粒ずつ切り出すことが不可能なので、仮に被害粒と選別しても空気銃で排除する際に周辺の良品の米粒もまとめて排除する恐れがある。

3. NN で構成する色彩選別機

2. で現状の米用色彩選別機は、大量の米の選別を行う際にもみ除去率が低下し、もみ除去率を高く維持するために米の供給量を少なくしなければならないという落下流量の問題点がある。本研究では、このような問題点を解決するために、紙幣識別、掌紋、筆圧による個人認証などで、有用性が示されている非線形識別手法の NN [(1)～(15)] による選別手法と線形識別手法のテンプレートマッチング選別手法を融合させたニューロテンプレートマッチング選別手法 [(1)、(2)] を使用して米の選別を行なうことを提案する。また、現状の選別機では処理できなかった米を一粒ずつ切り分けるラベリング処理を応用したアルゴリズムを提案する。さらに、提案手法と米の切り出しアルゴリズムをＤＳＰを核とする自律型ニューロボード [(3)] に搭載し、新たに色彩選別機に搭載することも提案する。このニューロボードを搭載することにより、米画像データの学習、選別が可能になる。また、この章では、提案した選別手法と融合させた新しい米用色彩選別機の基本構成や色彩選別機全体の制御フロー、NN の構成、色彩選別機内の米を一粒ずつ切り出すソフトウェア部分について説明する。

＜3.1＞新しい色彩選別機の基本構成

NN で構成する色彩選別機の基本構成は、米を撮影するラインセンサ部分や米が落下してくるシュータ部分、米の除去を行なうための空気銃などである。提案する新しい色彩選別機は、入力された米の画像データを学習、選別を行ない、大量の米が撮影されている画像データから 1 粒ずつの米の画像データに切り出すことが可能である、図 3. 2 に示す自律型ニューロボードやその自律型ニューロボードの諸設定を行なうＰＣなどを選別機の内部に搭載する。

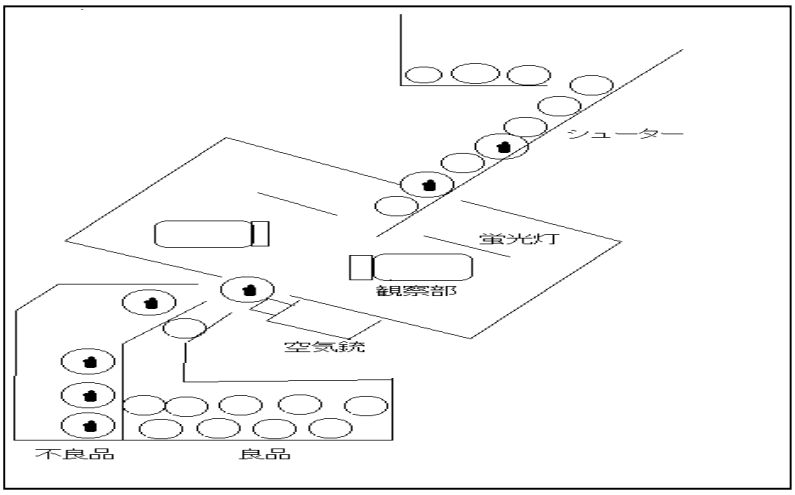


図 3. 1 機構概要

図 3.2　自律型ニューロボード

図 3.3　自律型ニューロボード内部

＜3.2＞色彩選別機全体の制御フロー

この節では、提案手法を融合させる米用色彩選別機全体の制御の流れについて説明する。

米を選別機に投入する前に、判断基準となる米（良品粒、被害粒）の画像データを選別するニューロボードであらかじめ学習する。つぎに、実際に選別機内のシュータを自由落下してくる米をラインセンサで取込む。ラインセンサで取込んだ米の画像データを 1 フレーム（取込単位）に変換する。その後、判断基準となる米 1 粒の画像を切り出す。その際、フレームを基準にした画像切り出す手法ではなく、米 1 粒を基準にした切り出す手法を用いる。切り出した画像には位置情報を付けておく。あらかじめニューロボードに学習させておいた米の画像データを基に、位置情報付きの米 1 粒画像データを選別する。最後に、選別した結果、良品と選別された米はそのまま落下し米袋に集められる。被害米と選別された米は、位置情報を基にしてその部分に空圧を当て除去を行なう。

＜3.3＞各パーツ概要

この節では、本研究において学習、選別に使用する NN の構成や米の切り出しに使用するソフトなど、色彩選別機のソフトウェア部分について説明する。

（3.3.1）米の切り出し

この項は、米を 1 粒単位で切り出すソフトウェアについて説明する。ただし、このソフトウェアは、実システム装置で使用するラインセンサで撮影した画像を使用せず、研究室内のＵＳＢカメラで撮影した米の画像データを使用して、米の切り出しを実施する。

＜アルゴリズム＞

この切り出しソフトのアルゴリズム [16] ～ [18] は以下の通りである。

まず、256 色ＢＭＰファイルから米粒画像データを取り出す（処理の制限上ＢＭＰファイルとして取り扱えるのは 256 色ＢＭＰファイルのみ)。RGB の内、青だけ取り出して画像処理を行なう。

つぎに、しきい値を用いて、画像データを 2 値化する。

（a） しきい値以上の画素を“ 1 ”画素とする。

（b） しきい値より小さい画素を“ 0 ”画素とする。

ここで、“ 1 ”画素は米の部分で“ 0 ”画素は他の背景部分になる。しきい値は、BLUE の色レベルで決定する。

さらに、画像内のノイズを除去するために 8－近傍収縮処理を行なう。収縮処理に関しては、収縮回数を 1 回から 100 回までの内から任意に決めることが可能である。8-近傍収縮処理を行なうために以下のラベル処理を行なう。

(ⅰ) “ 1 ”画素をさがす。

(ⅱ) さがした“ 1 ”画素に、現在のラベル番号を付ける。

(ⅲ) 現在のラベル番号の 8 近傍に“ 1 ”画素があれば、その画素にラベル番号を設定する。

最後に、求めたラベル領域ごとに、領域の位置の最大値、最小値を求める。
その後、各々のラベル領域の中心位置を求める。

＜処理の制限＞

この切り出しソフトの処理の制限として、抽出できるラベル領域の個数は、最大で 30 個である。画像ファイルの大きさの制限は、最大幅 320 画素、最大高さ 200 画素である。

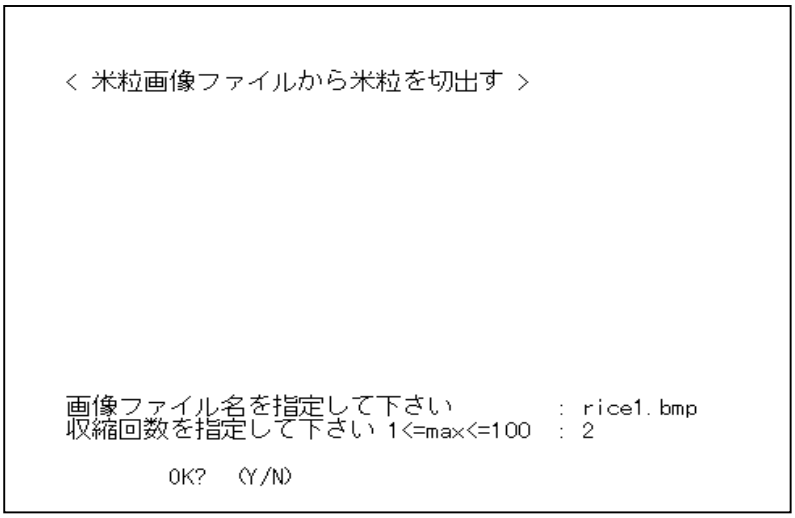


図 3．4　切り出しソフト入力画面

＜性能確認実験 1＞

ここでは、切り出しソフトの性能確認の実験を実施し、その結果について検討する。
実験条件は以下のように設定する。

① 米の撮影時の隣り合う米の内接間距離を測定する。
② お米は縦に平行に並んでいる 2 粒のみの距離間を測定する。
③ 収縮回数は 1 回

性能実験の結果は表 3．1 に示す。

表 3．1 切り出し結果（1）

	撮影時間隔	切り出し結果
Rice1	3.0cm	○
Rice2	2.5cm	○
Rice3	2.0cm	○
Rice4	1.5cm	○
Rice5	1.0cm	○
Rice6	0.5cm	○
Rice7	0.4cm	○
Rice8	0.3cm	○
Rice9	0.2cm	×
Rice10	0.1cm	×

図 3．5　元画像（1）

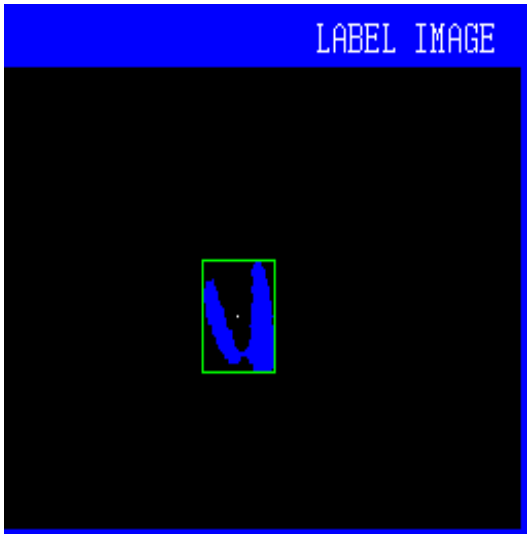


図 3．6　切り出し結果（1）

表 3．1 より、収縮回数が 1 回の場合、隣接間距離が 0．2ｃｍ以下になると米を 2 粒に切り出すことが不可能になる。

＜性能確認実験 2＞

　ここでは、先程の性能確認実験 1 の時より、収縮回数を増やし同様の性能確認実験を実施する。

図 3．7　元画像（1）

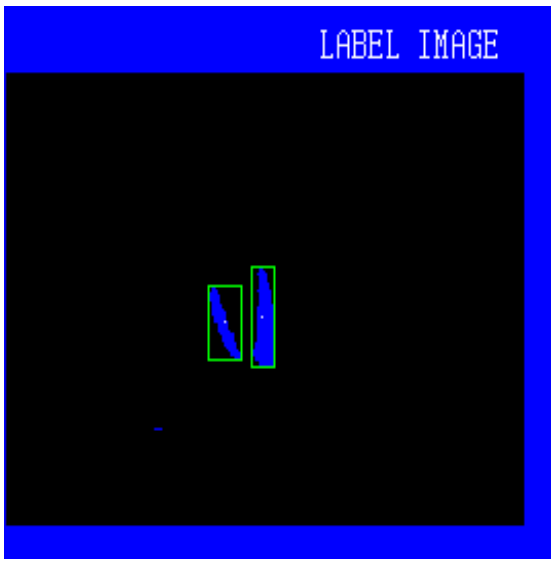


図 3．8　切り出し結果（2）

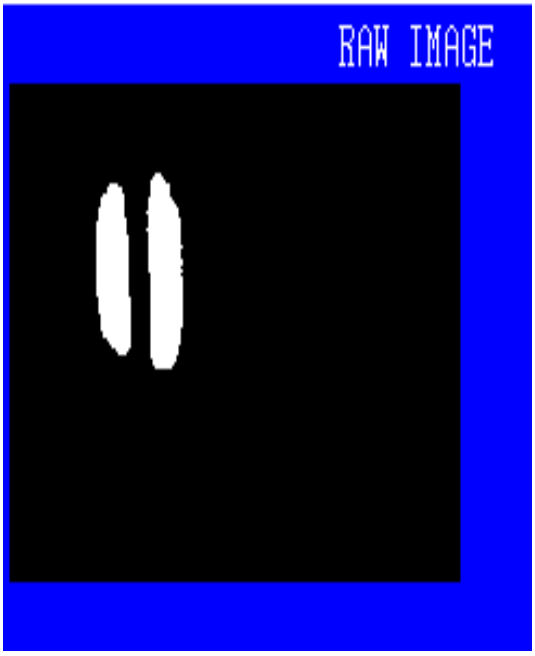


図 3．9　元画像（2）

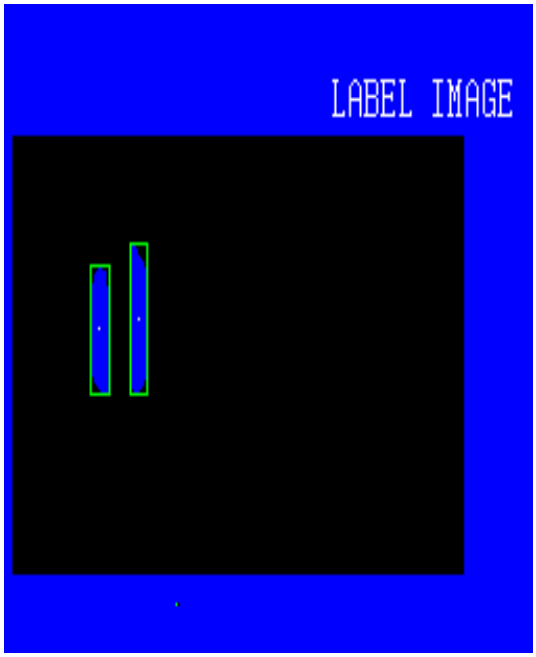


図 3．10　切り出し結果（3）

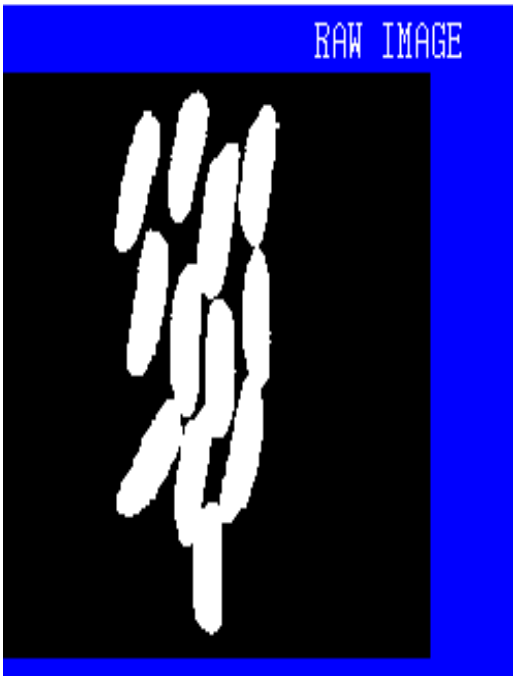


図 3．11　元画像（3）

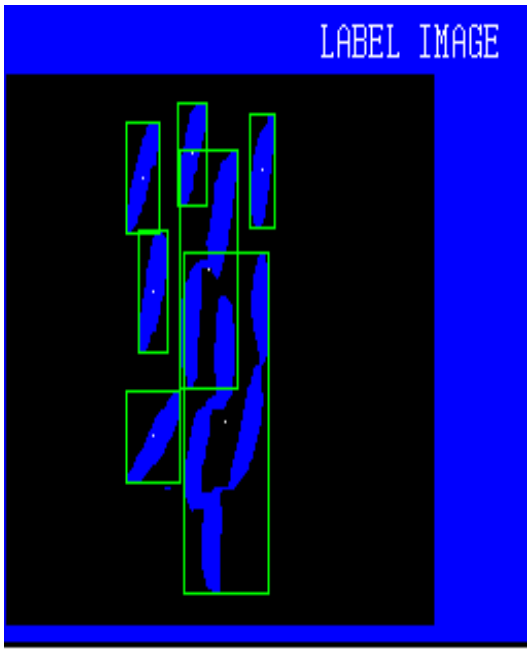


図 3．12　切り出し結果（4）

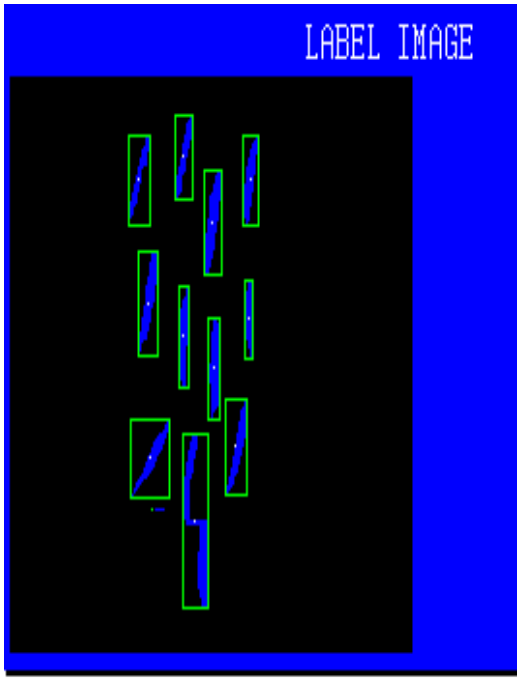


図 3．13　切り出し結果（5）

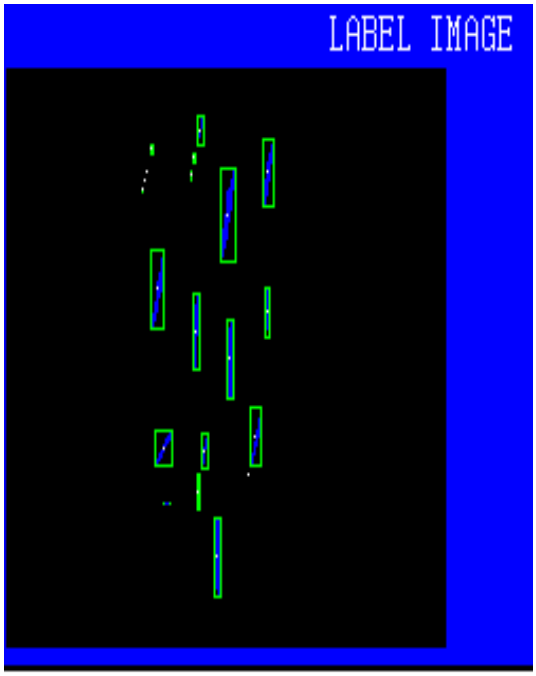


図 3．14　切り出し結果（6）

実験条件は以下のように設定する。

① 米の撮影時の隣り合う米の内接間距離を測定する。

② お米は縦に平行に並んでいる 2 粒のみの距離間を測定する。

③ 収縮回数は 2 回

性能実験の結果は、表 3．2 に示す。隣り合うお米の間隔が 1mmの場合においても正確にお米を切り出すことが可能である。

表 3・2 切り出し結果（2）

	撮影時間隔	切り出し結果
Rice16	1.0cm	○
Rice17	0.9cm	○
Rice18	0.8cm	○
Rice19	0.7cm	○
Rice20	0.6cm	○
Rice21	0.5cm	○
Rice22	0.4cm	○
Rice23	0.3cm	○
Rice24	0.2cm	○
Rice25	0.1cm	○

また、実際の落下状態を図 3．11 で示すように擬似的に再現し、切り出し実験を実施する。正確に、1 粒ずつお米を切り出せる収縮回数は平均 4 回程度である。ただし、

収縮回数を多くすると、図 3．14 で示すように切り出し領域が小さくなりすぎて、必要以上に米を切り出す。そのため、今後、切り出す領域の面積にあらがじめ制限を付けて米を必要以上に切り出さないようにする。

（3.3.2）NN の構造

この項では、開発システムに搭載する選別手法の基礎となる NN の基本構造について説明する。

NN の計算メカニズム[15]には、大雑把に分類して 2 つある。1 つは、ネットワークが階層的なネットワーク（層状結合ネットワーク）と、もう 1 つは相互結合のある非階層的なネットワークである。本研究でシミュレーションに使用し、実システムに搭載する NN の計算メカニズムは階層的ネットワーク―Multi layer Perceptron(MLP)型ネットワークである。

＜MLP 型ネットワーク＞

ここでは、MLP 型ネットワークについて説明する。

ネットワークはいくつかの層からなる階層的なもので、各層は適当な数のユニットからなり、各層内の結合はなく、層間の結合は入力層（第 1 層）から出力層（最終層）へ向けての一方向の結合のみとする（feedforward type）。入力層を除く各層のユニットは、前の層のユニットからの重み（ウエイト）付き入力を受けて、その総和を計算し、それに適当な関数 f をかけたもの出力する。すなわち、$I^k{}_I$ 、$O^k{}_I$ をそれぞれ第 k 層の第 I ユニットの入力の総和、出力とし、$w^{k-1}{}_j{}^k{}_I$ を第 k－1 層の第 j ユニットから第 k 層の第 I ユニットへの結合のつよさ、$\theta^k{}_I$ を第 k 層第 I ユニットにしきい値とすると、

$$I^k{}_I = \Sigma\, w^{k-1}{}_j{}^k{}_I\, O^{k-1}{}_j - \theta^k{}_I$$
$$O^k{}_I = f\,(I^k{}_I) \qquad (3．1)$$

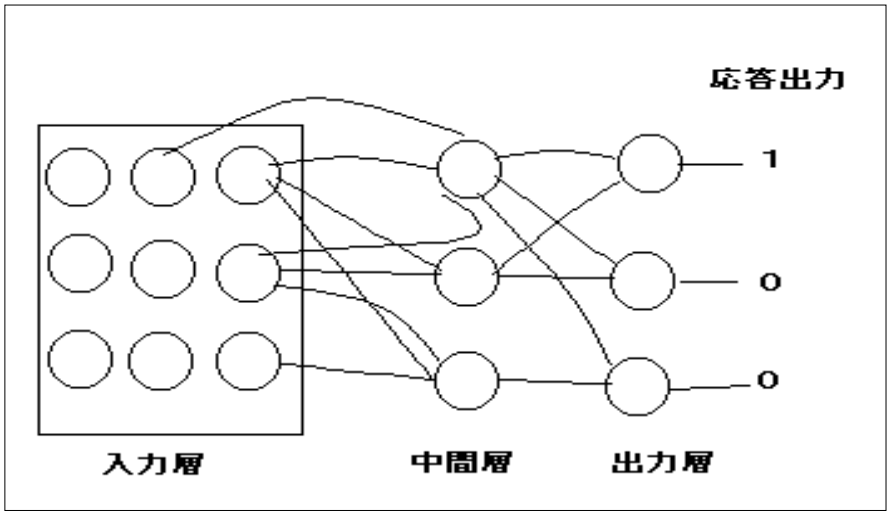
ユニットの入出力関数 f としては、しきい関数、区分線形関数、ロジスティック関数、恒等関数などがある。このうち、本研究ではロジスティック関数（シグモイド関数）を使用する。

$$f(x) = \frac{1}{1+\exp(\frac{-x+\theta}{T})} \qquad (3．2)$$

式（3．2）において、x は各ユニットへの入力値で、f（x）はそのユニットの出力値です。T はネットワークの温度と呼ばれる正の数で、T が大きくなるほどグラフはなだらかになりす。θ はユニット単位のしきい値である。

このようなネットワークでは、入力層にパターンを提示すると、それが次々と層をを進むごとに変換され、最終的には出力層の出力パターンが変換結果として得られる。

フィードバックがないために、回路網での振舞いは比較的単純だが、中間層のユニット数を増やすことで、かなり複雑なパターン変換が可能になる。
また、望ましい変換を得るための学習アルゴリズムにも強力なものがある。



*処理の流れは左から右へ

図 3．15　3 層型 NN

＜学習アルゴリズム＞

ここでは､実際の学習アルゴリズムである誤差逆伝搬アルゴリズム[15),(16)]について説明する。

具体的には、いくつかの入力パターンの例（学習データ）を与え、その時の出力パターンと期待する出力パターン（教師値）との誤差が減少するようにウェイトを修正する。

ある入力パターンを与えた時の出力層の第 j ユニットの出力を Oj､このときの出力層の第 j ユニットの期待値（教師値）を Tj とすると、第 j ユニットの誤差 Ej はつぎのようにして求める。

$$E_j = \frac{1}{2}(T_j - O_j)^2 \qquad (3．3)$$

したがって、1 つの学習パターン P における出力層の誤差 Ep は次のようにして求める。

$$E_p = \frac{1}{2}\sum_j (T_j - O_j)^2 \qquad (3．4)$$

全学習パターンの誤差の総和を E とし、総合誤差と呼ぶことにする。

$$E = \sum_p E_p = \frac{1}{2}\sum_p \sum_j (T_j - O_j)^2 \qquad (3．5)$$

上記の総合誤差 E が最小になるように各ユニット間の結合の重み（ウェイト）を修正する。

修正の方法としては最急降下法を用いる。具体的には、各学習パターン毎に、各ユニットの誤差 Ej が最小になる方向へ微小な変更を加えていく。

第 t 回目の学習における、k-1 層の第 I ユニットから k 層の第 j ユニットへのウェイト Wij の修正量Δwij(k-1,k)(t)は以下のようにして求める。

$$\Delta W_{ij}^{k-1,k}(t) = -\varepsilon \delta_j^k O_j^{k-1} + \alpha \Delta W_{ij}^{k-1,k}(t-1) + \beta \Delta W_{ij}^{k-1,k}(t-2) \qquad (3.\ 6)$$

上の式において”ε”は学習定数、”α”は慣性定数、”β”は振動定数である。

また、”δj(k)”は k 層の第 j ユニットの一般化誤差で、k 層が出力層の場合と中間層の場合によって算出方法が異なる。以下に一般化誤差の算出方法を示す。

k 層が出力層の場合、(Ij(k)は k 層の第 j ユニットの入力総和)

$$\delta_j^k = (T_j - O_j^k) f'(I_j^k) \qquad (3.\ 7)$$

k 層が中間層の場合、（ただし、m は出力層のユニット番号）

$$\delta_j^k = \left(\sum_m W_{jm}^{k,k-1} \delta_m^{k+1} \right) f'(I_j^k) \qquad (3.\ 8)$$

以上が誤差伝搬法によるウェイト修正の概略である。

＜学習パラメータ＞

先に示したウェイトの修正式（3．6）における、”ε”学習定数、”α”慣性定数、”β”振動定数は環境設定ファイルで設定する。

“ε”学習定数は大きな値にするとウェイトの修正量が大きくなり学習は早くなるが、あまり大きくすると逆に学習が収束しなくなる[13]。総合誤差が上下するときは学習定数を小さくし、誤差の減少速度が小さいときは学習定数を大きくする必要がある。この操作は学習プログラムが自動的におこなう。ユーザーは学習開始時の学習定数の初期値を設定する。初期値は（0.1< ε <1.0）の範囲で設定してしなければならない。デフォルト値は 0.5 である。

“α”慣性定数は総合誤差の振動を減らし、学習の収束を加速させる働きをする。

“β”振動定数は総合誤差を上下に振動させて極小値から脱出させる働きをする。

αとβには関連性[13]があり、以下の範囲（塗りつぶした部分）で設定してしなければならない。

慣性定数“ α ”のデフォルト値は 0.95、振動定数“ β ”のデフォルト値は‐0.1 である。

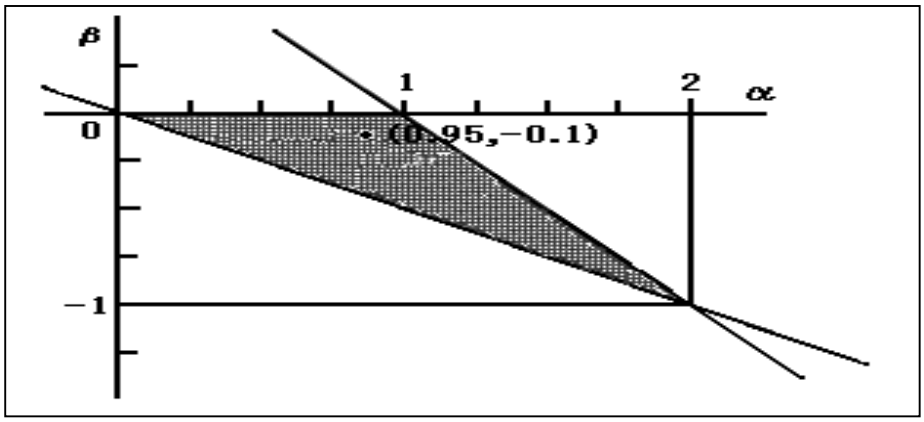


図 3．16　α とβの関連図

（3.3.3）ニューロテンプレートマッチング選別手法

この項では、提案手法のニューロテンプレートマッチング選別手法について説明する。

ニューロテンプレートマッチング選別手法は、非線形識別手法の NN による選別手法と、線形識別手法のテンプレートマッチング選別手法を融合させた選別手法である。

その基本的な構成法[(1)、(2)]は、個々の選別パターンごとのテンプレートを NN で構成し、非線形テンプレートによるマッチング処理を行なう。NN の構造は、（3．3．2）で説明したように階層型のＭＬＰ型 NN である。層の数は、非線形識別が可能な最小構成の 3 層構造である。なお、各テンプレートに対応する NN は目的とする選別パターン（目的パターン）と目的としない選別パターン（目的外パターン）の 2 つだけを分離する機能を有する。また、NN の出力ユニットの構成法は 2 種類ある。1 つは、選別パターンに対し出力層に目的ユニットと目的外ユニットの 2 つを設定する方法である（以下 NN2 と略記）。もう 1 つは、出力層に 1 つのユニットを設定し、あるしきい値より出力値が大きければ目的ユニット、出力値が小さければ目的外ユニットと判断する手法である（以下 NN1 と略記）。

NN の入力値として画像データを使用する際、複数の画素の有効／無効を選択する必要がある。この有効／無効を表すデータをマスクデータと呼ぶ。また、マスクに基づいて有効なマスクの画素値を総和したものをスラブ値とする。

＜NN1 の基本構成＞

ここでは、NN1 の基本構成について説明する。入力ユニットは 50 種類のマスク[(4)~(11)]により特徴抽出されたお米のスラブ値[(14)]を入力として対応させるためユニット数は 50、中間層のユニット数は 20 とする。出力ユニット数は 1 とする。よって、NN の構成は 50×20×1（入力層ユニット数×中間層ユニット数×出力層ユニット数）の 3

層の階層型ネットワークとなる。テンプレートに関しては、米の種類数の分だけテンプレートを用意する。

これは、目的パターンのテンプレートのみで、学習・認識を実施すれば、無数に存在する目的外パターンを 1 つのカテゴリにすることは学習の面から考慮すれば得策ではないからである。そこで、シミュレーションなどで使用する学習データは、以下の表 3．3 に示す。米の種類は良品である標準米と被害米 1、被害米 2 の 3 種類を用意する。被害米 1 と被害米 2 の違いは、被害米 1 は、日本米の被害粒とインド米の被害粒をランダムに用意したもので、被害米 2 は、日本米の被害粒、インド米の被害粒をそれぞれ整理したものである。

表 3．3 学習データ

テンプレート	目的パターン	目的外パターン
標準米	標準米	被害米1+被害米2
被害米1	被害米1	標準米+被害米2
被害米2	被害米2	標準米+被害米1

NN1 による学習において、目的テンプレートに対して目的パターンを提示した場合は“ 1 ”を教師し、目的外のパターンを提示した場合は“ 0 ”教師する。認識結果の示し方は、何らかのしきい値を試行錯誤で設定し、このしきい値を用いて目的パターンか目的外パターンかを判断する。

例をあげると、標準米のテンプレートで標準米を選別すると、教師信号“ 1 ”を提示、被害米 1、被害米 2 を選別すると、教師信号“ 0 ”を提示する。

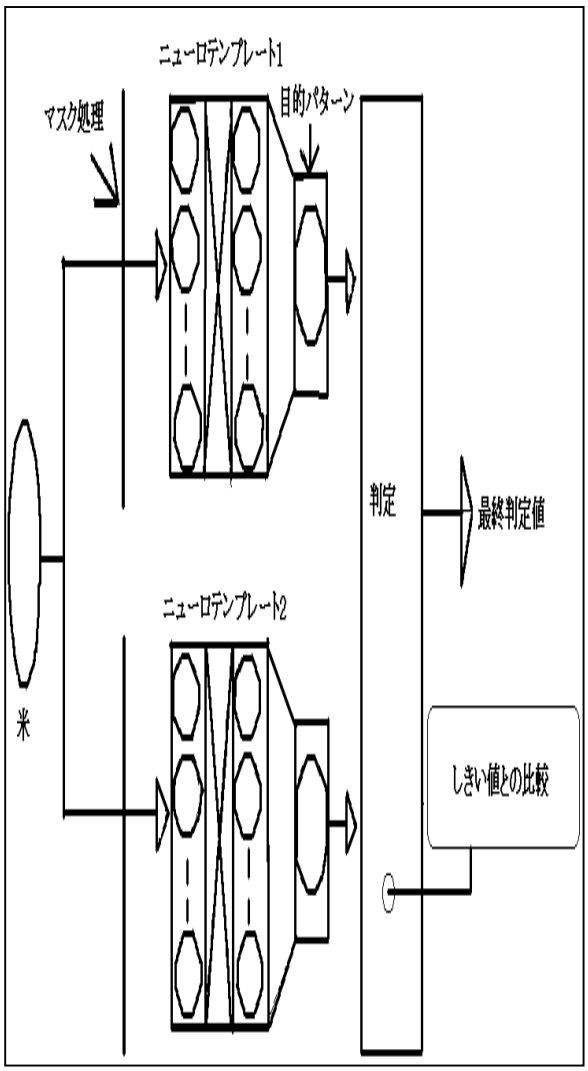


図 3．17　NN1 の構成図

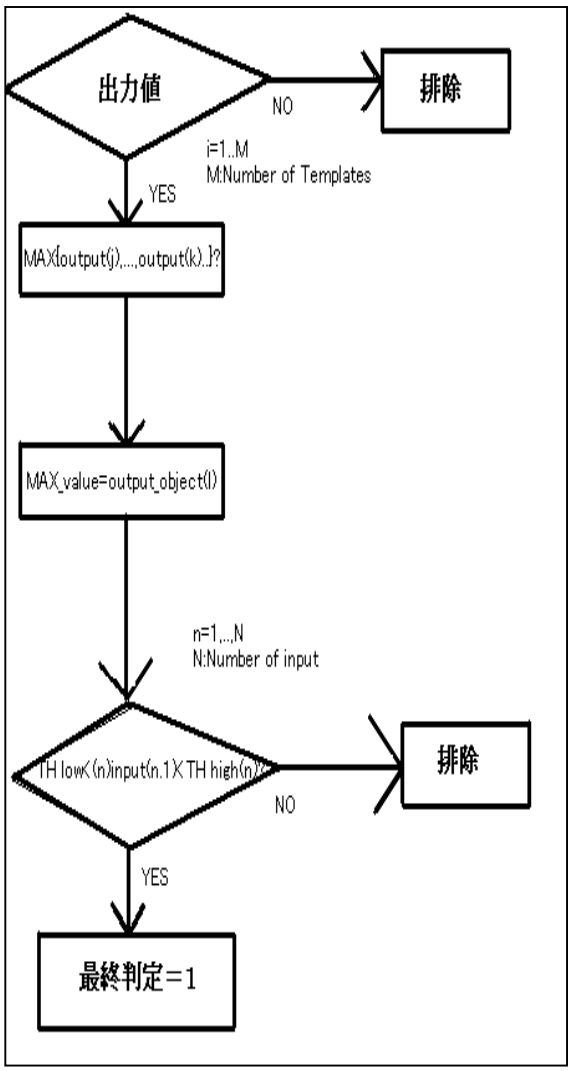


図 3．18　最終判定フローチャート（NN1）

＜NN2 の基本構成＞

ここでは、NN2 の基本構成について説明する。個々の NN の構成は 50×20×2（入力層ユニット数×中間層ユニット数×出力層ユニット数）の 3 層の階層型となる。また、テンプレートの数は NN1 と同様で米の種類数と同じである。

NN2 による学習において、目的パターンのデータを提示した場合、目的パターンに対応するユニットには“1”を、目的外パターンに対応するユニットには“0”を教師する。また、目的外パターンの提示した場合、目的パターンに対応するユニットには“0”を、目的外パターンに対応するユニットには“1”を教師する。認識結果の示し方は、NN1 と違い目的パターンの値と目的外パターンの値の大小関係で判断する。つまり、目的パターンのユニット値が目的外パターンのユニット値より、大きい場合のみ正解とする。

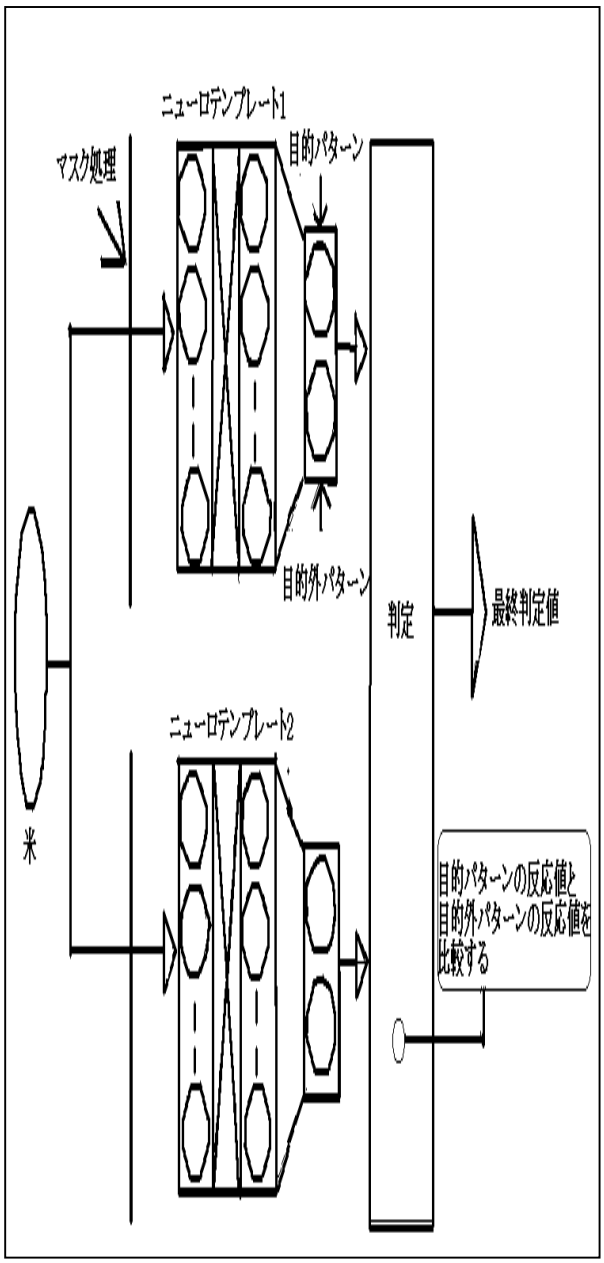


図 3．19　NN2 の構成図

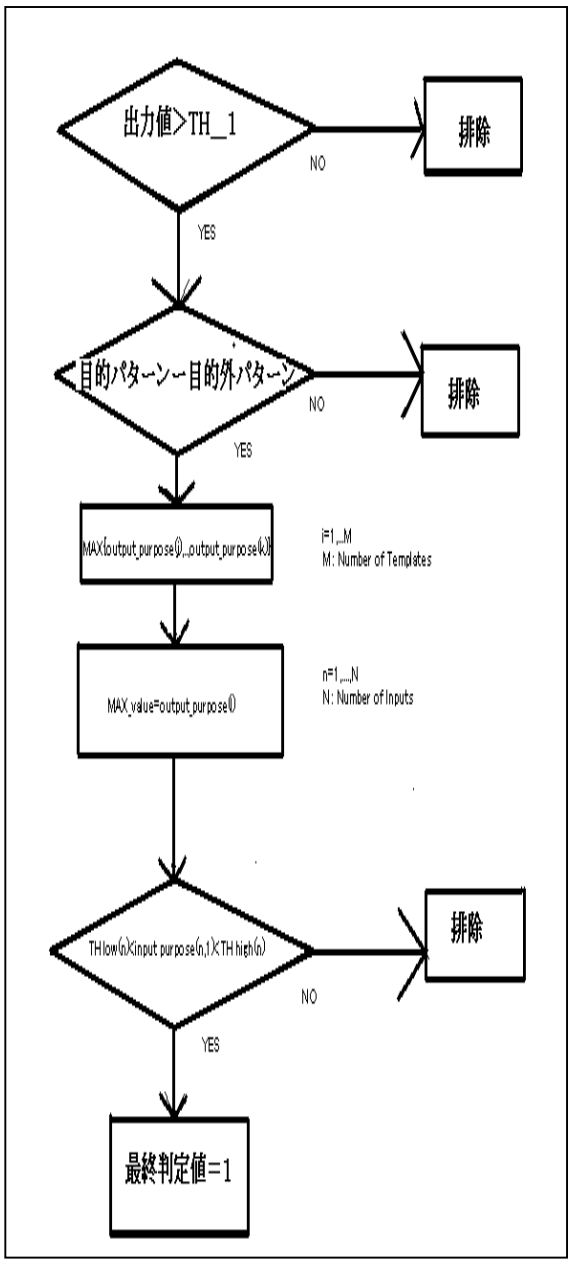


図 3．20　最終判定フローチャート（NN2）

とくに、NN2 を使用した場合、ノイズを多く含んだ未学習データに対する NN の挙動をユニットの反応値の大小関係で明確に見極めることが可能になると予想される。また、反応値の大小関係で選別を行なうので NN1 のようにしきい値を用いる必要はない。

ただし、NN の規模はウエイトの個数などを考慮すると、NN1 のウエイトの個数が（50×20）+（20×1）であるのに対し、NN2 では（50×20）+（20×2）となり、NN2 のテンプレートが大きくなる不都合さも併せ持つことになる。

4．シミュレーション

この章では、実際の米を使用し、3．で提案した選別手法である NN1、NN2 による米選別の学習と未学習データによる選別実験をシミュレーションで実施する。さらに、このシミュレーションでの結果を基に NN1、NN2 の両者の内どちらが米の色彩選別に適しているか比較検討を行なう。このシミュレーションで使用する選別対象の米は、（株）セイレイ工業内の従来の米用色彩選別機によって選別されたインド米の標準米と被害米、日本米の標準米と被害米を使用する。

＜4．1＞実験手順

この節では、シミュレーション用の米のデータ採取から NN1、NN2 用学習シミュレーションシステムを起動させるまでの手順と学習アルゴリズムについて説明する。なお、米の撮影時において実際の色彩選別機で使用するラインセンサは、今回のシミュレーションでは使用できないため、代用として研究室内のＵＳＢカメラを使用する。

（4．1．1）データ採取手順

実際の米からシミュレーション用画像データは以下の通り採取する。
まず、図 4．1 に示す撮影用筐体とＵＳＢカメラを用いて米を撮影し（図 4．2)、撮影した画像データを図 4．3 に示すグレースケール、256 階調のビットマップファイルに変換する。米の画像データの数は、標準米・被害米 1 各 100 枚ずつ被害米 2 を 210 枚撮影し、計 410 枚用意する。その後、グレースケール、256 階調のビットマップデータに変換した画像データを NN1、NN2 学習シミュレーション用画像データに変換する。その際、SCANDEMO.EXE というデータ変換用プログラムを起動して、画像データを図 4．4 に示すシミュレーション用の画像データに変換する。
画像データを変換した後、標準米用のデータファイル（以下 DATA01．DB と表記）と被害米 1 用のデータファイル（以下 DATA02．DB と表記）被害米 2 用のデータファイル（以下 DATA03．DB）の 3 種類のデータファイルを作成する。各ファイルには、各々100 枚ずつ（ＤＡＴＡ03．ＤＢは 210 枚）のシミュレーション用に変換された画像データが保存される

図 4．1 実験室用米撮影用筐体

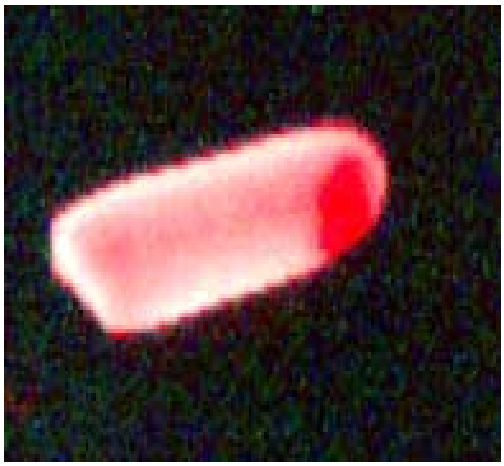


図 4．2 撮影した画像データ

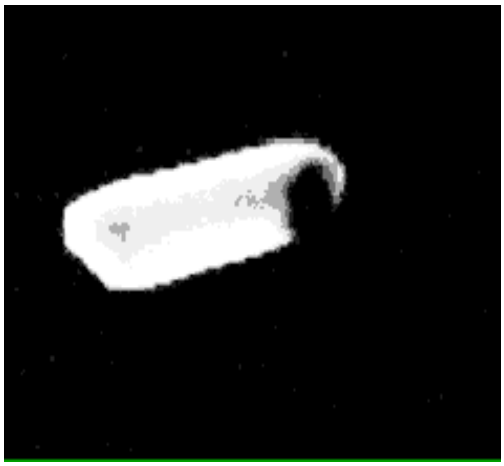


図 4．3 グレースケール・256 階調に変換した画像データ

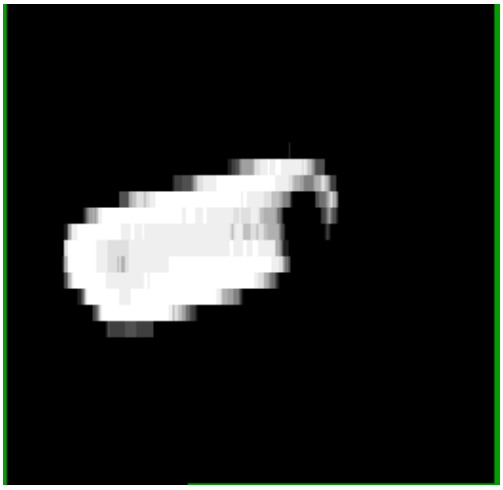


図 4．4　シミュレーション用画像データ

（4．1．2）シミュレーション手順

この項では、NN1、NN2 学習シミュレーションシステムの設定やシミュレーションの流れについて説明する。

＜設定＞

まず、シミュレーションシステムの各設定について説明する。シミュレーションシステムの設定部分は、シミュレーションシステムの内、NEURO.CFG で入力層のユニット数や出力層のユニット数、学習枚数、最大学習回数などの NN の学習・選別シミュレーションにおける基本的な設定を行なう。また、DATABASE.CFG で、シミュレーション用に変換した DATA01.DB、DATA02.DB、DATA03.DB などのデータファイルの指定を行なう。また、シミュレーションシステムの初期値などは、DEFINE.H というヘッダファイル内で設定を行なう。

＜学習シミュレーション手順＞

学習のメカニズムは、（3．3．2）で説明したように、与えられた入力に対して期待する出力が得られるようにネットワークの内部状態を調整することである。

学習シミュレーションシステムの処理の流れについて説明する。シミュレーション起動は、MODULE.BAT で行なう。シミュレーションを起動すると、先に設定した NEURO.CFG や DATABASE.CFG などの確認を行なう。それが完了すると、学習を行なうための画像データを DATA01.DB、DATA02．DB と DATA03.DB から抽出する。抽出する画像データは、NEURO.CFG で設定した学習枚数分だけを各 DATA の先頭部から抽出する。抽出した画像データの特徴を抽出するためにマスク処理 [(4)、(5)、(12)] を行なう。

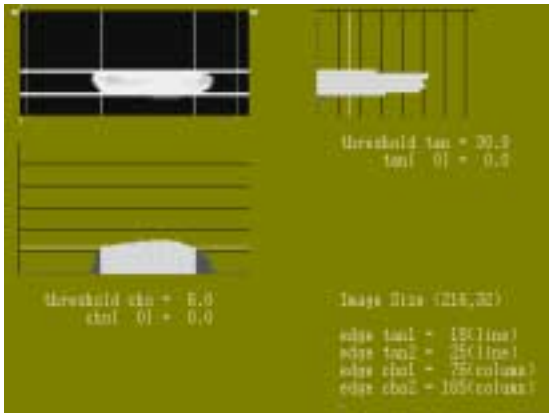


図 4．5　エッジ検出

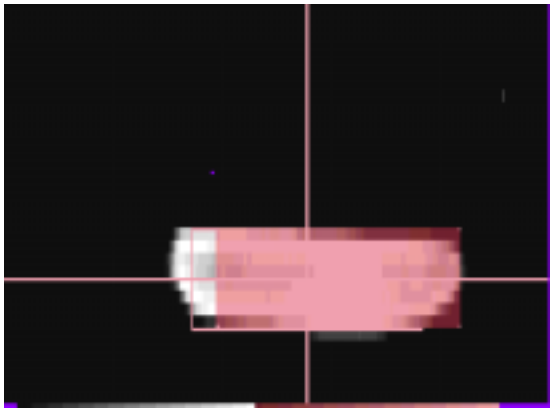


図 4．6　マスク処理画面

マスク処理の手続きは、米粒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 [(14)]）を米粒の特徴量として NN への入力としている。NN の入力層のユニット数はマスク種類数に対応している。具体的に、このマスク処理では、

まず、図 4．6 で示すようにエッジ検出を行なう。このエッジ検出は上下左右から画像の白い部分を検出し、米の形を検出する。つぎに、図 4．6 で示すように米内部に薄紅色に被覆した部分から画像データの特徴を抽出する。詳しく説明すると、抽出する画像エリアを 50 画素に分割し各画素から画像の特徴量であるスラブ値を抽出する。また、MASK.TXT でマスク処理における 1 画素の縦方向、横方向のサイズ単位、マスクエリアなどを設定する。

スラブ値を抽出、作成が完了すると、選別基準となる教師信号を作成する。

表 4．1 教師信号表（日本円の場合）

	1000	2000	5000	10000	判定
1000	1	0	0	0	
2000	0	1	0	0	
5000	0	0	1	0	
10000	0	0	0	1	
提示					

学習の基となる学習データを作成するために各画像データから抽出したスラブ値を並び替えて 1 つの学習データ（LEARN.S）を作成する。

最後に学習を行なう。

＜学習方法＞

学習方法は、先程の（3．3．2）で説明したように、いくつかの学習データを与え、その時の出力パターンとこちらが期待する出力パターン（教師値との誤差が減少するようにウェイト（結合間の重み）を修正する誤差逆伝搬（Error Back Propagation）アルゴリズムを使用する。また、学習は用意したニューロ用に変換したデータの種類の分だけ学習を行なう。学習シミュレーションの終了とともに米種類数と同じ数のテンプレートも同時に作成される。

＜実験条件＞

表 4．2 実験条件

米の種類	3
学習データ枚数	10
評価データの枚数	100
マスクの数	50
2乗平均誤差	1.0E-4
学習方法	改良型BP法

*ＤＡＴＡ03．ＤＢの評価データの枚数

このシミュレーションの初期条件は表 4．2 に示す。米の種類は、標準米、被害米 1、被害米 2 の 3 種類用意する。当初は、標準米と被害米 1 の 2 種類のみを使用する予定であったが、米の種類を多くすることにより正確な目的パターンと目的外パターンのデータを得るため、被害米の種類を多くする。

＜選別シミュレーション手順＞

学習シミュレーションが終了すると、最後に学習に使った画像データと未学習の画像データを用いた選別シミュレーションを実施する。

選別シミュレーションは、シミュレーションプログラム内の EVAL.BAT を起動する。選別シミュレーションにおけるテンプレート、ＤＢファイルや選別結果の出力場所などの設定は、EVAL.BAT にて行なう。

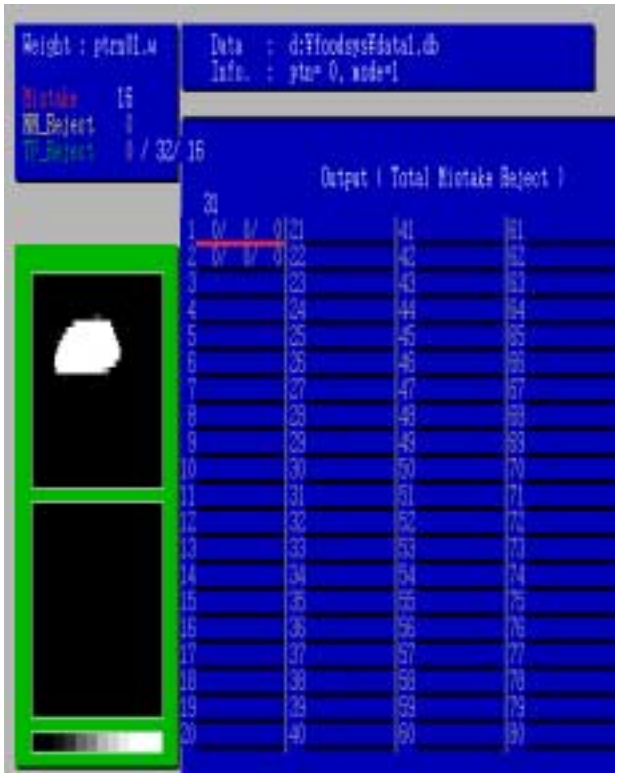


図 4．7　選別シミュレーション画面

```
0  0  0   0  0  0.992362 0.007493  1  0↵
0  0  0   1  0  0.995576 0.004168  1  0↵
0  0  0   2  0  0.992674 0.006654  1  0↵
0  0  0   3  0  0.999712 0.000227  1  0↵
0  0  0   4  0  0.992182 0.008026  1  0↵
0  0  0   5  0  0.987851 0.012294  1  0↵
0  0  0   6  0  0.998567 0.001325  1  0↵
0  0  0   7  0  0.982527 0.016408  1  0↵
0  0  0   8  0  0.996534 0.003280  1  0↵
0  0  0   9  0  0.980981 0.018685  1  0↵
0  0  0  10  0  0.993818 0.005682  1  0↵
0  0  0  11  0  0.982189 0.016386  1  0↵
0  0  0  12  0  0.998748 0.001167  1  0↵
0  0  0  13  0  0.994474 0.005195  1  0↵
0  0  0  14  0  0.961114 0.038819  1  0↵
0  0  0  15  0  0.998651 0.001219  1  0↵
0  0  0  16  0  0.974447 0.024430  1  0↵
0  0  0  17  0  0.965178 0.034223  1  0↵
0  0  0  18  0  0.993273 0.006528  1  0↵
0  0  0  19  0  0.998474 0.001391  1  0↵
0  0  0  20  0  0.327400 0.683967  4  1↵
0  0  0  21  0  0.995898 0.003884  1  0↵
```

図 4．8　ログファイル

選別結果は、ログファイル（DATA*.LOG）に出力される。

＜4．2＞NN1 による実験

（4．2．1）初期学習

この項では、表 4．2 の実験条件や＜4．1＞に基づき NN1 で構成されるテンプレートの作成とこれらによる未学習データの選別実験を実施する。最初に、米の画像データを用いて＜4．1＞で説明したようにテンプレートを作成する。テンプレートにお

いては、1 つの米の種類に対して 10 枚を学習データとする。

ただし、表 3．3 で説明したように目的外パターンは 3－1＝2 で 2 パターン存在するので実際のの目的パターンと目的外パターンに対する学習データは、それぞれ、10×2＝20 枚となる。また、選別評価のためのデータは各米の種類につき学習データを含めた 100 枚（DATA03.DB は 210 枚）とする。

＜評価方法＞

ここでは、このシミュレーションでの評価方法について説明する。
評価方法は、目的パターンを目的テンプレートで選別する方法である。具体的に説明すると、標準米の画像データは標準米テンプレートで、被害米 1 の画像データは被害米 1 のテンプレートで、被害米 2 の画像データは被害米 2 のテンプレートでそれぞれ評価を行なう。

＜実験結果＞

初期学習シミュレーションの結果は表 4．3 に示す。

表 4．3　NN1 による実験結果

米の種類	%	認識率
標準米		100%
被害米1		57%
被害米2		85%

NN1 のしきい値は 0．7 に設定。

実験結果により NN1 によるテンプレートでのパターンマッチングにおいて、標準米は、良好な選別結果を得た。しかしながら、被害米 1 と被害米 2 に関しては、期待した選別結果を得られなかった。

（4．2．2）追加学習

初期学習において被害米 1 と被害米 2 の認識率は期待した結果と異なる。そこで、この項では、初期学習で誤認識した被害米 1、被害米 2 の画像データを新たに学習用画像データとして追加し、再度学習シミュレーションを実施し、認識率の変化を測定、検証を行なう。

追加した画像データ：初期学習で誤認識した被害米 1，2 の画像データ

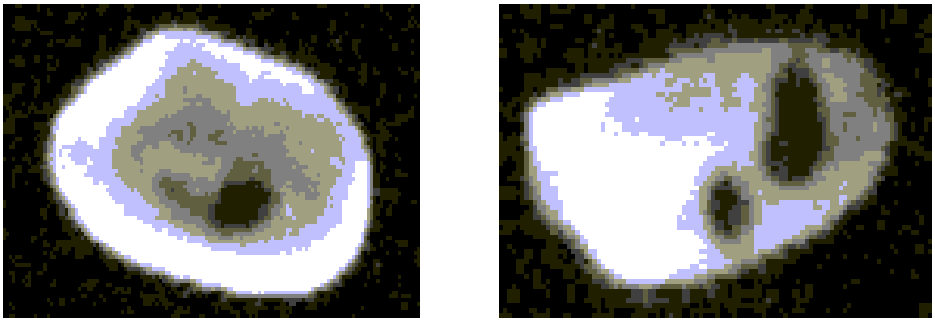


図 4．9　追加学習用画像データ

＜実験条件＞

追加学習シミュレーションの条件は、基本的には初期学習時と同じであるが、学習枚数が追加学習分を加えた 20 枚となる。

表 4．4 実験条件（追加学習時）

米の種類	3
学習データ枚数	20
評価データの枚数	100、110、220
マスクの数	50
2乗平均誤差	1.0E−4
学習方法	改良型BP法

*評価データの枚数は標準米、被害米 1、被害米 2 の順

評価方法は、初期学習時と同様で標準米画像データを標準米テンプレート、被害米 1、2 画像データを被害米 1、2 テンプレートで評価を行なう。

＜学習データ内訳＞

・標準米：
　初期学習の学習データ
・被害米 1：
　初期学習で誤認識した画像データ　＋　初期学習の学習データ
・被害米 2：
　初期学習で誤認識した画像データ　＋　初期学習の学習データ

＜評価データ内訳＞

・標準米：
　初期学習の評価データ
・被害米 1：

初期学習で誤認識した画像データ　＋　初期学習の評価データ

・被害米 2：

初期学習で誤認識した画像データ　＋　初期学習の評価データ

＜実験結果＞

追加学習シミュレーションの実験結果は表 4．5 に示す。

表 4．5NN1 の実験結果（追加学習時）

米の種類	%	認識率
標準米		100%
被害米1		93%
被害米2		99%

NN1 のしきい値は 0．7 に設定

追加学習を行なった後のシミュレーション結果は、初期学習に比べ被害米 1 と被害米 2 の認識率が向上し、NN1 によるテンプレートマッチングにおいて、良好な選別性能が得られることが判明した。

表 4．6 追加学習後の認識率の変化（被害米 1）

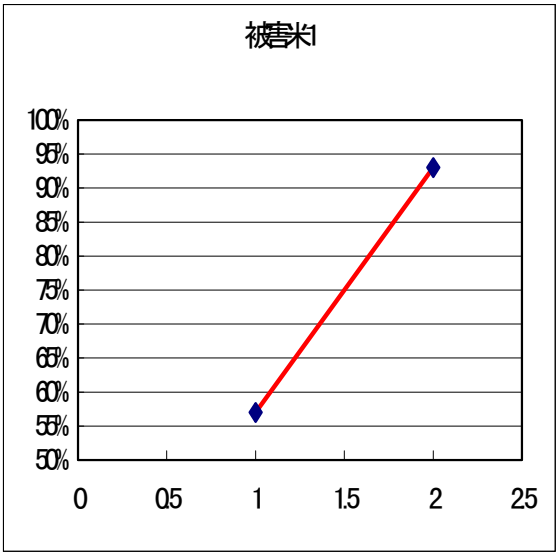
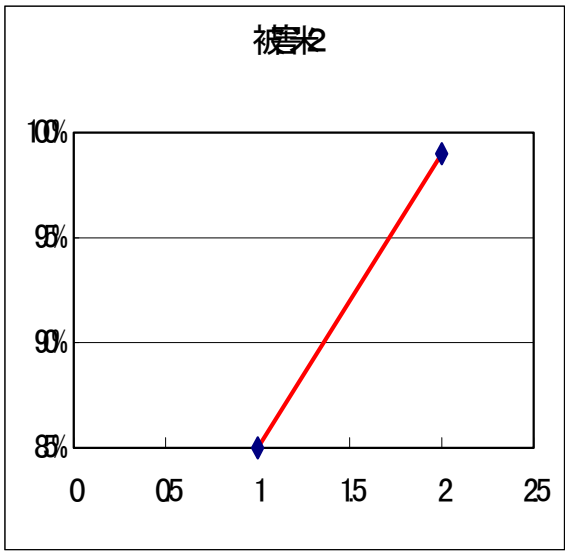


表 4．7 追加学習後の認識率の変化（被害米 2）



＜比較＞

初期学習と追加学習を認識率などで比較すると、認識率は表 4．9、表 4．10 より追加学習の方が高い認識結果を示す。

表 4．8　NN1 による認識分布の理想形

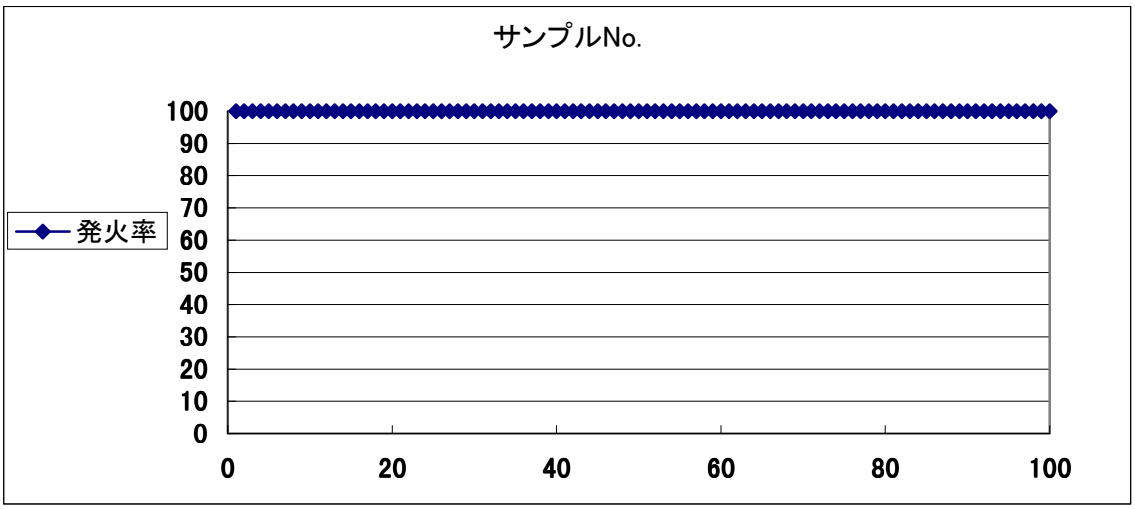


表 4．9 初期学習時の認識分布（被害米 1）

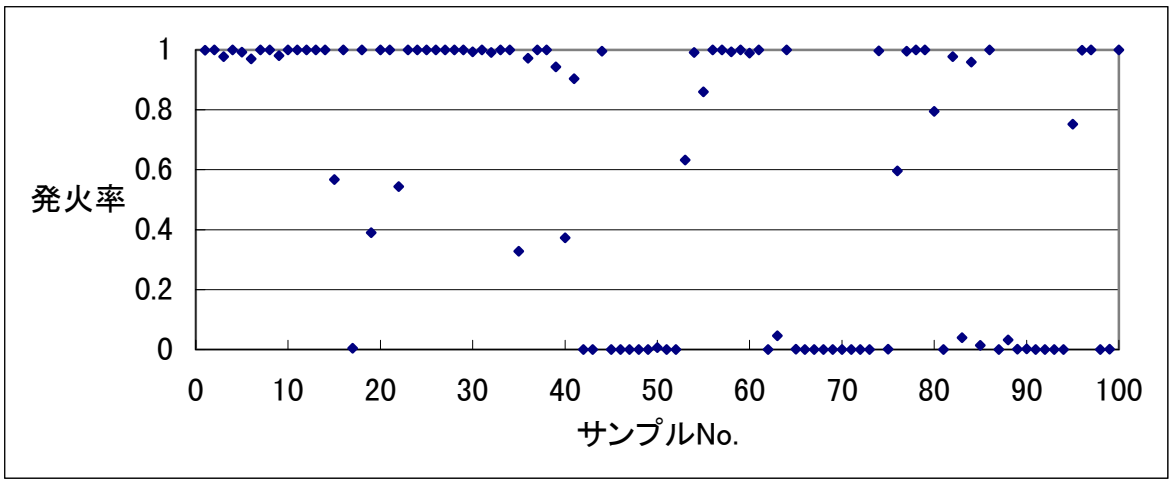


表 4．10 追加学習時の認識分布（被害米 1）

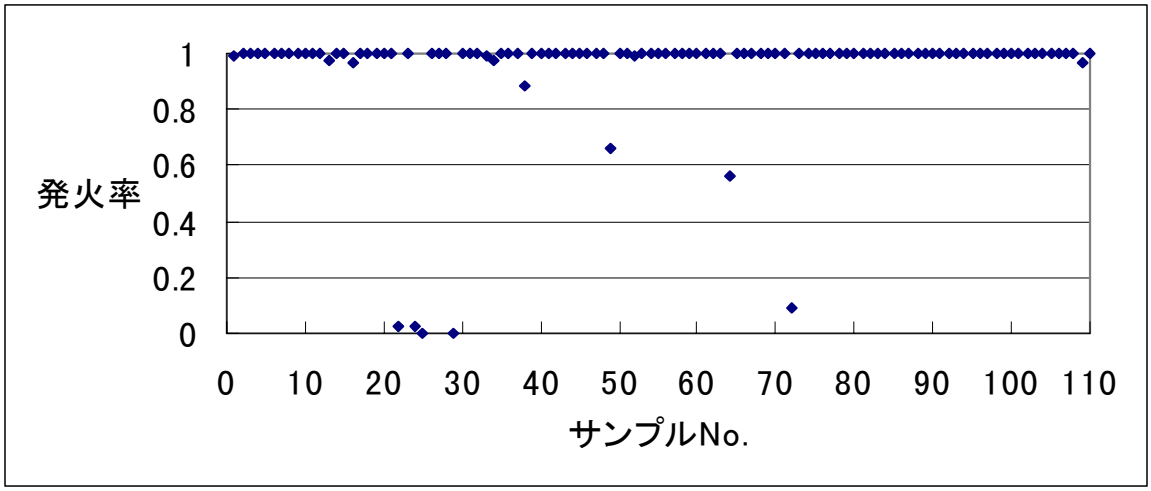


表 4．11　学習回数の比較

	初期学習	追加学習
標準米	52	22
被害米1	156	230
被害米2	106	112

表 4．11 より学習回数を比較すると、標準米は初期学習時の学習回数が 52 回で、追加学習時は 22 回と半分以下に減少し、学習かかる時間が短縮する。しかしながら、被害米 1、被害米 2 に関しては、初期学習が 156 回に対し追加学習が 230 回（被害米 1)、106 回から 112 回（被害米 2）と、いずれも学習回数が増加し、学習時間も初期学習時よりも多く費す。

＜4．3＞NN2 による実験

（4．3．1）初期学習

この項では、NN2 によるテンプレートの作成と未学習データに対する選別実験を実施する。初期学習での実験条件（表 4．2)、評価方法、および、学習データは NN1 の場合と同様である。

＜実験結果＞

NN2 の初期学習シミュレーションの実験結果は表 4．12 に示す。
初期学習における NN2 の評価のうち、標準米、被害米 1 においては NN1 の場合と同様あるいは、NN1 以上の選別性能が得られた。また、NN2 におけるシミュレーションにおいて、個々のテンプレートを作成のための学習も問題なく、実験条件で提示している 2 乗平均誤差（1.0Ｅ－4）に収束する。

表 4．12　NN2 の実験結果

米の種類	%	認識率
標準米		98%
被害米1		84%
被害米2		79%

被害米 2 の選別性能に関しては、NN1 の選別性能に比較すると若干性能が向上する。

（4．3．2）追加学習（1）

この項では、NN1 による学習シミュレーション時と同様に、初期学習において誤認識した米の画像データを学習用画像データとして追加し、再度学習シミュレーショ

ンを実施し、認識率の変化を測定、検証を行なう。

＜実験条件＞

この NN2 における追加学習シミュレーションの実験条件の内、2 乗平均誤差、学習法、マスクの数などの基本的な部分は、これまでの実験条件と同じであるが、学習枚数は初期学習シミュレーションと異なり、初期学習時に誤認識した米を学習に追加する。

ただし、DATA03.DB（被害米 2）に関しては、誤認識した米の画像データが多いため誤認識した画像データを追加学習に使用せず、学習枚数のみ増加する。

表 4．13　NN2 の実験条件（追加学習 1）

米の種類	3
学習データ枚数	20
評価データの枚数	102、117、210
マスクの数	50
2乗平均誤差	1.0E-4
学習方法	改良型BP法

*評価データの枚数は標準米、被害米 1、被害米 2 の順

評価方法はこれまでの実験と同様に標準米の画像データを標準米テンプレートで評価、被害米 1，2 の画像データを被害米 1，2 のテンプレートで評価を行なう。

＜学習データ内訳＞

・標準米：
　初期学習で誤認識した画像データ　＋　初期学習の学習データ
・被害米 1：
　初期学習で誤認識した画像データ　＋　初期学習の学習データ
・被害米 2：
　初期学習の学習データ

＜評価データ内訳＞

・標準米：
　初期学習で誤認識した画像データ　＋　初期学習の評価データ
・被害米 1：
　初期学習で誤認識した画像データ　＋　初期学習の評価データ
・被害米 2：
　初期学習の評価データ

＜実験結果＞

NN2 における追加学習（1）シミュレーション（学習枚数：20）の実験結果は表 4. 14 に示す。

表 4．14　NN2 の実験結果（追加学習 1）

米の種類	%	認識率
標準米		100%
被害米1		100%
被害米2		95%

この実験結果より NN2 での追加学習は、NN1 での追加学習同様、初期学習時よりも認識率が向上する。

（4．3．3）追加学習（2）

標準米・被害米 1 この 2 種類の米に関しては、10 枚を追加した追加学習（1）において認識率 100%と期待した結果を残した。被害米 2 に関しても認識率 95%と期待した認識結果を示した。

この項では、さらに、追加学習を実施し、認識率の変化を測定、検証をする。また、NN の学習がきちんと実験条件で示す 2 乗平均誤差に収束するか検証する。

＜実験条件＞

マスク数、2 乗平均誤差などの基本的な実験条件は、これまでの実験条件と変更はない。ただし、学習データ枚数がこれまでの 10 枚、20 枚を、さらに 10 枚増加し 30 枚にする。

学習データは、標準米・被害米 1 については、追加学習（1）で使用した学習データと同様。被害米（2）に関しては、追加学習（1）で誤認識した画像データを追加する。

表 4．15　NN2 の実験条件（追加学習 2）

米の種類	3
学習データ枚数	30
評価データの枚数	102、117、220
マスクの数	50
2乗誤差	1.0E-4
学習方法	改良型BP法

*評価データの枚数は標準米、被害米 1、被害米 2 の順

評価方法は、これまでの実験と同様に標準米の画像データを標準米テンプレートで評価、被害米 1，2 の画像データを被害米 1，2 のテンプレートで評価を行なう。

＜学習データ　内訳＞

・標準米：
　追加学習（1）の学習データ
・被害米 1：
　追加学習（1）の学習データ
・被害米 2：
　追加学習（1）で誤認識した画像データ　+　追加学習（1）の学習データ

＜評価データ内訳＞

・標準米：
　追加学習（1）の評価データ
・被害米 1：
　追加学習（1）の評価データ
・被害米 2：
　追加学習（1）で誤認識した画像データ　+　初期学習の評価データ

＜実験結果＞

NN2 における追加学習（2）シミュレーション（学習枚数：30）の実験結果は、表 4．16 に示す。

追加学習（2）では各米の選別が 100%になる。

表 4．16　NN2 の実験結果（追加学習 2）

米の種類	%	認識率
標準米		100%
被害米1		100%
被害米2		100%

表 4．17　標準米の認識率変化

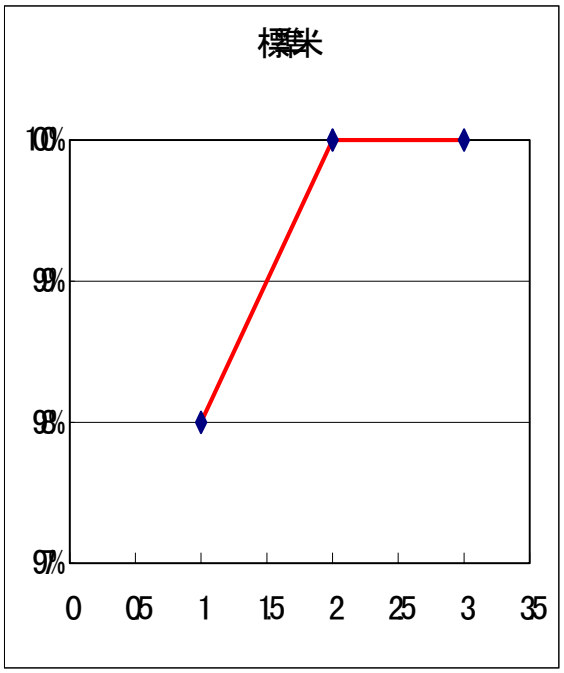


表 4．18　被害米 1 の認識率変化

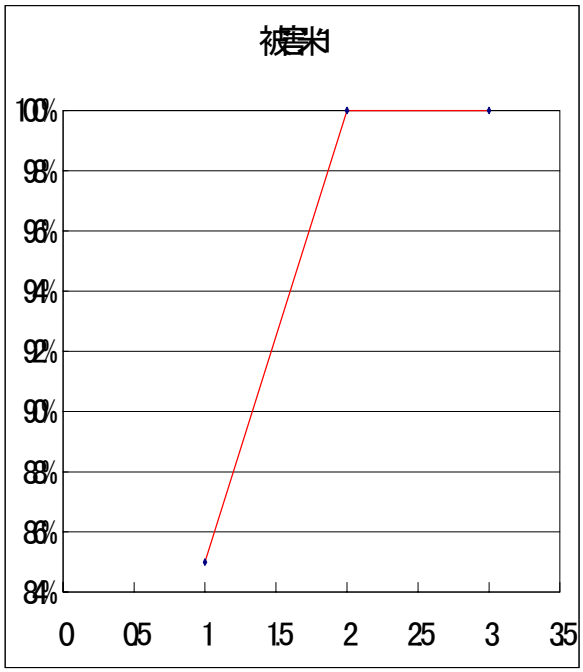
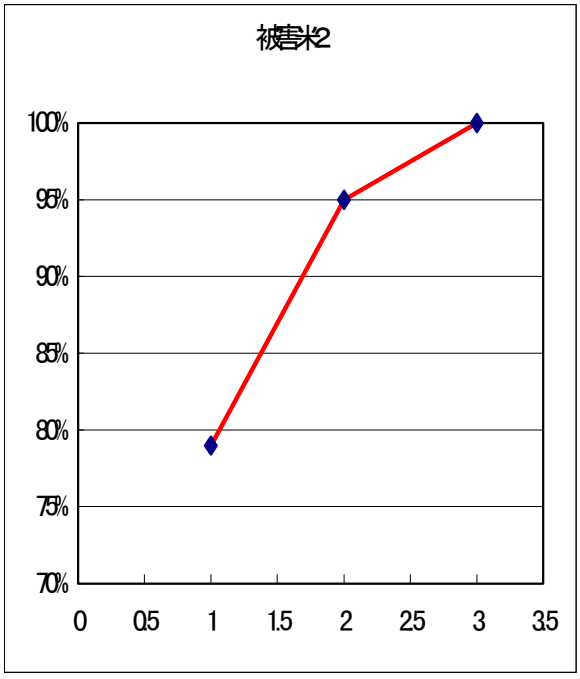


表 4．19　被害米 2 の認識率変化



＜比較＞

NN2 においても、初期学習と追加学習の比較を認識率や学習回数の観点から行なう。

認識率の点で比較すると、表 4．17、表 4．18、表 4．19 より追加学習を行なう方が、認識率は向上する。また、学習回数の変化は表 4．20 に示す。この表 4．20 より一部を除いて追加学習を行なう度に、学習回数が減り学習時間も短縮する。

表 4．20　NN2 の学習回数変化

	初期学習	追加学習(1)	追加学習(2)
標準米	47	25	167
被害米1	48	30	26
被害米2	78	26	28

表 4．21　NN2 による認識分布の理想形

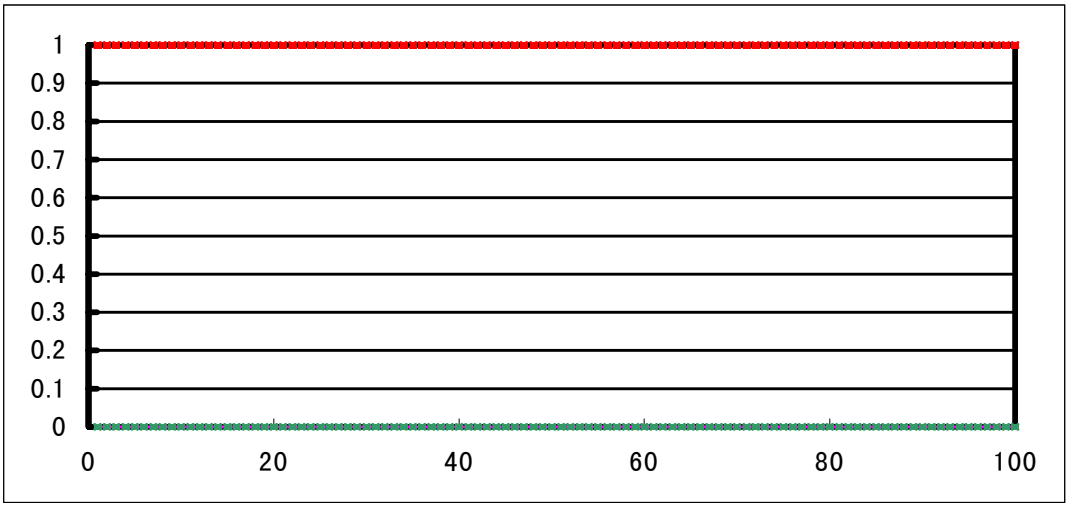


表 4．22　NN2 の初期学習時の認識分布（標準米）

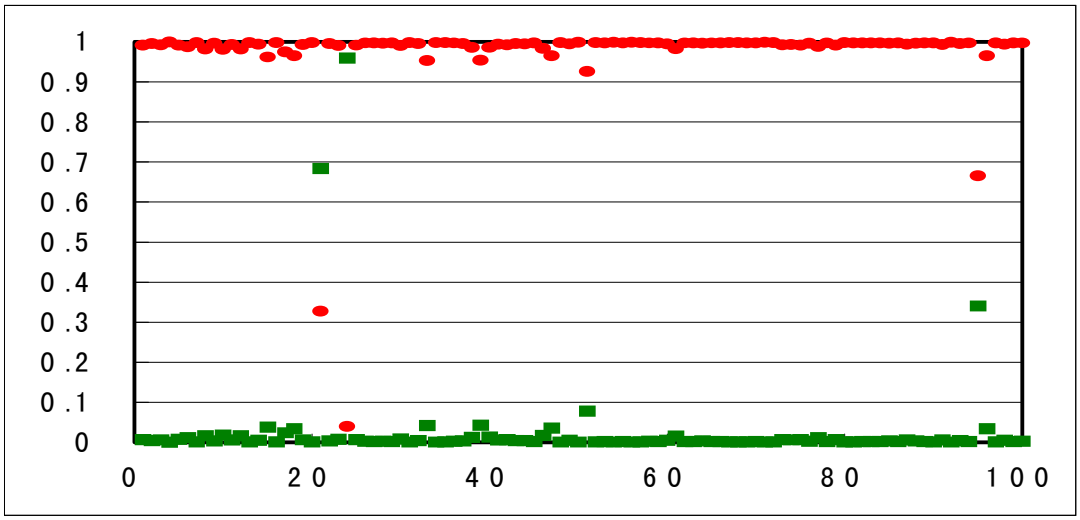
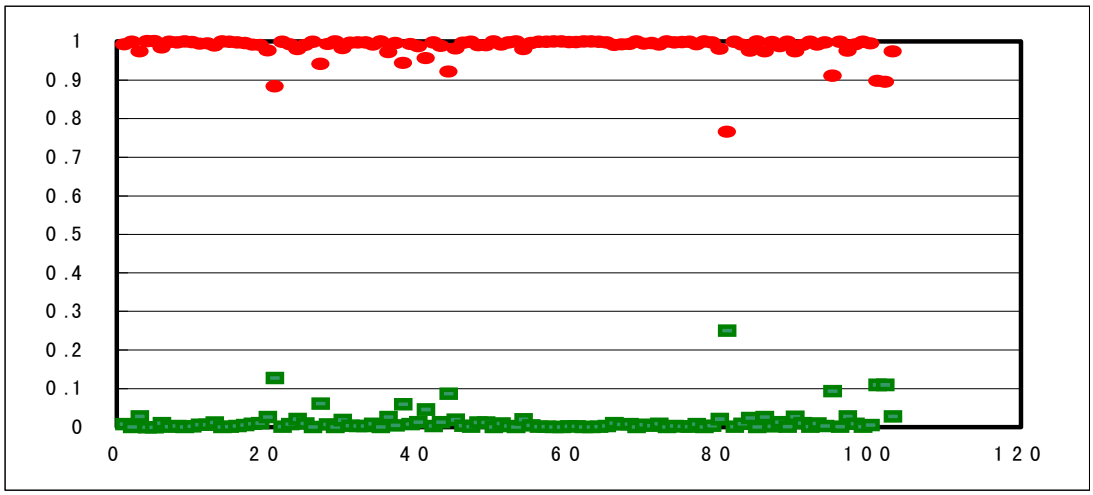


表 4．23　NN2 の追加学習（1）の認識分布（標準米）



＜4．4＞実システムライン画像での実験

これまでの＜4．2＞、＜4．3＞でのシミュレーションでは、＜4．1＞で説明したように撮影用筐体と研究室内のＵＳＢカメラを用いてお米を撮影し、NN シミュレーションを実施した。この節では、㈱セイレイ工業で実システム装置を使用し撮影した米の画像データを、現状の NN1、2 シミュレーションシステムで起動するか確認をする。

（4．4．1）実システム撮影装置の構成

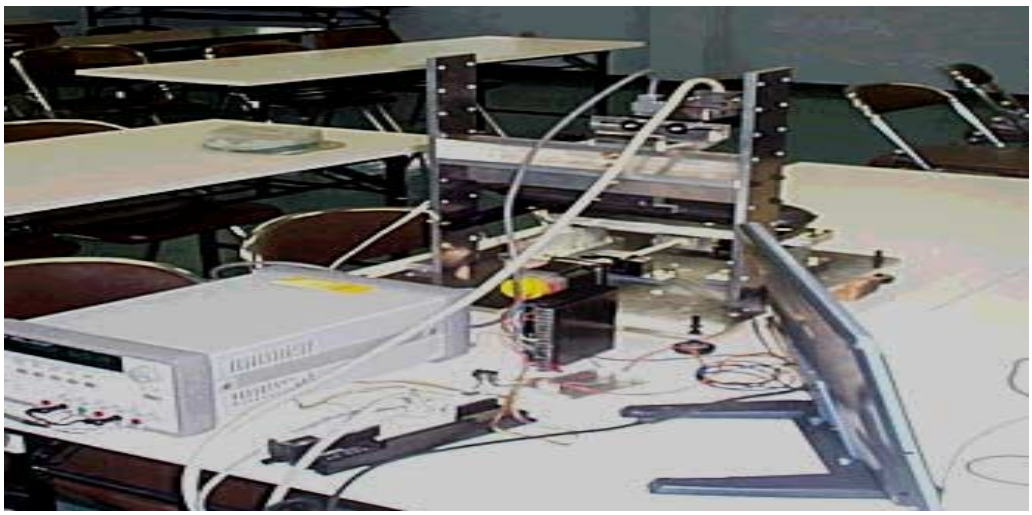


図 4．10　実システム撮影装置

これまでの米画像の撮影は研究室内の USB カメラを使用していた。この USB カメラを使用する撮影において、米を搭載するトレーを常に撮影装置内に固定し、USB カメラを上下に移動させる。

図 4．10 に示す実システム撮影装置による米画像撮影は、USB カメラを用いた撮影と異なる。USB カメラを用いた撮影法は、画像の上部から走査を行い画像の最下部まで走査を行うため、一度の撮影で画像を 2 次元に撮影可能である。一方、実システム装置においては、ラインセンサを使用する。ラインセンサを使用する場合、USB カメラを用いた場合と異なり画像の走査が一度のみである。したがって、米を搭載するトレー移動させなければ、正確な米画像を撮影することが不可能である。

この実システム撮影装置を使用し米画像を撮影することにより、USB カメラに撮影では実現することができない移動中の米を撮影することが可能になる。

（4．4．2）データ採取

データ採取は、図 4．10 で示す実システムを使用し米を撮影する。撮影後の画像データは、これまでの画像データと異なり撮影後にグレースケール、256 階調のＢＭＰファイルに保存される。

図 4．11 実システム撮影装置で撮影した米

（4．4．3）シミュレーション

次に、＜4．1．1＞と同様にシミュレーション用の画像データに変換する。その際、画像データのサイズがかなり小さいので、画像サイズを拡大変更する。

画像データを変更した後、（4．1．2）のように NN 学習シミュレーションを行う。実験条件は、表 4．23 に示す。

表 4．24　実験条件

米の種類	2
学習データ枚数	2
評価データの枚数	4
マスクの数	50
2 乗誤差	1.0E-4
学習方法	改良型BP法

＜問題点＞

図 4．11 の画像データや表 4．23 の実験条件を基に、シミュレーションシステムを起動すると、マスク処理の部分でバグが生じシステムが途中で停止する。

バグの理由は、図 4．12 で示すようにマスク処理においてエッジ検出部分のしきい値が高かったため、エッジ検出が正常に動作できなかったためである。

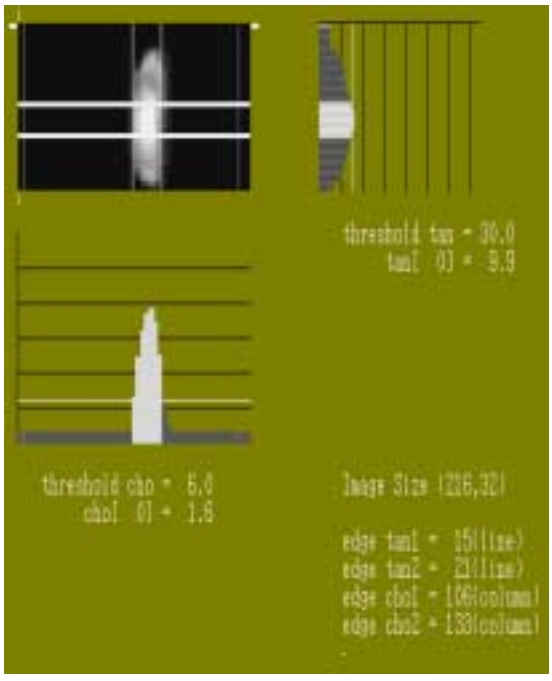


図 4．12　エッジ検出（1）

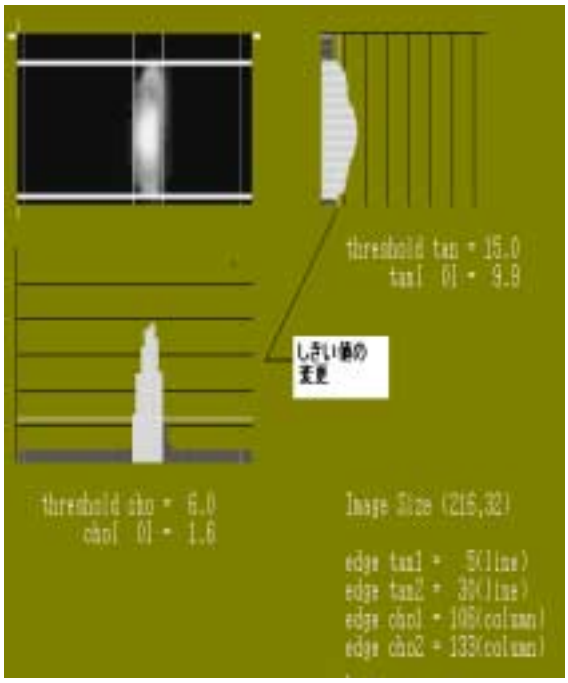


図 4．13　エッジ検出（2）

＜解決策＞

エッジ検出部分の問題点を解決するために、しきい値などの初期値を設定する DEFINE．H ファイル内のエッジ検出部分のしきい値変更を行なう。しきい値変更後のエッジ検出は図 4．13 に示す。

学習、選別、評価方法については、これまでのシミュレーションと同様である。また、選別手法は NN2 を使用する。

＜実験結果＞

（株）セイレイ工業から提供された画像データが全部で 8 枚と少ないため、正確な実験結果とは言いがたいが、実験結果は 4．24 に示す。

表 4．25　実験結果

米の種類	%	認識率
rice01		75%
rice02		50%

rice01, rice02 は、8 枚ある画像データを任意で分別したものである。

このシミュレーションは、実システム撮影装置で撮影する画像データで NN1、2 シミュレーションシステムが起動するかの確認が目的である。そのため、認識率については考慮しない。

＜4．5＞考察

この項では、これまでに実施した各々のシミュレーションを得ての考察を行なう。3．で提案した新しい米用選別手法であるニューロテンプレートマッチング選別手法を用いた米の色彩選別シミュレーション結果は、良好と判断できる。
この選別手法を使用することにより、従来の色彩選別機と異なりしきい値や取り扱う米の量に関係なく、常に高い認識率を維持することが可能である。

つぎに、NN1 と NN2 の比較を検討する。どちらの選別手法を使用する場合においても、90%以上を超える高い選別結果を示す。

NN1 を使用する場合、ある一定のしきい値を設定しなければならないので、若干しきい値により選別性能に影響を及ぼす可能性がある。一方、NN2 を使用する場合、NN1 を使用する場合と異なり、しきい値を設定する事がなくユニットの反応値の大小関係で明確に見極めるので、選別性能としては信頼を持てる。

5. まとめ

本研究では、現状の米用色彩選別機の問題点を解決する上で、非線型処理能力を有し紙幣識別、掌紋、指紋、顔画像、筆圧による個人認証などでその有用性が示されている NN による選別手法と従来の米選別手法であるテンプレートマッチング選別手法を融合させたニューロテンプレートマッチング選別手法による効率的な米の選別を提案した。また、本研究では、提案手法を用いての米の色彩選別シミュレーションを PC 上で実施し、米の色彩選別においてニューロテンプレートマッチング選別手法が有効であることを示した。

6. 今後の課題

本研究でのニューロテンプレート選別手法の有用性の検討において、実際の米用色彩選別機における実時間性は考慮してない。とくに、数十ミリ秒で 1 粒の米の選別処理を実施する必要がある場合、テンプレートと格納するメモリ空間の制約以上に、どれほどのテンプレートを処理できるかの問題がある。また、米を実システム装置で撮影し、米を一粒ずつの切り出しを行ない、米選別までの実時間についても問題が残される。これらについては、切り出しアルゴリズムとニューロテンプレートマッチング選別手法をＤＳＰを核とするニューロボードに移植し、実時間処理の検定が必要になる。

さらに、もう 1 つの問題点はマスク設定領域である。本研究で説明したマスク領域は 4．シミュレーションの　図 4．6 の通りであるが、実際の米、特に、被害米については被害部分が常に一定部分にあるのではなく、米の中央部分や先端部分などに点在している。このマスク領域では、若干の不都合が生じるために、マスク処理を一部変更しなければならない。

これらの問題点については、今後の課題として取り組んでいかなければならない。

7. 謝辞

一年間に渡りご指導いただいた竹田史章教授に心から御礼申し上げる。本研究において、米や撮影用筐体などのご提供していただいた（株）セイレイ工業高知工場第 2 開発部浜口正様はじめ開発グループの皆様に厚く御礼申し上げる。
最後に、ニューロテンプレートマッチング選別シミュレーションシステムなどのソフトウェア部分のご提供ならびに、ご指導していただいた（株）グローリー工業研究開発センター長吉田邦夫様はじめ、新事業開発部西蔭紀洋様に対しても心から厚く御礼申し上げる。

8. 参考文献

(1) 竹田史章・西蔭紀洋：「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（2001）」Ｖｏｌ．121－Ｃ、pp196～205

(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」、第 44 回システム制御情報学会論文誌 Vol.13、No.4　pp415～416　(2000)

(3) Takeda,F.,Nakahara,M.,Ichiryu,Y and Uchida,H：”Autonomic Neuro-Recognition Board for Paper Currency, SPAT2000”, ｐp85―90 (2000)

(4) 竹田史章・大松繁：「マスク方式によるニューロ紙幣選別機の開発、システム制御学会論文誌」、Vol.6,　pp.283―289（1993）

(5) 竹田史章・大松繁：「ランダムマスクを前処理機構に有する小規模ニューラルネットワークによる貨幣識別技術」、電気学会論文誌　Ｃ分冊、Vol．113 No.10、pp87―92（1994）

(6) Takeda, F. and Omatu, S.: “high speed paper currency recognition by neural networks”,IEEE Transaction oNNeural Networks, Vol.6,No.1, pp.73-77 (1995)

(7) Takeda, F. and Omatu, S.: “ Bank note recognition system using neural Network with random masks”, Proceeding of the World Congress oNNeural Networks, Portland, USA, Vol.1,pp.241- 244　(1993)

(8) Takeda, F. and Omatu, S.: “A neuro-money recognition using optimized Masks by GA “, Advance in Fuzzy Logic , Neural Networks and Genetic Algorithms LNAI 1011, pp190-201　(1995)　Springer

(9) Takeda, F. and Omatu, S.: “A neuro - system technology using bank note recognition“, Proceedings of the Japan/USA Symposium on Flexible Automation , Buston,USA, Vol.2, pp.1511- 1516　(1996)

(10) Takeda, F. and Omatu, S.:“A neuro-recognition technology using for paper currency using optimized masks by GA and its hardware, proceedings of the International Conference on Information Systems Analysis and Synthesis“, Orlando,USA,pp147-152　(1996)

(11) Takeda, F. and Omatu, S. and Nishikage, T: ”Neural network recognition System tuned by GA and design of its Hardware by DSP”, Proceedings of International Symposium on Artificial Intelligence in Real-time Control, Malaysia, pp356-362　(1997)

(12) 竹田史章・西蔭紀洋：「ニューロ識別における軸対象マスクの提案とＧＡによる最適化および最適結果の統計的解析」、システム制御情報学会論文誌 Vol.12、No.1、pp11―19 (1999)

(13) S.Ngata, M. Sekiguchi & K. Asakawa: “Mobile Robot Control by a Structured Hierarchical Neural Netwark” IEEE, Control System ,A p.69 (1990)

(14) Widrow,B., Winter,R.G. and Baxter,R.A: ”Layered Neural Nets for Pattern Recognition”,IEEE Transaction Acoustic,Speech & Signal processing, Vol.36, No.7, pp.1109-1118 (1988)

(15) 麻生秀樹:「ニュ―ラルネットワーク情報処理」pp10～18、pp50~54（1989）オーム社

(16) 舟久保登：「パターン認識」pp154~157 pp114~120 (1993) 共立出版

(17) 末松良一・山田宏尚「画像処理工学」pp120~140 (2000) コロナ社

(18) 安居院猛・長尾智晴「Ｃ言語による画像処理入門」 pp47―74 (2000) 昭晃堂

≪付録≫

・切りだしプログラム

```
/************************************
***
***   ファイル名          ： MKSLAB.c
***
***   概要                ： 米粒画像から 米粒を切出す。
***
***
***
***
************************************/
#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    200       // ２値化の
しきい値

#define             LABEL_START_NO
        200                 /* 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
        30                  /* 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 */

/************************************************/
/*
```

```
                                        */
/*              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 ;

          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);
//        }
          else if( argc != 3 )
          {
                        printf("¥n Usage: CUTOUT [in_file]
[num]");
                        printf("¥n      ( in_file   : 入力フ
ァイル                ) ");
                        printf("¥n      ( num       : 収
縮回数     max 100      ) ");
                        printf("¥n¥n   Push   any   key!");
getch(); exit(1);
          }
          else
          {
                        strcpy( fname_i , argv[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 */

          disp_information() ;    /* 画像情報表示 */

          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 ＞");

                        printf("¥x1B[20;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)
                        {
                                printf("¥x1B[21;30H                                ");
                                printf("¥x1B[21;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 ;
                                }
                        }

                        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 ;

                _setcolor(0x0a);
                for ( i=0 ; i<labelnum ; i++ )
                {
                                        /* 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
***" );

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

                /* 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" );

        _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);
}
```