培養幼植物体の形状情報の画像計測および生育状態の同定と将来予測に関する研究

課題番号　08660315

平成８年度～平成９年度科学研究費補助金（基盤研究（Ｃ）（２））

研究成果報告書

平成１１年３月

研究代表者　庄野　浩資
（岩手大学農学部助手）

はしがき

本研究の目的は次のようである。すなわち、バイオテクノロジーなどの作物の大量栽培技術の実用化によって、将来、植物性食料の原材料として大量生産されるであろう培養幼植物体、すなわち生育段階の最初期にある作物幼苗の生産性向上を図るために、画像情報などの非接触計測可能な情報を最大限に利用することにより、作物の生育情報を効率的かつ非破壊的に判定し獲得する手法を開発する。この計測手法の基礎となる考え方は、多数の方向から撮影した2次元画像から、葉傾斜角などの3次元的形状情報を計算し、さらに、その結果から、「しおれ状態」や「草姿」などの生育状態の診断結果を求めようとするものである。ここでの計測手法は従来はその複雑さのために効率的な計測が困難であった葉群の葉傾斜角を、効率的かつロバストすなわち繁茂状況などの影響を受けずに3次元計測を実現しようとする点に特徴がある。

本研究では、まず、実験材料として入手しやすいトマトあるいはコマツナ苗などの比較的大型の幼苗を計測対象としてその計測手法を確立した。さらに、計測の各手順を整理して処理の流れを最適化することにより計測全体の効率化を図った。さらに、対象物の形状に固有なパラメータを抽出して別途指定可能とすることにより、当初の計測目的である幼苗などスケールのミクロな作物に適用可能とした。さらに、作物の色情報にも着目し、走査型2次元分光分析計を試作した。最後に、作物の形状を栽培者である人間がどのように認識し、評価判断するかという問題に関して、アンケート調査に基づく初歩的な考察も試みた。このようにして得られた熟練栽培者の作物の生育状態の評価方法を、将来的に、例えば人工知能など知識工学的手法によって定式化し、すでにここで獲得可能となった形状情報と色情報を基礎情報として用いることにより、当初の研究目的である培養幼植物体の生育状態の同定と将来予測が可能になると考えられる。

ここで、研究代表者は平成9年度4月当初、「胸椎脊髄症」に罹病し、手術をともなう長期入院加療を余儀なくされたため、体力および研究実行期間上、無視できない制約を被ることとなった。このため、本研究では当初の目的のうち、前半の培養幼植物体の生育状態の計測手法の開発はほぼ予定通りに遂行できたが、主に平成9年度実施を予定していた後半の生育状態の同定と将来予測に関してはこれらの制約のため必ずしも成就したとは言い難い。これらの残された部分に関しては今後も積極的な研究が必要と考えられる。

最後に、第3章「走査型2次元分光分析計の開発」の研究実施を平成10年度卒業研究として担当していただいた大場恭子氏に、また、第4章「アンケート調査に基づく作物の形状認識および評価方法に関する解析」の研究実施を平成8年度卒業研究として担当していただいた石原拓朗氏の両氏に心からの感謝の意を表したい。

研究組織

研究代表者　：　庄野　浩資　（岩手大学農学部助手）

研究経費

平成８年度	１，９００千円
平成９年度	３００千円
計	２，２００千円

研究発表

（１）学会誌等

なし

（２）口頭発表

なし

（３）出版物

なし

研究成果

目次

1．葉傾斜角の画像計測に関する研究

1．葉傾斜角の画像計測に関する研究

葉傾斜角は、作物の水分状態などの内的な状態を端的に反映する重要な情報であり(楊・蔵田(1991)、大原ら(1992))、その定量的な測定値は、作物の生育状態を適宜把握するための指標など、栽培管理上極めて有用な情報となり得る。ここでは幼苗などの鉢植えされた作物個体の葉群を対象に、葉群全体の平均的葉傾斜角および葉傾斜角の3次元分布を、テクスチャ特徴に基づいて求める手法を提案し、その有効性を検討する。提案する手法の基本的な特長は以下の通りである。

・特長1：特殊な撮影装置を必要とせず、写真・ビデオなどの一般的可視画像を材料とする。
・特長2：基本的に非接触・非破壊計測手法であり、圃場での測定が容易である。
・特長3：個々の葉を抽出・認識せずに葉群全体の平均傾斜角を一度に測定することが可能である。
・特長4：葉傾斜角の3次元分布の測定が可能である。

1．1．葉傾斜角の画像計測手法の提案

近距離から撮影した葉群の画像には、各葉の様々な3次元的形状情報が透視変換により2次元的な模様(テクスチャ)として投影されている。このような2次元的なテクスチャの特徴から対象物の3次元的形状情報の復元が可能であれば極めて有効な計測手法となり得る。実際、地面などの大きな曲面上のテクスチャの特徴から曲面自身の起伏を認識した研究例がある(Witkin(1981))。しかし、葉群のように、小さく複雑な物体の3次元的形状情報の復元を目的とした研究例はほとんどない。これは、近接撮影した画像中の葉群のテクスチャが極めて複雑多様であり(庄野ら(1994a))、しかも撮影装置の幾何的な配置などの付帯的条件の影響が大きいなど、その取り扱いが容易ではないことが大きな理由である。そこで本研究では、比較的単純な3次元的形状情報である葉群の傾斜角に焦点を絞り、そのテクスチャからの復元を試みる。

1．1．1．葉傾斜角の測定基本原理

1）パワースペクトラム法におけるくさび状特徴の特性

ここでは、パワースペクトラム法(例えば、森・坂倉(1990b))によるテクスチャ特徴を用いて葉傾斜角の計測を行う。パワースペクトラム法の原理は、画像に2次元フーリエ変換を施して得られる2次元パワースペクトラム(以下、単にパワースペクトラムと呼称)の分布状況から、画像に含まれる物体像の2次元的形状情報を求めようとするものである。この際重要なのは、物体像が画像中のどの位置に存在しても、その形状が同じであればパワースペクトラム分布形状は変わらない点、さらには複数の物体像を含む画像からは、個々の物体像のパワースペクトラム分布を重ね合わせた分布が得られる点である。

本手法にはリング状テクスチャ特徴とくさび状特徴があるが、特に本研究では、対象物を様々な角度から撮影した画像群から得られるくさび状テクスチャ特徴から、物体の3次元的形状情報である葉傾斜角の復元が可能である点に注目した。なお、本研究では特に断らない場合、葉面が垂直の場合を0度、水平の場合を90度とする。

2）小平面の傾斜角とくさび状特徴の関係

葉面のモデルとして最も単純な小平面(ここでは円盤)を考える。45度の傾斜角を持つ小平面を、水平面方向から多角度撮影したシミュレーション画像(以下、回転撮影画像と呼称、ここでは、24度間隔で、合計15枚)を、透視変換に基づき作成した。画像の作成方法は以下の通りである。小平面の半径は32ドットであり、仮想的にピンホールカメラを小平面中心から充分遠い1024ドットの距離に設置し、小平面群を垂直中心軸の周りに24度刻みで回転させながら撮影した。ここで、Fig.2.1-1は本研究における線分像の傾斜角とくさび状テクスチャ特徴(64および128段階)との対応関係である。

Fig.2.1-2は、作成画像のうち真横から正面方向までの8枚である。さらに本図に、各小平面の像から求めた64段階のくさび状テクスチャ特徴を示す。本図より、小平面に対して正面付近から撮影した像は、一定の広がりを持つ面となり直線性が低く、くさび状テクスチャ特徴に明確なピークが現れないのとは対照的に、真横付近から撮影した像はほぼ直線となり、くさび状テクスチャ特徴に傾斜角45度に対応した位置で明確なピークが発生することが判る。この際重要な点は、くさび状テクスチャ特徴が小平面の位置の影響を受けないため、小平面の位置を考慮することなく、くさび状テクスチャ特徴から傾斜角を計ることが可能である点である。

3）平均葉傾斜角の計測

小平面の撮影角度とくさび状テクスチャ特徴との対応関係を利用して、葉方位角がランダムな葉群の傾斜角をテクスチャの特徴から測定する方法について述べる。

まず、葉群の全体画像からその平均葉傾斜角を測定する場合を考える。葉方位角の分布がランダムであれ

ば、一枚の画像中に撮影方向に対し真横付近を向く葉面が複数存在し得る。これらの葉の像から得られるパワースペクトラム分布は、くさび状テクスチャ特徴にそれぞれの葉傾斜角の位置に対応した明確なピークを発生させるが、その他の葉の像はくさび状テクスチャ特徴に明確なピークを発生させない。したがってこの場合は、くさび状テクスチャ特徴に平均葉傾斜角を示すピークが現れると考えられる。

一方、この条件が充分満たされない場合は、葉群を複数角度から撮影することにより、葉が撮影方向に対し真横付近を向く機会を人為的に増やす作業が必要となる。さらに、各回転撮影画像から得たくさび状テクスチャ特徴において、例えば各成分毎の最大値をとることにより、平均葉傾斜角に対応するピークを得ることが可能と考えられる。

4）葉傾斜角の3次元分布の計測

次に、葉傾斜角の3次元分布を測定する場合を考える。

まず、各回転撮影画像を適当な大きさの小区画に分割し、各小区画のくさび状テクスチャ特徴から葉傾斜角を求める。この結果、各回転撮影画像毎に葉傾斜角の2次元分布が得られる。この段階では、各小区画の葉傾斜角は、それぞれ単一の角度からの撮影画像から得られたものであり、葉方位角の分布状況によっては必ずしも正しい葉傾斜角ではない場合がある。このため最終的には、各部位の葉傾斜角を全撮影角度において総合し、正しい葉傾斜角の3次元分布を再構成する処理が必要となる。本研究では、3次元画像再構成法の基本的原理である逆投影法を以下のように応用し、葉傾斜角の再構成を試みた。Fig.2.1-3に、逆投影法の概要を示す。

具体的には、計算機内に、それぞれの要素を葉群の各部位に対応させた3次元配列を準備する。さらに、葉傾斜角の各2次元分布を撮影角度に対応した角度でこの3次元配列に正射影的に投影する。投影光の明るさは葉傾斜角と同じ値であり、投影光が通過した配列要素にはこの葉傾斜角が加算される。以上の操作を全撮影角度で行う。すなわち、本処理は、各回転撮影画像から得られた葉傾斜角を各部位毎に平均することと等価である。

本研究における再構成法は、可視光による撮影画像から求めた葉傾斜角を基本データとしている。このため逆投影法においても、外から見えない部分の葉傾斜角の再構成は難しい。しかし、隠蔽が著しくない葉群、あるいは葉密度が高い場合においても表面付近の葉群への適用は可能と思われる。また本研究では、透視変換的に撮影された画像から求めた葉傾斜角を正射影的に逆投影することにより生じる幾何的な歪みの補正や、再構成像を先鋭化するためのフィルタリングは行わず、今後の課題としている。

1．2．葉傾斜角の画像計測手法の有効性の検討

1．2．1．シミュレーション画像を対象とした計測

あらかじめ傾斜角の分布状況が判明しているシミュレーション画像に本手法を適用し、まず小平面群の平均傾斜角の計測を、さらに小平面群の傾斜角の3次元分布の再構成をそれぞれ試みることで、その基本的な有効性を検討する。

1）材料および方法

(a)小平面群の平均傾斜角の計測

平均傾斜角がそれぞれ20度、40度、60度の複数小平面(円盤)からなる3種の小平面群をそれぞれ45度間隔で水平方向から回転撮影したシミュレーション画像(それぞれ8枚)を透視変換(例えば、出口(1991))に基づき作成した。画像の作成方法は以下の通りである。

縦・横・高さそれぞれ256ドットの立方体内で一様分布する小平面群を用意した。小平面の半径は16ドットであり、各小平面群の小平面数は125個である。各小平面の傾斜角の分布は、正規分布(平均値は平均傾斜角、標準偏差は10度)となるように、また方位角の分布は、0度から360度までの一様分布となるようにした。仮想的にピンホールカメラを立方体中心から充分遠い2048ドットの位置に設置し、小平面群を垂直中心軸の周りに45度刻みで回転させながら撮影した。最終的に、撮影した画像を256×256ドットにスケーリングした。

具体的な計算手順は以下の通りである。

各画像(256×256ドット)に対し2次元FFT演算を施し、得られたパワースペクトラム分布から128段階のくさび状テクスチャ特徴を求めた。

(b)小平面群の傾斜角の3次元分布の再構成例1

平均傾斜角が0度、20度、40度、60度および90度とそれぞれ異なる5種類の小平面(円盤)群を水平方向に並列して作成した複合群落を、水平方向から45度間隔で回転撮影すると想定したシミュレーション画像(8枚)を透視変換に基づき作成した。Fig.2.2-1に正面画像を示す。

画像の作成方法は以下の通りである。

縦・横・高さそれぞれ256ドットの立方体の中で一様分布する小平面群を5種類用意した。Fig.2.2-2に、傾斜角の水平分布図を示す。小平面の半径は16ドットであり、各小平面群の小平面数はそれぞれ81個である。それぞれの小平面群の方位角の分布は、それぞれ0度から360度の一様分布となるようにした。さらに、各立方体を中心間の間隔が256ドットとなるように水平方向に並列した。次に、仮想的にピンホールカメラを立方体全体の重心から充分遠い2048ドットの距離に設置し小平面群を垂直中心軸の周りに45度刻みで回転させながら撮影した。最終的に、撮影した画像を896×896ドットにスケーリングした。

具体的な計算手順は以下の通りである。

まず、各画像(896×896ドット)に対し、左上隅から右下隅まで16ドット刻みで縦横それぞれ位置をずらして128×128ドットの小区画を抽出した。小区画は各画像において、合計49×49(=2401)区画得られた。さらに、各小区画において、2次元FFT演算により得られたパワースペクトラム分布から64段階のくさび状テクスチャ特徴を求めた。次に、各小区画において、くさび状テクスチャ特徴から傾斜角を求めた。ここで、各小区画の傾斜角は、くさび状テクスチャ特徴の重心値とした。最後に、前述の逆投影法により、傾斜角の3次元分布を再構成した。

(c)小平面群の傾斜角の3次元分布の再構成例2

平均傾斜角が20度、40度および60度とそれぞれ異なる3種類の小平面(円盤)群を同軸の3層をなすように作成し、さらに、この複合群落を、水平方向から45度間隔で回転撮影すると想定したシミュレーション画像(8枚)を透視変換に基づき作成した。Fig.2.2-3に正面画像を示す。

小平面群の分布範囲は縦・横・高さそれぞれ256ドットの立方体の中に収まるようにした。Fig.2.2-4に、傾斜角の水平分布図を示す。また、小平面の半径は8ドットであり、各小平面群の小平面数は20度が79個、40度が226個、60度が424個である。それぞれの小平面群の方位角の分布は、それぞれ0度から360度の一様分布となるようにした。

仮想的にピンホールカメラを立方体の重心から充分遠い2048ドットの距離に設置し小平面群を垂直中心軸の周りに45度刻みで回転させながら撮影した。最終的に、撮影した画像を896×896ドットにスケーリングした。具体的な計算手順は、小平面群の傾斜角の3次元分布の再構成例1と同様である。

2）結果および考察

(a)小平面群の平均傾斜角の計測

Fig.2.2-5は、シミュレーション画像群から得たくさび状テクスチャ特徴である。本図では、それぞれの小平面群において、平均傾斜角に対応する位置(±20度、±40度、±60度)にくさび状テクスチャ特徴のピークが生じていることが確認できる。ピークが0度に関して対称な位置に2つ同時に生じたのは、小平面はその方位角によって、正の傾斜角を持つ像を投影する場合と負の傾斜角を持つ像を投影する場合があるためである。したがって、本手法により小平面群の平均傾斜角を計ることが可能と考えられる。

実際の葉群においては、湾曲など葉面の変形が著しい場合あるいは葉傾斜角のばらつきが大きい場合には、くさび状テクスチャ特徴のピーク形状が不明瞭となることが考えられる。しかし、このような場合は、光学的倍率を上げて撮影部位をクローズアップし、測定範囲を限定することで、ピーク形状が明確となると考えられる。

(b)小平面群の傾斜角の3次元分布の再構成例1

Fig.2.2-6に、Fig.2.2-1に示した画像群から得られた傾斜角からその3次元分布を再構成した結果を、水平断面による2次元分布で示す。本図に明らかなように、再構成像は5種類の小平面群の平均傾斜角の配置状況を反映している。

(c)小平面群の傾斜角の3次元分布の再構成例2

Fig.2.2-7に、Fig.2.2-3に示した画像群から得られた傾斜角からその3次元分布を再構成した結果を、水平断面による2次元分布で示す。本図に明らかなように、再構成像は3層の小平面群の平均傾斜角の配置状況を反映している。

以上のシミュレーション計測の結果より、本手法を用いて、小規模な小平面群の平均傾斜角の測定、および傾斜角の3次元分布の再構成を行うことは、充分可能であると思われる。

1．2．2．苗個体画像を対象とした平均葉傾斜角の計測

遮光処理および過剰潅水処理を施して形状差を人為的に形成した作物群を準備し、それぞれから回転撮影画像を作成、さらにこれらの画像に本手法を適用して作物の形状差の計測を試みることで、本手法の有効性

を検討する。

1）材料および方法

ここでの供試画像は全て以下のようにして作成した。

ASA100のカラースライドフィルムを用いて各対象を水平方向から撮影した。この際、対象から充分離れ、レンズによる撮影画像の幾何的歪みを避けた。さらに、各フィルムをフィルムスキャナー(クールスキャン、ニコン(株)、東京)にて、画像処理コンピュータ(Macintosh 8100/80AV、アップルコンピュータ(株)、東京)に800DPI(Dot Per Inch)の密度でRGBカラー画像データとして取り込んだ。さらに、画像データから作物個体を含む部分を256×256ドット、あるいは896×896ドットの大きさに切り出した。RとGの濃度差が背景と作物部分とで異なることを利用して、画像データから背景のみを消去し、最終的に、画像データを256階調のグレースケール画像とした。

(a)遮光処理によるコマツナ苗の形状差の解析

本処理の概要は以下の通りである。

8月下旬に、コマツナ(品種：ミズキ)をビニールポット20鉢に播種し、ビニールハウス内にて育苗を行った。ほぼ全個体が発芽した3日後に、コントロール区10個体と処理区10個体とに2等分し、処理区を2重の寒冷紗にて遮光した。両区とも、他の栽培管理は等しくした。播種後16日目に、各区における平均的な個体をそれぞれ3個体選別しそれぞれ回転撮影した。

Fig.2.2-8に画像例を示す。撮影角度は、45度毎に8方向である。同図には、参考のために各撮影画像を多重露出した画像を示している。なお、ここでは、画像中の葉柄の影響を排除するために、撮影画像からマニュアル的に葉柄部分を削除している。

コントロール区苗は、処理区苗と比較すると各葉の面積が大きく、また湾曲化が著しい。コントロール区苗の主葉の平均傾斜角は、葉基部と先端部分を結ぶ直線の傾きではほぼ60～70度である。一方、処理区苗は、いわゆる「徒長」傾向にあり、各葉の面積は小さく、平面性が高い。また、処理区苗の主葉の平均傾斜角はほぼ90度の水平方向である。

具体的な計算手順は以下の通りである。

まず、各回転撮影画像(256×256ドット)に対し2次元FFT演算を施し、得られたパワースペクトラム分布から128段階のくさび状テクスチャ特徴を求めた。さらに、各回転撮影画像から得たくさび状テクスチャ特徴群において各成分毎の最大値を求めた。最後に、各区において、くさび状テクスチャ特徴を成分毎に平均した。

(b)過剰潅水処理によるトマト苗の形状差の解析

本処理の概要は以下の通りである。

7月中旬に、トマト(品種：フジミ)をビニールポット20鉢に播種し、ガラスハウス内にて育苗した。約1ヶ月後にワグネルポットに、コントロール区10個体と処理区10個体とに2等分して定植し、さらに処理区を常に水を浸した容器の中に設置した。この際、両区とも、他の栽培管理は等しくした。さらに、定植後3日目に、各区における平均的な個体をそれぞれ3個体選別し苗画像を撮影した。

Fig.2.2-9に画像例を示す。撮影角度は、90度毎に4方向である。同図には、参考のために各撮影画像を多重露出した画像を示している。なお、ここでは、画像中の葉柄の影響を排除するために、撮影画像からマニュアル的に葉柄部分を削除している。コントロール区においては、生長点付近の各葉はゆるやかな弓なり状態で、各小葉はほぼ水平方向にのびており、その角度のばらつきは少ない。一方、処理区の生長点付近の各葉は、コントロール苗と比較すると傾斜による左右非対称性が著しく、また各小葉も上方に向かうなど角度のばらつきが大きい。

具体的な計算手順はそれぞれ以下の通りである。まず、各画像(256×256ドット)に対し2次元FFT演算を施し、得られたパワースペクトラム分布から128段階のくさび状テクスチャ特徴を求めた。さらに、各回転撮影画像から得たくさび状テクスチャ特徴群において各成分毎の最大値を求めた。最後に、各区においてくさび状テクスチャ特徴を成分毎に平均した。

2）結果および考察

(a)遮光処理によるコマツナ苗の形状差の解析

Fig.2.2-10に、各区において成分毎に平均したくさび状テクスチャ特徴を示す。コントロール区のくさび状テクスチャ特徴のピークは(±)60度から(±)90度付近に現れたが、そのピーク形状はなだらかであった。この位置は、主葉の傾斜角とほぼ一致する。一方、処理区のくさび状テクスチャ特徴には、実際の葉傾斜角に対応する(±)80度から(±)90度付近に比較的急なスロープが現れたが、これはコントロール区には見られ

なかった。以上より、くさび状テクスチャ特徴から推定される葉傾斜角と実際の角度は良く一致する。さらに、コントロール区にのみ現れた水平方向付近のなだらかなピークは、主葉の著しい湾曲を反映したものと考えられ、くさび状テクスチャ特徴は処理による葉群の形状差を良く示したと言える。

(b)過剰潅水処理によるトマト苗の形状差の解析

Fig.2.2-11(a)は、各区において成分毎に平均したくさび状テクスチャ特徴である。本図に明らかなように、多重露出画像に見られた両区の形状の相違はくさび状テクスチャ特徴にほとんど現れなかった。Fig.2.2-9における多重露出画像に見られるように、各区の形状の主な相違点が小葉の傾斜角のばらつきと本葉全体の緩やかな傾斜による左右非対称性にあることに留意すると、この結果の原因としては以下の2つが考えられる。

・測定解像度の限界

小葉の傾斜角のばらつきは、くさび状テクスチャ特徴の細かなピークとして現れることが期待される。このような細かなピークは、各成分の最大値をとる段階で他の大きな構造物由来のくさび状テクスチャ特徴の中に埋没したと考えられる。この対策としては、細かな構造物の葉傾斜角を測定するために、光学的倍率を上げて画像の撮影範囲をせばめ、より大きな構造物の影響を避けるなど、測定の解像度を向上させる方策が必要と考えられる。

・位置情報の欠落

本例では、処理区苗の本葉は、ほぼ一直線に伸びた2本の本葉が、ゆるやかに傾斜している。この苗の回転撮影画像を総合した多重露出画像に見られるように、本葉の傾斜角の分布は結果的にコントロール区と大差がない。このため両区の違いが検知されなかったと考えられる。この問題は、くさび状テクスチャ特徴が画像中の各葉の位置を反映しないために起こる。この対策としては、前述の方法により、葉傾斜角の3次元分布を測定することで形状差が明確になると考えられる。あるいは簡易な対策として、各回転撮影画像を上部、下部、あるいは左半分、右半分などと分割した画像を解析対象とすることにより、形状の違いを明確化することも可能と考えられる。

そこで、処理区苗の各画像を主茎を中心に左右2つに分割し、左側の画像のみを用いて同じ解析を施した結果をFig.2.2-11(b)に示す。本図において、両区の相違点が、左半分(0度～－90度)に明確に現れた。これは、コントロール区苗に比べて、処理区苗の方に上方に向かう葉群が多く含まれていることを示し、これは実際の形状差と良く一致する。

以上の結果より、本手法を用いて、小規模な葉群の平均葉傾斜角の測定、および葉傾斜角の3次元分布の再構成を行うことが充分可能であると思われる。このため、本手法は栽培管理のための生育指標として個々の作物の葉傾斜角の即応値を提供し得ると考えられる。

１．２．３．作物個体画像を対象とした葉傾斜角の3次元計測

ここでは、本手法を複数のトマト個体に適用することにより、その有効性すなわち、トマト個体間の形状的特徴の違いを適切に説明し得る情報を実際に計測可能か否かについて実証的に検討する。特にここでは、作物外観の概略的な形状(以下、概略的形状と呼称)を新たに計測項目に追加し、その葉傾斜角との同時計測を試みる。ここで、概略的形状とは、例えば作物体を遠方から眺めることにより個々の構成物の細かな形状が省略されて観察される作物体表層の形状を意味する。これは作物体に仮想的な布を密着被覆することにより形成される3次元曲面の形状と考えることができる。本情報は、葉面積などの生体量と、必ずしも1対1の関係にはないものの関連性は高い。既述のように、葉傾斜角は作物の水分状態などの比較的「短期的」な生育状況との関連性が高い(楊・蔵田(1991)、大原ら(1992))のに対し、概略的形状は、例えばトマトにおいて、各部位間の生長のバランスなど、作物が現在までに経験した「中・長期的」な生育環境の影響との関連性の高い有用な情報と考えられる。さらにここでは、計測により得られた形状情報を事後の計算処理が容易となるように独自の方法(後述)で簡略化する。さらに主成分分析により、作物の形状の違いを第三者が容易に理解可能な指標に統合化する試みを行う。なお、ここでの葉傾斜角は、水平を0度、垂直を90度のスケールに対応させているので注意する。

１）材料および方法

(a)供試材料

・トマト個体群の栽培と写真撮影

育苗用培土を詰めたワグネルポット(断面積1/2000a、高さ30cm)にトマト苗(品種：強力米寿)15個体を定植し、ビニルハウス内で栽培した。ここで、各ポット間の距離および潅水量を不均質とすることで光および水

環境のむらが生じるように調整し、各個体間に形状のばらつきを故意に発生させた。定植から約一ヶ月後、本葉展開数が平均約16枚となった時点で、9個体を無作為に選択した。

さらに、各個体をそれぞれ8方向(45度等角度刻み)からASA100のリバーサルフィルムを使って写真撮影した。この際、個体からは充分離れて撮影し、レンズによる像の歪みを極力さけるとともに、照明を可能な限り間接光とすることで個体表面上に顕著な影が生じることを避けた。また撮影の際には、背景として赤色紙のカーテンを設置した。これは、後の前処理過程で、葉群と背景とを相互の色の違いを利用して分離するためである。

・トマト個体群の形状的特徴

ここで、概略的形状に重点をおいた目視による観察から、各個体の形状的な特徴について述べる。

No.2、No.4、No.5、No.7の4個体は生長点付近に比べて下部の本葉の繁茂が進み、概略的形状は三角形型を呈していた。これらの葉傾斜角は、生長点付近および下部の小葉の下方への垂れが特徴的に見られた。また、No.1、No.6、No.8の3個体は、下部の本葉の生長が進まず、前4者とは対照的に概略的形状は逆三角形型を呈していた。これらの葉傾斜角は、生長点付近では小葉の下方への垂れが見られる一方、下部では前4者と比較すると顕著ではなかった。残るNo.3、No.9はこれら2グループの中間的形状を呈していた。

・前処理による供試材料(画像)の作成

各カラー写真をフィルムスキャナ(CoolScan、ニコン(株)、東京)を用いて、800DPI(Dot Per Inch)の密度で画像処理用コンピュータ(Macintosh　　8100/100AV、アップルコンピュータ(株)、東京)にデジタル画像データ(RGBカラー)として取り込んだ。各画像データ(RGBカラー)において、背景と葉群のRおよびGチャンネルの差が大きく異なることを利用して後者を抽出し、前者を濃度値0で置き換えた。さらに最終的に、RGBカラー画像を896×896ドットのサイズのグレースケール画像とした。Fig.2.2-12に各個体の正面画像を示す。

(b)方法

・葉傾斜角の3次元分布の測定原理

既述のように、葉傾斜角の3次元分布の測定原理は、前節にて考案した本手法を用いた。本手法は、小規模な葉群の、複数方向から撮影されたグレースケール可視画像(回転撮影画像)から葉傾斜角の3次元分布を簡便に再現可能な点に特長がある。

・概略的形状の測定方法

本手法で用いられる回転撮影画像は、適当な閾値を用いることにより、白黒2階調画像(シルエット)に変換される。葉傾斜角の3次元分布を再構成する際、同時にこれらの白黒2階調画像に逆投影法を適用すると、影像を外側から削り出すようにして概略的形状が3次元的に再構成される。

・本手法の実行方法

各画像において、128×128ドットの小区画を左上方から右下方まで16ドット毎に位置をずらしつつ抽出し、本手法の手順により、葉傾斜角の2次元分布を求めた。さらに、本手法の手順により、各個体の葉傾斜角の3次元分布を再構成した。この結果得られた3次元分布は(49×49×49)の大きさのマトリックスとなった。さらに、概略的形状の測定結果も葉傾斜角に合わせて(49×49×49)の大きさのマトリックスとした。

・形状情報の簡略化

今回計測対象とした形状情報はすべて3次元情報であるため、これらを用いて作物個体間の形状の違いを定量的に評価する場合、その情報量の簡略化が重要となる。なぜならば、形状情報が(n×n×n)の3次元マトリックス内部に分布する場合、例えば個体間の形状差をユークリッド距離として定量化するには、単純には(n×n×n)要素のベクトル間の距離を求める必要がある。これでは、nがある程度大きくなると計算処理負荷が膨大となるため、効率的ではない。

しかし実際には、作物の形状の違いを表すために必要充分な次元数は(n×n×n)よりは小さい。例えばトマト個体を対象とする場合、葉傾斜角の3次元分布において、中心軸の回りの回転の自由度の成分は軸対称性を仮定すれば分布の特徴を表すためには必ずしも必要ではない。さらには、葉傾斜角の3次元分布には概略的形状に関する情報も重複しており、これらの中から不必要な情報を除去すれば次元数は充分簡略化されると考えられる。そこで本研究では、以下に示す方法で形状情報の簡略化を試みた。

まず、概略的形状に関しては、その3次元形状の水平断面を同面積の円に近似し、その近似円の半径の1次元プロファイル(49要素)を求めた。次に、49要素を8要素毎の平均により6要素に集約した。但し端数が生じるため、作物体の最下部に対応する6要素目は9要素による平均とした。また、葉傾斜角の3次元分布に関し

ては、その分布形状を円筒形状に正規化し、円筒を内側から3等分(以下、内側から順に内層、中層、外層と呼称)した。さらに各層において、葉傾斜角を同一水平面上で平均し、1次元プロファイル(3層×49要素)に変換した。最終的に、各層において49要素を6要素に概略的形状の場合と同様の方法で集約した。この作業の概略をFig.2.2-13に示す。以上の作業により、形状情報はプロファイルを成分とする4種類(葉傾斜角の内、中、外層および概略的形状)のベクトル(各6要素)となった。なお、プロファイル位置の表記は、作物体最上部に対応する要素をNo.1、最下部に対応する要素をNo.6とした。

ここで、円筒の分割数を3、プロファイルの要素数を6としたのは、元のマトリックスデータの大きさと、その後の統計的分析にかかる計算量とを勘案した結果である。

・形状情報の指標化

ここで、前述の方法で簡略化された形状情報であっても、これらを予備知識のない第三者が理解し、逆に作物の具体的な形状の特徴を想像することは難しいと思われる。形状情報の有用性を高めるためには、その可読性に留意する必要がある。そのためには、具体的な想像の容易な形状の特徴の指標を形状情報の統合により作成し、各作物の形状的特徴をそれらの得点として再表現するなどの作業が有効と考えられる。そこで今回は主成分分析を用い、形状情報の指標化を試みた。

主成分分析は、与えられた変数群を線形に結合し、互いに直行する新しい変数群(主成分)を合成する。このため、形状情報間に相関関係などの情報の重複が存在しても、本手法の適用により、トマトの形状の特徴を端的に表現する独立した指標を合成することができると期待できる。

2）結果および考察

(a)測定結果の概要

供試材料No.1とNo.2における葉傾斜角の3次元分布をFig.2.2-14に示す。概略的形状に関しては、葉傾斜角の3次元分布の表面形状とほぼ同一であることからここでは省略した。本図は、主軸を含む垂直平面で葉傾斜角の3次元分布を切断して疑似3次元形式で可視化したものである。また、同図には、矢印で指示した位置での水平断面が同時に示されている。本図には両者の形状的特徴の違いが視覚的に良く表現されている。ここで、同図における3次元分布の可視化には専用ソフトウェア(Spy Glass Dicer 2.0、Spyglass社、米国)を使用した。

(b)形状情報の変動係数

簡略化した各形状情報の変動係数(標準偏差を平均値で除した値で、変数のばらつきの程度を表す統計量)をFig.2.2-15に示す。変動係数は変数の持つ情報量の大きさを反映する指標と考えることができる。したがって今回の場合は、各形状情報において、変動係数の大きなプロファイル位置ほど、あるいは各プロファイル位置において、変動係数の大きな形状情報ほど、供試材料間の形状の違いを良く反映していると考えることができる。

・概略的形状

概略的形状の変動係数は、プロファイル位置1から4までの間では生長点に対応するプロファイル位置1において最も高く約20%であったが、その他の位置では約10%と低かった。これは、今回の供試材料においては、概略的形状の上部におけるばらつきが極めて小さかったことを示している。ここで、Fig.2.2-12の正面像では概略的形状の上部におけるばらつきは必ずしも視覚的には小さく感じられないが、これは、上部の本葉に対する撮影角度が個々に少しずつ異なるため概略的形状が見かけ上ばらついて見えるためであり、矛盾していない。

プロファイル位置5以下では変動係数は急激に増加し、最下部のプロファイル位置6で最大となった。これは、概略的形状のこの位置でのばらつきが各供試材料間で最も大きかったことを示している。これは、前述の如く今回の供試材料の下部葉の概略的形状が三角形型と逆三角形型の形状に大きく分けられる状況と合致している。

・葉傾斜角

葉傾斜角の変動係数は、プロファイル位置上部(1～3)と下部(4～6)との間で以下のような大きな傾向の違いがあった。生長点を含むプロファイル位置上部(1～3)では、外層において最も高い変動係数を示した。これは、今回の供試材料においては、表面付近の葉群ほど供試材料間の形状の違いを良く表現していたことを示している。一方、プロファイル位置下部(4～6)では、外層の変動係数は上部とほぼ同程度であったが、内層の変動係数はそれ以上の高さを示した。これはプロファイル位置上部とは逆の結果である。次に、この傾向の違いの原因について考察する。

今回の供試材料においては、Fig.2.2-12に見られるように、プロファイル位置4より下部の葉群が三角形型を呈するものと逆三角形型を呈するものとの2グループに分かれていた。三角形型のグループでは下部の葉群の外層付近において下方への垂れが見立つ一方、内層、中層ではその傾向は見られなかった。逆三角形型のグループでは下部の葉群の密度が低く、各層毎の葉傾斜角の傾向の差は明確ではなかった。この両者の違いにより、結果的に内層、中層の葉傾斜角の供試材料間でのばらつきが高くなったと理解できる。また逆にこの結果は、葉群の繁茂状況により各層において葉傾斜角の傾向が異なる場合があるため、正しい計測結果を得るためには各層毎の葉傾斜角を別々に計測する必要のあることを示唆している。

基本的に1枚の作物画像に基づく従来の2次元的な画像計測手法では、密集した葉群の葉傾斜角を各層毎に分離して計測することは困難であり、ここに本手法のような3次元的計測手法の有用性が認められる。

(c)形状情報間の相関分析

簡略化した各形状情報の各プロファイル間の相関分析の結果をTable2.2-1に示す。但し、ここでは、結果を見通しやすくするために、葉傾斜角には各プロファイル位置毎の3層の平均値を用いた。また、以下の統計的処理には全て専用ソフトウェア(SAS JMP 2.0、SASインスティチュートジャパン(株)、東京)を使用した。

各プロファイル位置における葉傾斜角と概略的形状間の相関は、プロファイル位置1～4の、上位から中位葉においては総じて低かった。これは、これらの位置では本研究における計測法によって葉傾斜角と概略的形状がそれぞれ独立して計測されたことを示している。一方、プロファイル位置5～6の下部葉では若干高い相関を示した。これらの相関は、下部葉では葉群の存在する容積が増大し、概略的形状が増大するにつれ葉傾斜角が水平になる傾向が見られたことを示しており、Fig.2.2-12に見られるように実際の形状的特徴と一致している。

また、プロファイル位置4の概略的形状と、プロファイル位置6の葉傾斜角の間にかなり高い相関が見られた。これは、この位置の葉群の存在する容積が低下し、下部葉の概略的形状が三角形型に近づくにつれ、最下部葉の傾斜角が水平になる傾向を示しており上述の結果と同様であった。

以上の結果から、今回の供試材料においては、比較的若い部位である上部から中部葉において葉傾斜角と概略的形状をほぼ独立に計測することができたと考えられる。

(d)形状情報の指標化

全形状情報を一つのベクトル(24要素)にまとめて、主成分分析を適用した結果、累積寄与率は第5主成分において約90%となった。そこで、第5主成分までの分析結果をTable2.2-2に示す。

さらに以下において各主成分が表す形状的特徴の意味の解析およびその妥当性の検討を行う。ここで一般に、主成分は寄与率の低い下位になるほどその意味の解析が困難となること、さらには、形状的な特徴をグラフ化などにより可視化する場合、実用的な次元数はせいぜい3次元までと考えられることから、ここでの解析の対象は第3主成分までとした。第3主成分までの主成分ベクトルを形状情報毎にグラフ化した結果をFig.2.2-16に示す。

・第1主成分

本主成分の得点は、供試材料No.9で最大となり、逆にNo.5で最小となった。

概略的形状の主成分ベクトル要素は、プロファイル位置2、5、6で負符号となっており、この位置の概略的形状が小さい個体ほどその得点が高くなる傾向を示している。実際、Fig.2.2-12において供試材料No.9とNo.5の形状を比較すると、生長点を含む主茎が突出して上に伸び、下部葉が逆三角形状の「とっくり型」であるほど得点が高いことがわかる。

葉傾斜角の主成分ベクトル要素は、プロファイル位置3より下部において内層、中層が正符号であるのに対し、外層が負符号あるいはほぼ0である点に特徴がある。これはこの位置において、本主成分のうち、中層の葉傾斜角が垂直で、さらに外層の葉傾斜角が水平であるほど高い得点となる傾向を示している。

以上の結果を総合し、Fig.2.2-16の主成分ベクトルのグラフを、その意味する形状の傾向を理解しやすいようにFig.2.2-17(a)にイラスト化した。作図にあたっては、プロファイル位置が実際の位置に対応するように90度倒置し、作物体を模した。また、図中の細線は概略的形状の主成分ベクトルを表し、矢印は葉傾斜角の傾向を示している。葉傾斜角の傾向は便宜的に水平方向(主成分ベクトル要素が負符号)と垂直方向(主成分ベクトル要素が正符号)の2種類とした。また、葉傾斜角の主成分ベクトル要素の絶対値が小さく当該主成分への影響が少ない場合は矢印を省略した。なお主成分ベクトルは全供試材料における平均値および標準偏差で正規化されたデータから求められている。このため本図は、全供試材料の平均的形状からの乖離の傾向を誇張した表現となっている。したがって、実際の形状は、全供試材料の平均的形状を示すFig.2.2-17(d)を併せて参照する必要がある。

・第2主成分

本主成分の得点は、供試材料No.6で最大となり、逆にNo.9で最小となった。

概略的形状の主成分ベクトル要素は、プロファイル位置4で最大となっており、この位置の概略的形状が大きい「腰部肥大型」の個体ほどその得点が高くなる傾向を示している。また、プロファイル位置5、6で負符号となっており、この位置の概略的形状が小さい個体ほどその得点が高くなる傾向を示している。プロファイル位置1、2、3の主成分ベクトル要素の絶対値は小さく、上部および中部葉の概略的形状の得点への影響は小さいことを示している。

実際、Fig.2.2-12において供試材料No.6とNo.9の形状を比較すると、一見、概略的形状は類似しているが、詳細に観察すると、供試材料No.9の最下部における概略的形状が相対的に大きく、得点を下げる結果となったことが判る。

葉傾斜角の主成分ベクトル要素は、3層とも上部および下部葉で正符号となる一方、中部葉で負符号となっている。これは、葉傾斜角が上部および下部葉で垂直で、中部葉で水平であるほど高い得点となる傾向を示している。

以上の結果を総合し、本主成分の意味する形状の傾向を第1主成分と同様に図示するとFig.2.2-17(b)の様になる。

・第3主成分

本主成分の得点は、供試材料No.3で最大となり、逆にNo.5で最小となった。

概略的形状の主成分ベクトル要素は、上部および中部葉で正符号であり、プロファイル位置1で最大となった。逆に最下部では負符号となった。これは、上部および中部葉の概略的形状が大きい「頭部肥大」型の個体の得点が高くなる傾向を示している。

実際、Fig.2.2-12において供試材料No.3とNo.5の形状を比較すると、中部葉の概略的形状は類似しているが、供試材料No.3の生長点に対応する最上部の概略的形状が際だって大きく、得点を上げる結果となったと考えられる。

葉傾斜角は、各層ともプロファイル位置1において負符号となった。これは第1、第2主成分には見られない特徴である。これは、生長点付近および上部の葉群の傾斜角が水平になるほど得点が高くなる傾向を示している。またプロファイル位置3においても負符号となった。それ以外の位置では、おおむね正符号となった。

以上の結果を総合し、本主成分の意味する形状の傾向を第1主成分、第2主成分と同様に図示するとFig.2.2-17(c)の様になる。

ここで、指標化の結果を2次元散布図としてFig.2.2-18に示す。本図では、概略的形状に重点を置いた目視による観察から分類した前述の2グループを参考のために点線で囲った。結果として各グループは各主成分によって明確に分割されており、主成分分析による指標化が適切に機能したことを示している。ここで、各グループ内でのばらつきが大きいのは目視による分類で重点が置かれなかった葉傾斜角のばらつきのためと考えられる。本図と、各軸の意味を端的に表現したFig.2.2-17とを合わせて見ると、形状情報に関する供試材料の全体的傾向と各供試材料間の位置関係が視覚的に良く理解できる。

ここで、主成分分析による形状的特徴の解析は、主成分の内容が供試材料の形状的特徴の傾向により変わるため、分析の都度主成分の意味を読みとらねばならない面倒さがある。しかし、あらかじめ類型化された形状あるいは栽培目標となる形状が判明している場合は、例えばそれらの形状情報を教師データとし、各供試材料の形状情報と教師データとの距離を形状的特徴を表す指標とするなどの簡便な解析が可能である。この場合、指標の意味はあらかじめ明確であるため、第3者への分析結果の説明もより簡単となるなどのメリットが期待できる。

実際の圃場では、個体数および葉数が増加するにつれて、各個体の形状情報、特に葉傾斜角の3次元分布などの形状的特徴を目視や角度計などを使って手動的に認識し、形状情報に関する全体的傾向および各個体の特徴を判定することは極めて困難となる。現状では、せいぜい概略的形状の傾向を大まかに認識する程度であり、形状情報が充分有効利用されているとは言いがたい。本研究における一連の計測および解析作業は、作物の形状的特徴の傾向やばらつきなどを明確化かつ可視化することが充分可能であり、したがって栽培者に生育状況を判断するための重要な情報を提供する上で有望かつ有効な手段となり得ると考えられる。また本研究では、葉群の生育状況によっては、正しい葉傾斜角を計測するために、葉群を各層毎に3次元的に計測する必要のあることが示された。これは、本手法のような3次元的計測手法の有用性を改めて示唆している。今後は、回転撮影画像から得たくさび状テクスチャ特徴から最終的に葉傾斜角を決定する計算方法など、アルゴリズム自体のさらなる改良の努力とともに、様々な群落に対して応用事例を積み重ねることにより、

適用可能範囲の拡大など実用性を高めるための継続的研究が必要と考えられる。

１．３．まとめ

鉢植え作物個体を対象とした葉傾斜角およびその3次元分布の、テクスチャ特徴に基づく計測手法を提案した。本手法は、既存の3次元デジタイザなどによる機械的計測手法と比較すると計測精度にはさらなる改善の余地が認められるものの、現状においても同一品種作物体の形状的な特徴の差を識別する精度は充分有している。特に本手法は、既存の計測手法では不可能であった、葉傾斜角の定量計測の大幅な効率化の実現が可能である点にその最大の特長があり、例えば大規模圃場における大量の幼苗のしおれ状態の監視など、作物の栽培管理を適切かつ効率的に実行する上で、大いに有用と期待できる。

引 用 文 献
(引用順)

楊金宇・蔵田憲次(1991).水分ストレス変化に伴うトマト苗の外観の画像処理による同定.農業気象学会・生物環境調節学会合同大会講演要旨:210-211.

大原源二・細井徳夫・番喜宏(1992).トマトしおれ状態の葉の下垂程度に基づく認知.農業気象学会・生物環境調節学会合同大会講演要旨:38-39.

Witkin, A. P. (1981). Recovering surface shape and orientation from texture. Artif. Intell. 17 :17-45.

庄野浩資・岡田益己・樋口誠一郎(1994a).近接圃場画像に対するテクスチャ解析手法の有効性－牧草混生群落の草種割合推定を例として－.農業気象　49:227-235.

森俊二・坂倉栂子(1990b).パワースペクトラム法.画像認識の基礎[II]、オーム社、東京:200-200.

出口光一郎(1991).投影.画像と空間、昭晃堂、東京:7-38.

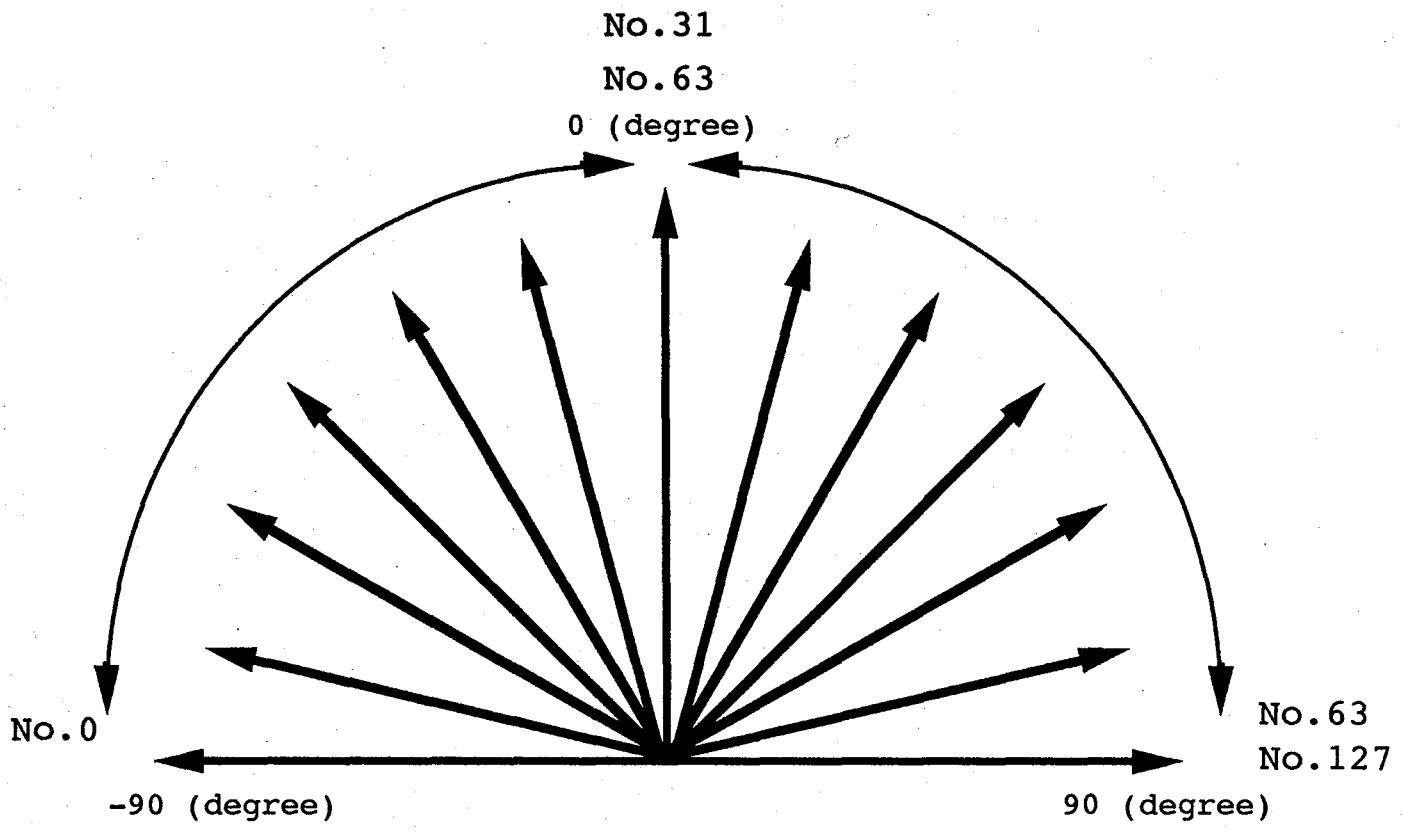
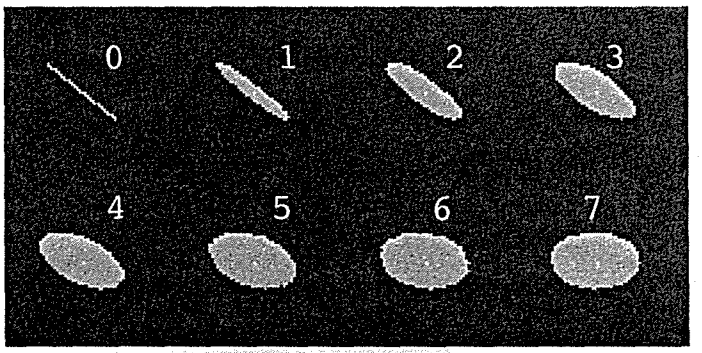
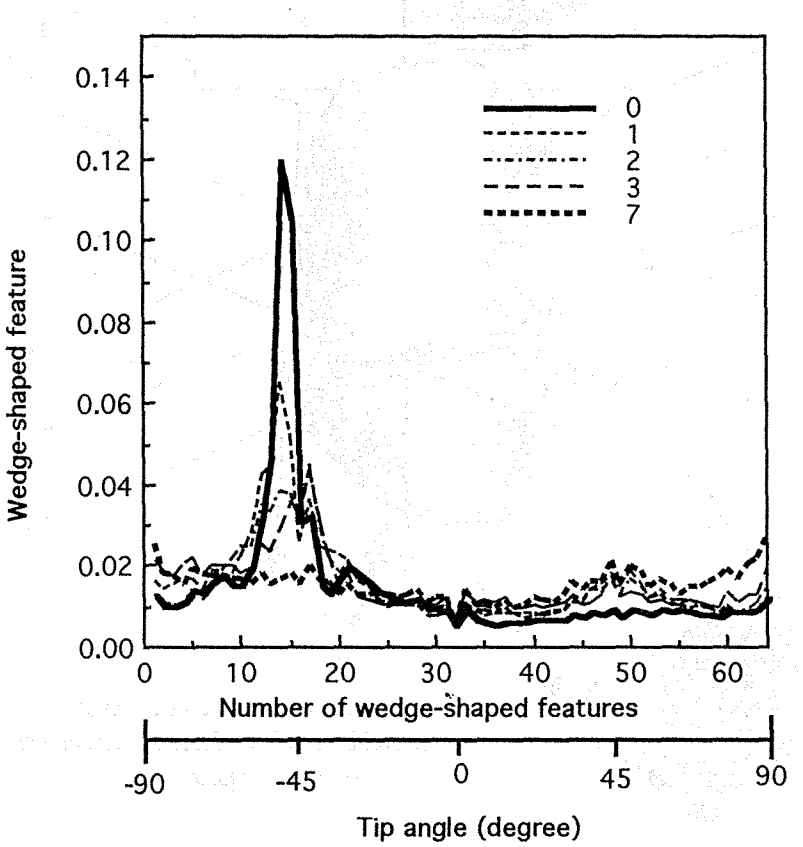


Fig. 2.1-1 Correspondence between tip angle of line in image and number of wedge-shaped feature.



(a)



(b)

Fig. 2.1-2 (a) Sample images of a three dimensional disk, whose tip angle is 45 degrees, taken from 8 directions. (b) Wedge-shaped features calculated from left images.

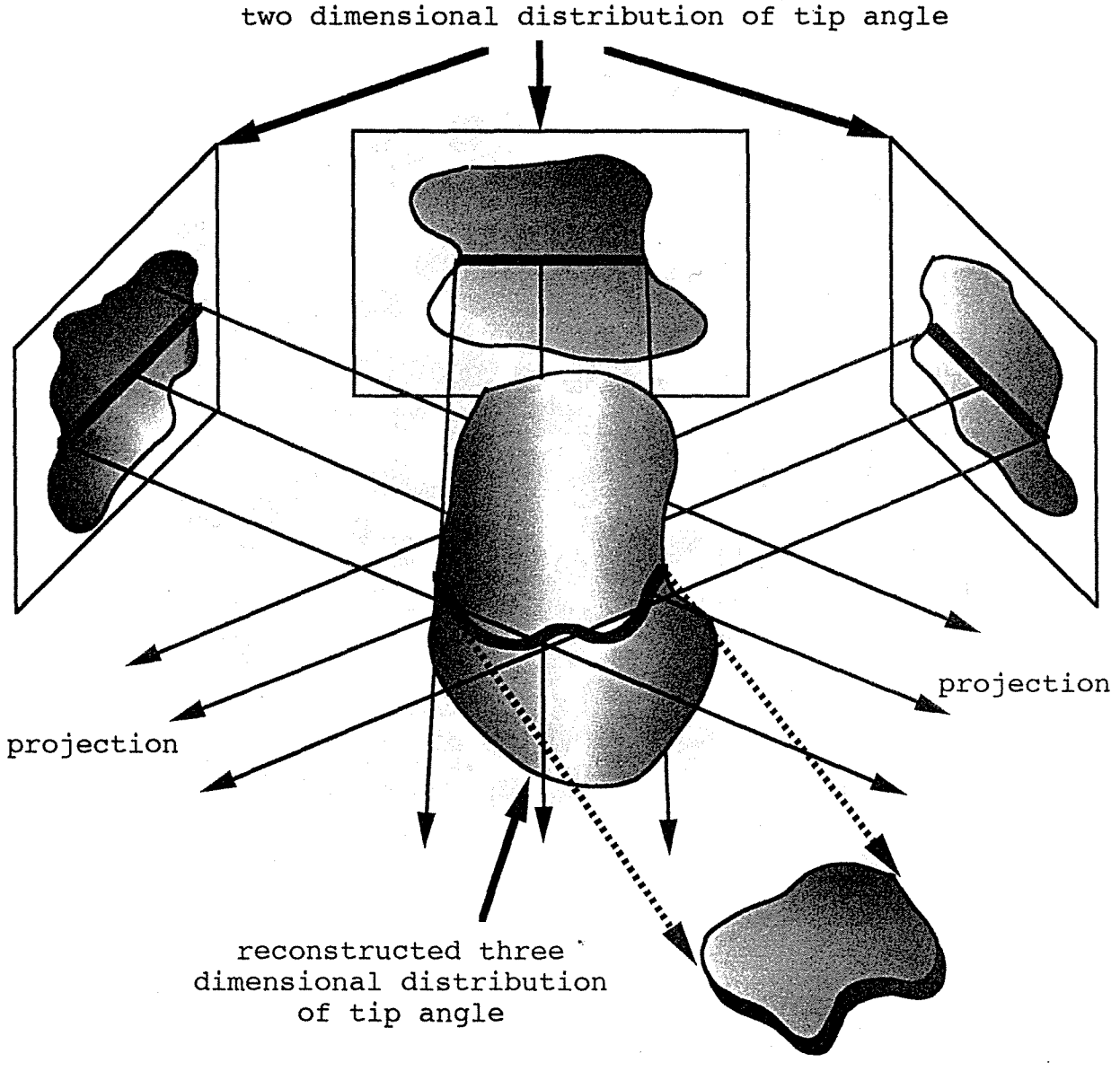


Fig. 2.1-3 Schematic view of the principle of back-projection method.

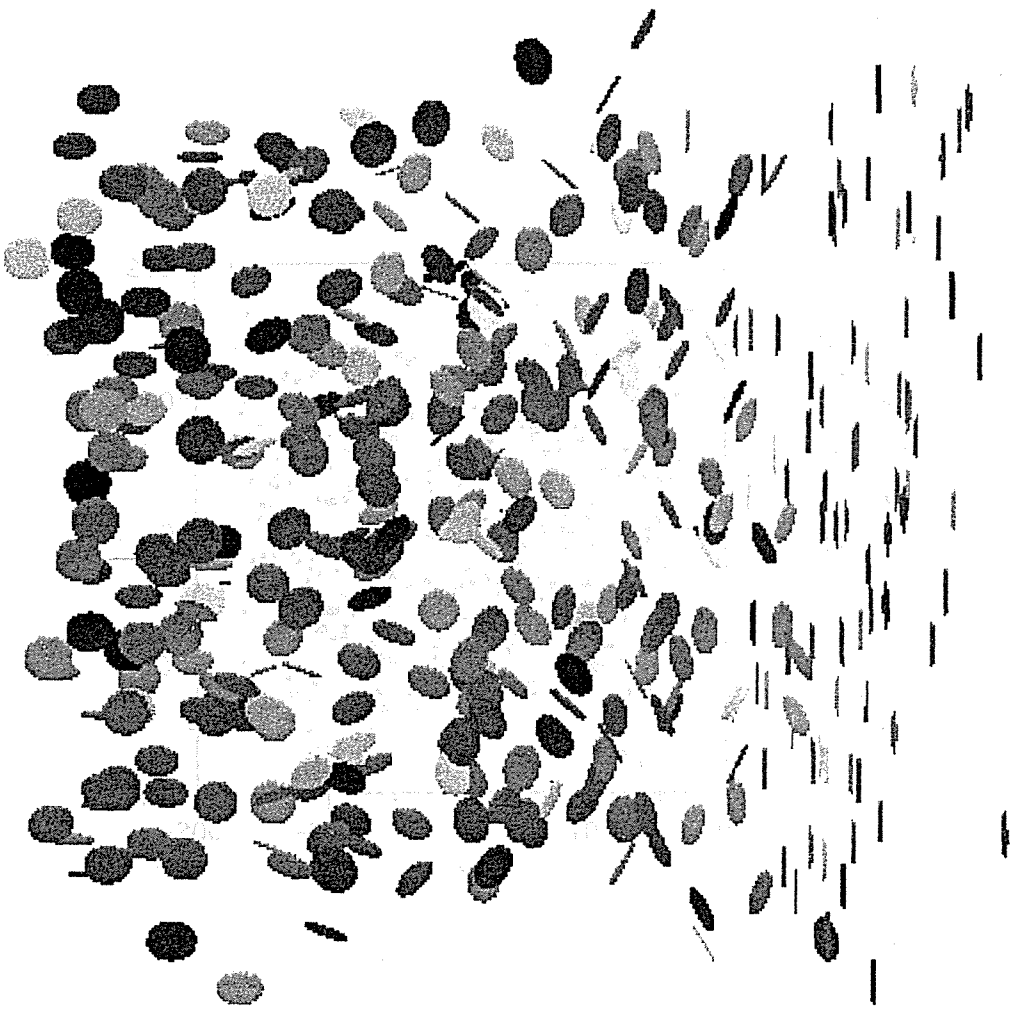


Fig. 2.2-1 Sample image of simulated canopy which is a composite of 5 canopies consist of small three dimensional disks which are distributed randomly in a cubic area.

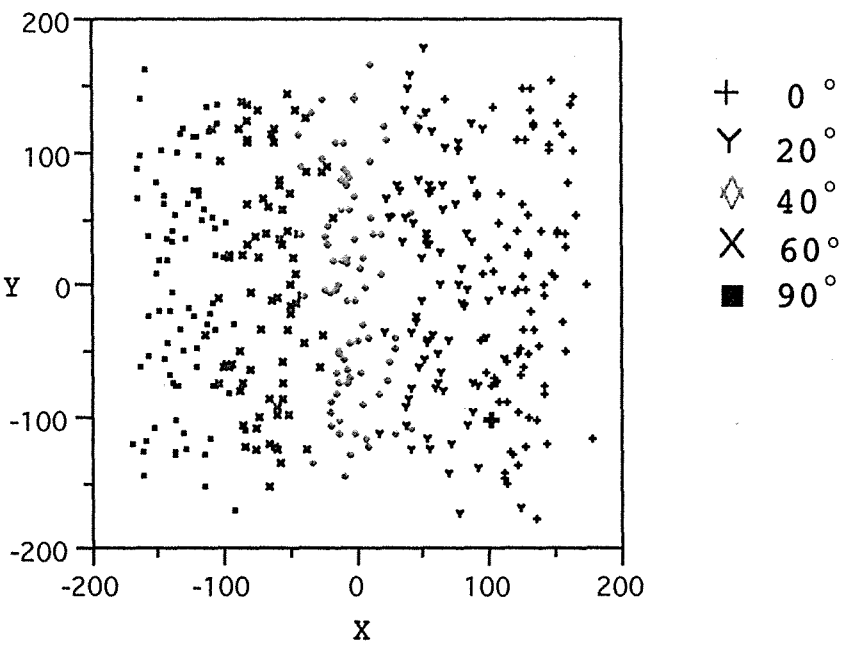


Fig. 2.2-2 Horizontal distribution of tip angles of simulated canopy which is a composite of 5 canopies consist of small three dimensional disks which are distributed randomly in a cubic area.

Fig. 2.2-3 Sample image of simulated canopy which is a composite of 3 layers consist of small three dimensional disks which are distributed randomly.

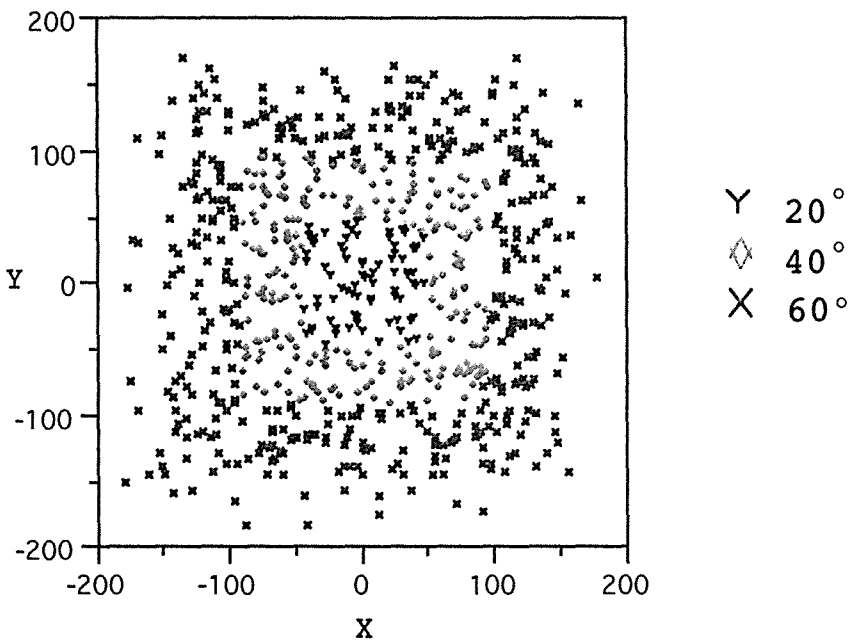


Fig. 2.2-4 Horizontal distribution of tip angles of simulated canopy which is a composite of 3 layers consist of small three dimensional disks which are distributed randomly.

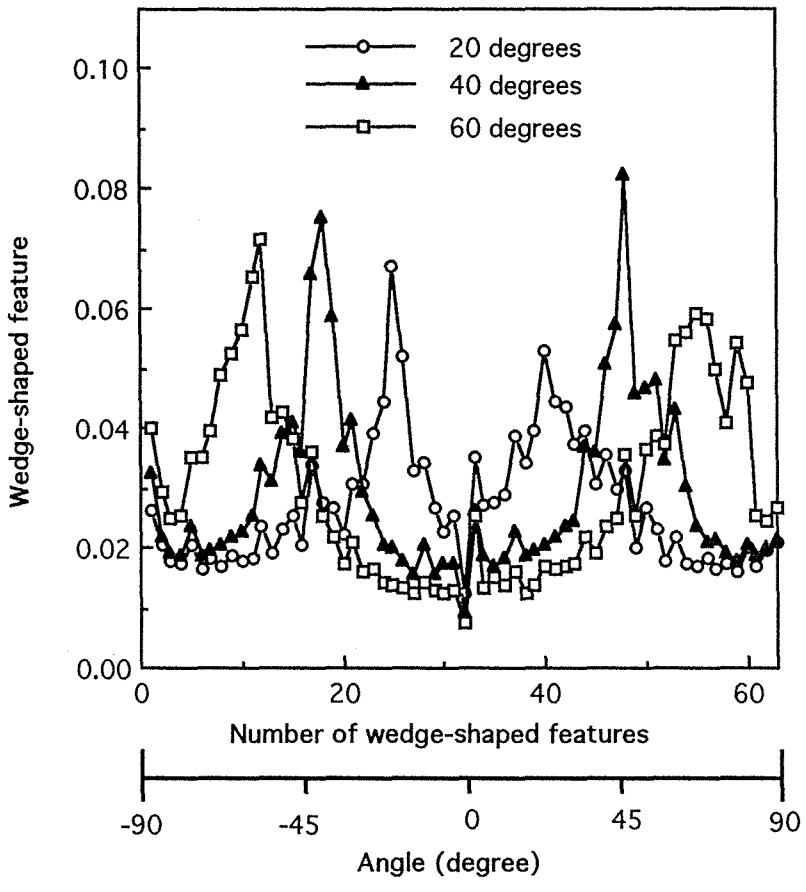


Fig. 2.2-5 Wedge-shaped features calculated from simulated images where small three dimensional disks are distributed randomly in a cubic area.

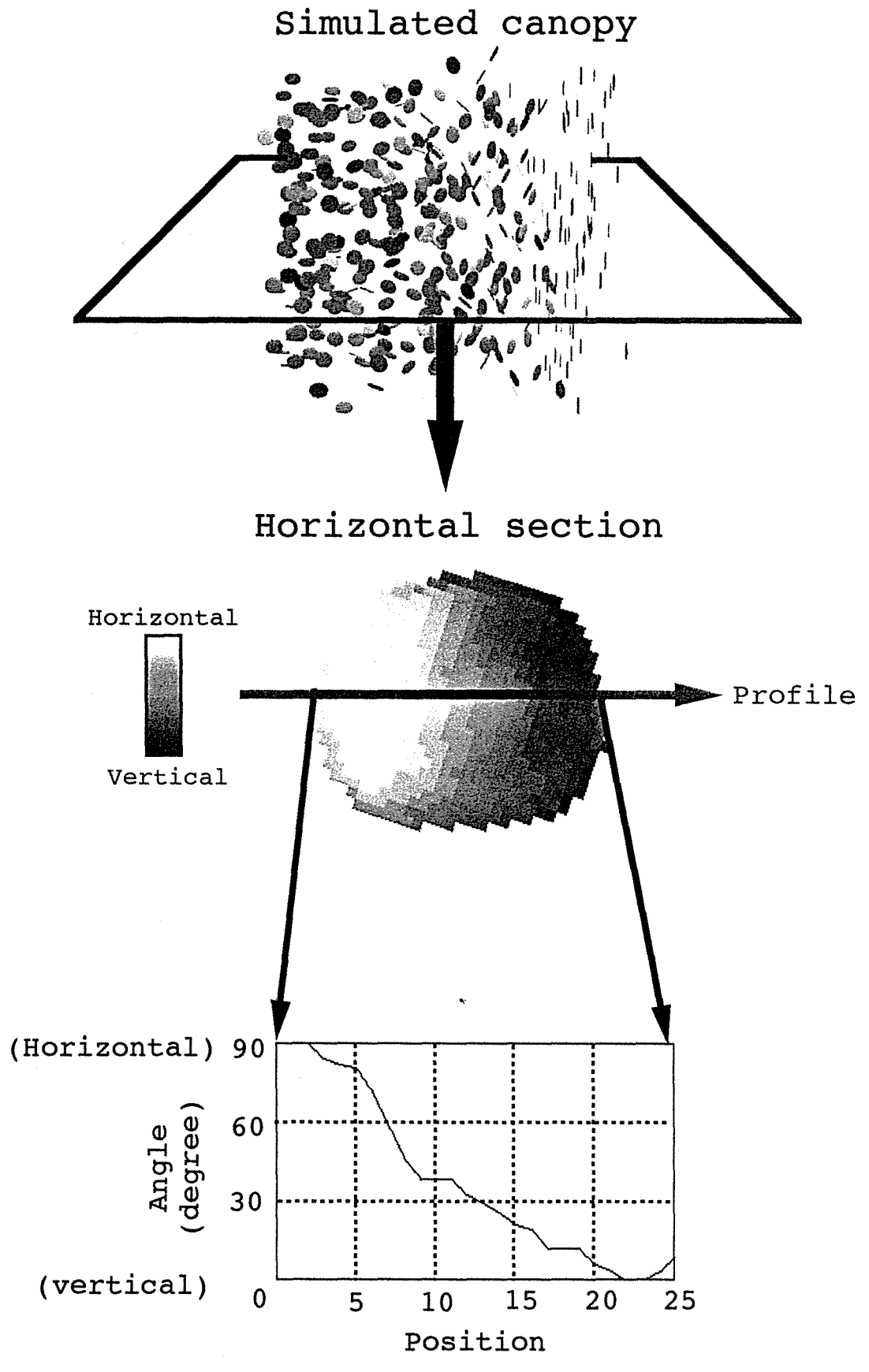


Fig. 2.2-6 Result of reconstruction of tree dimensional distribution of tip angles from simulated images(Fig. 2.2-1).

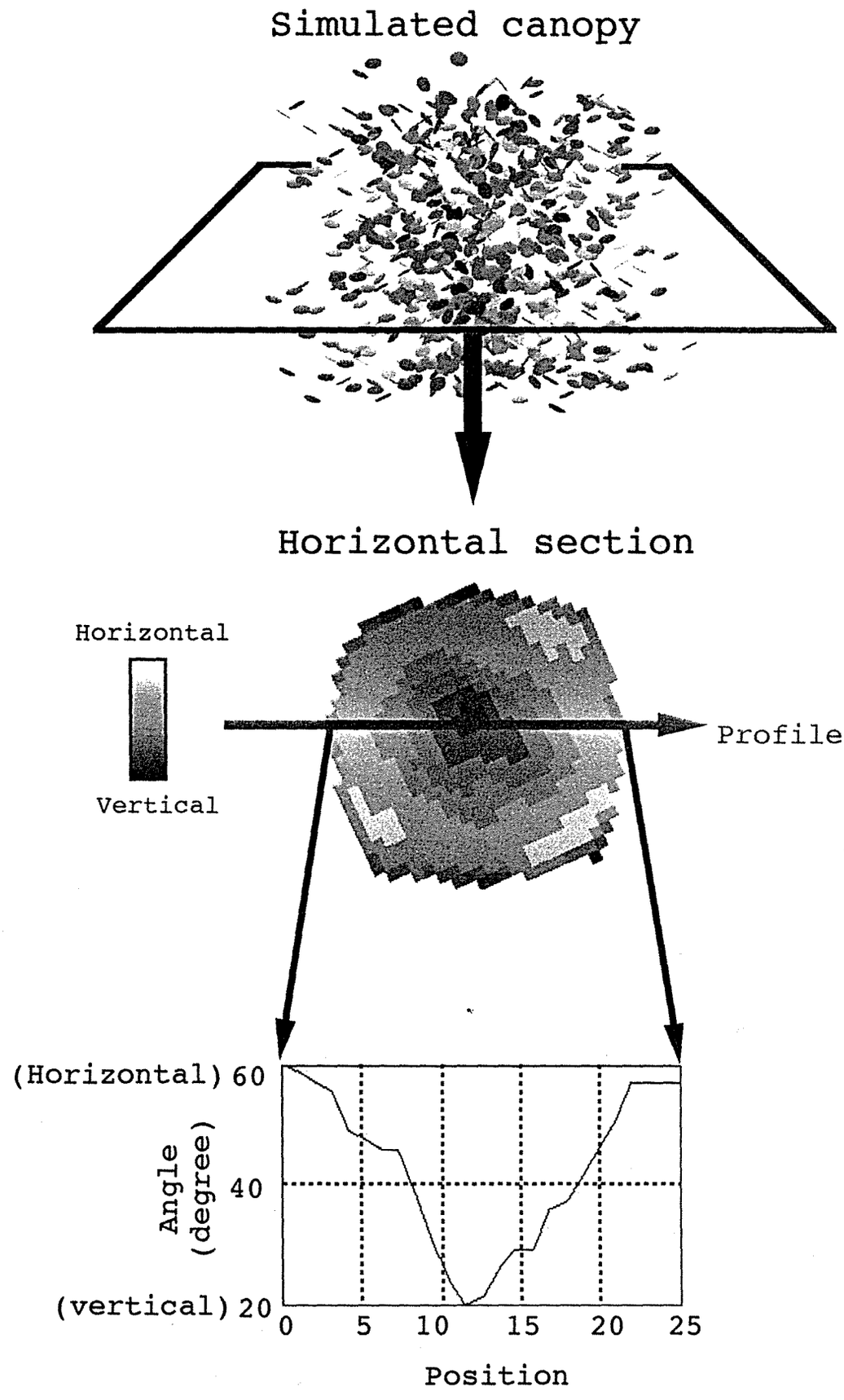


Fig. 2.2-7 Result of reconstruction of tree dimensional distribution of tip angles from simulated images(Fig. 2.2-3).

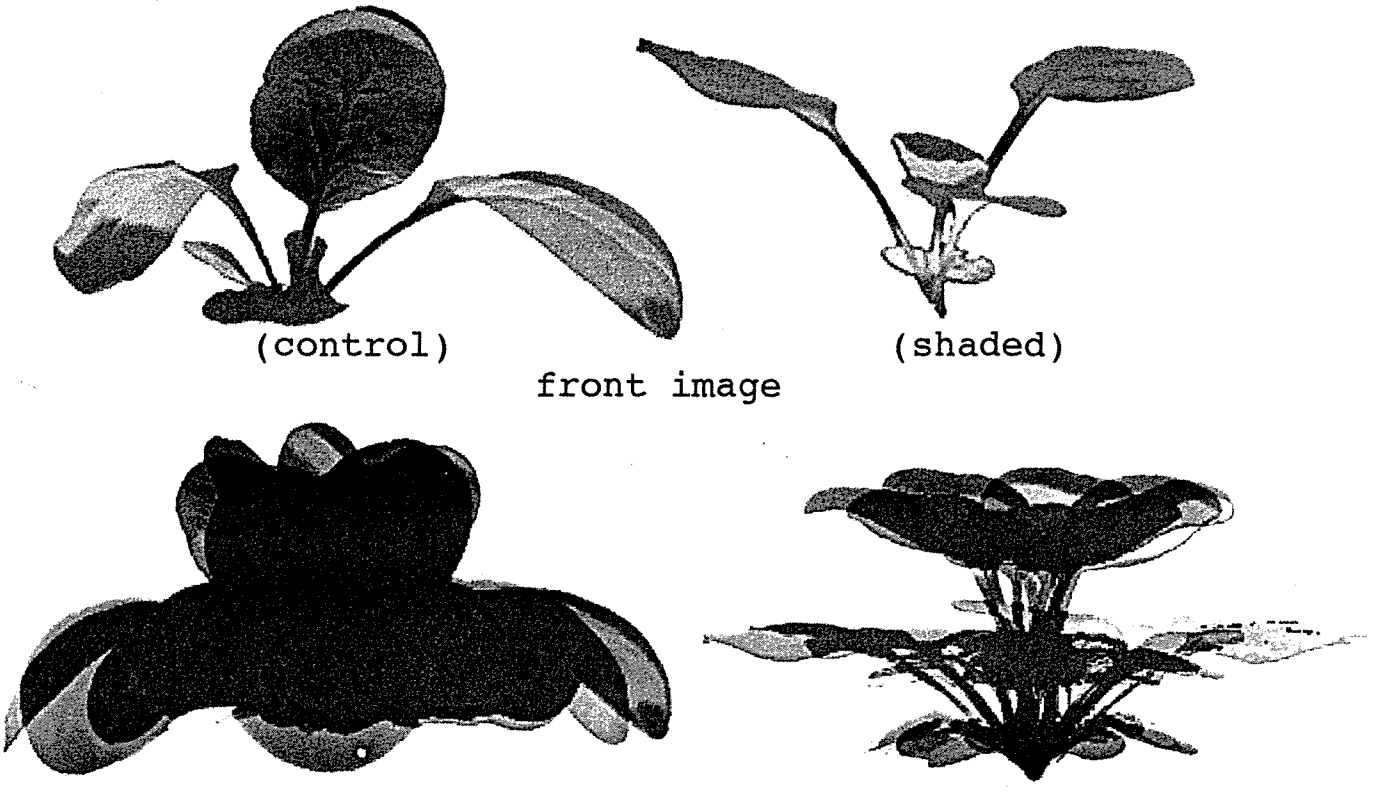
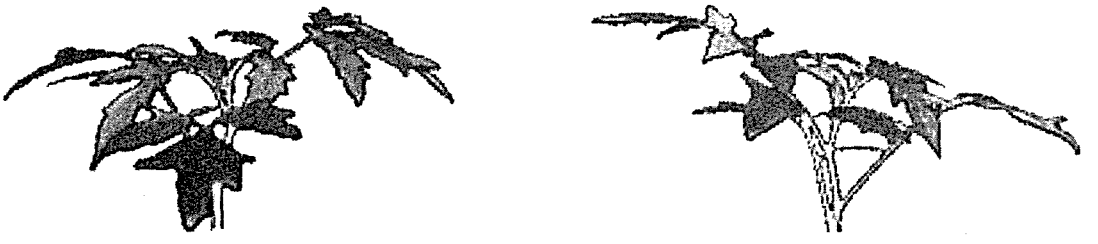


Fig. 2.2-8 Sample images of seedlings of Chinese-cabbage 'Mizuki'.



(control) (over-irrigated)

front image

(control) (over-irrigated)

composite of images
from 4 directions

Fig. 2.2-9 Sample images of seedlings of tomato 'Fujimi'.

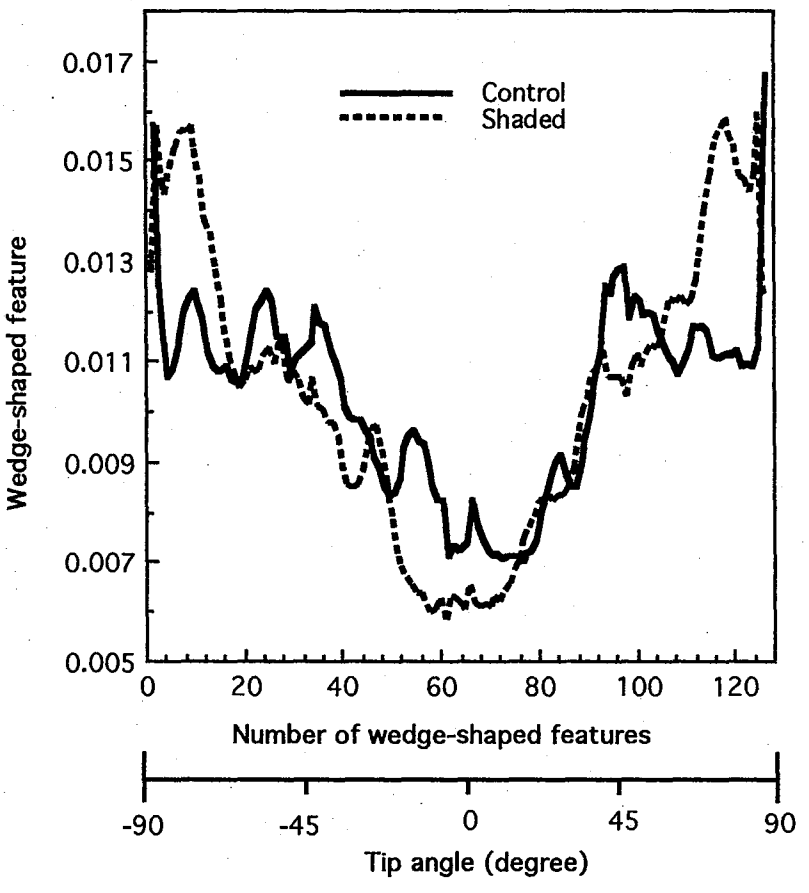
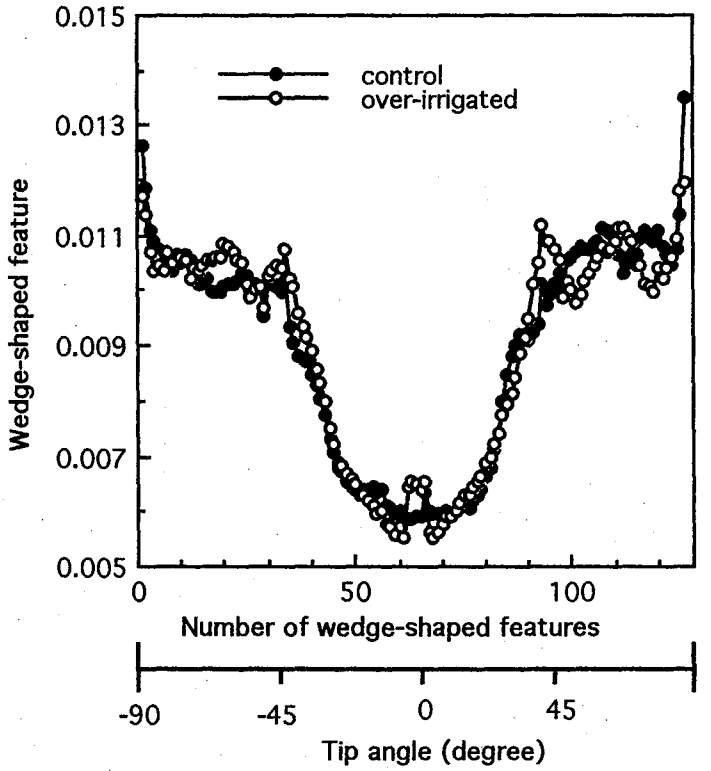
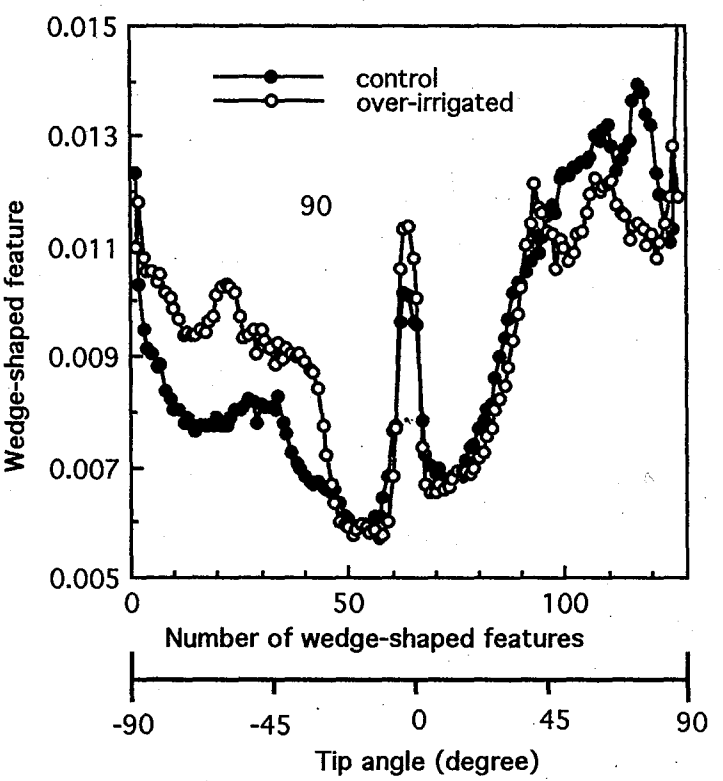


Fig. 2.2-10 Wedge-shaped features calculated from images of seedlings of Chinese-cabbage.



(a)



(b)

Fig. 2.2-11 (a) Wedge-shaped features calculated from entire images of seedlings of tomato. (b) Wedge-shaped features calculated from left halves of images of seedlings of tomato.

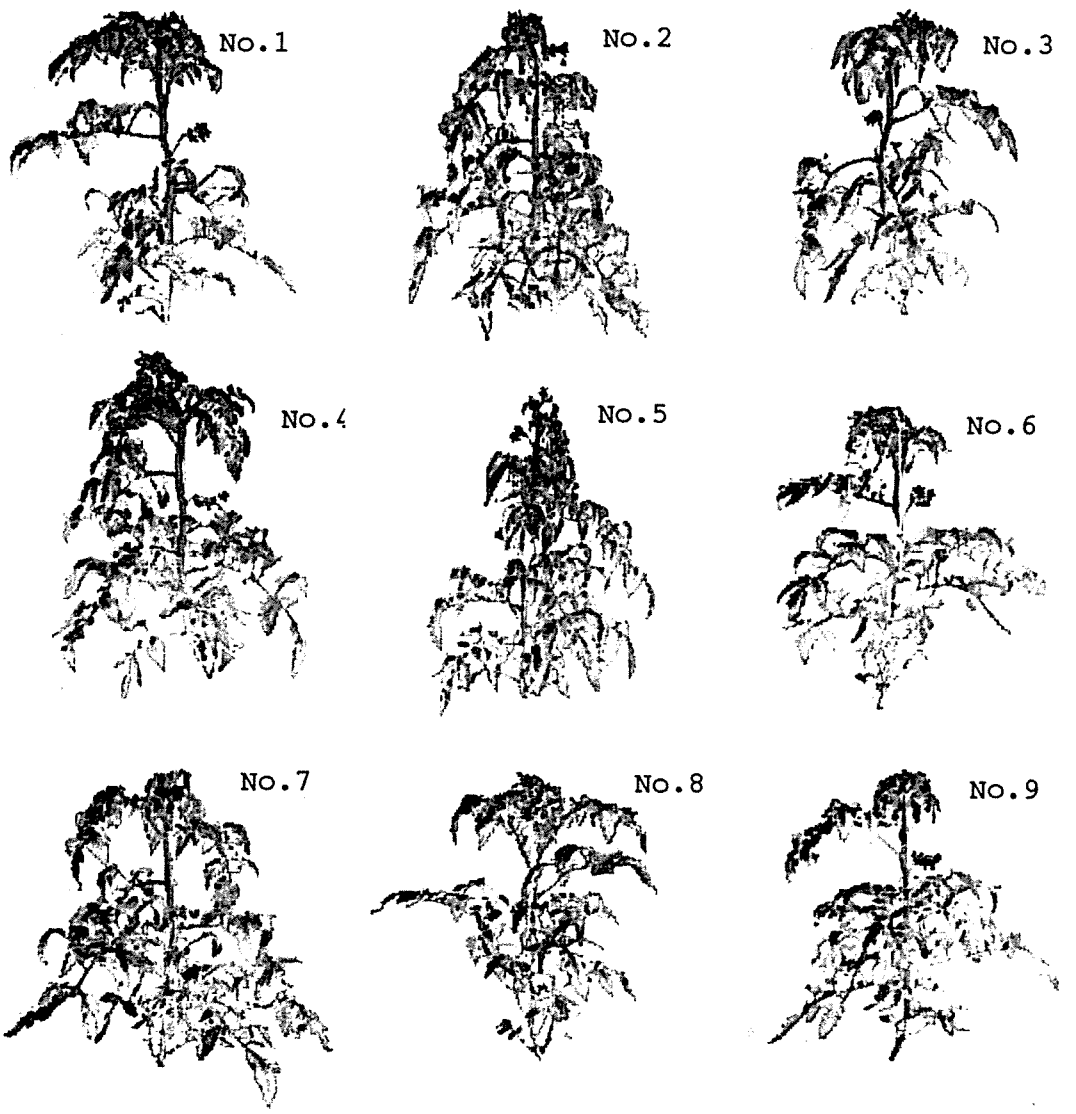
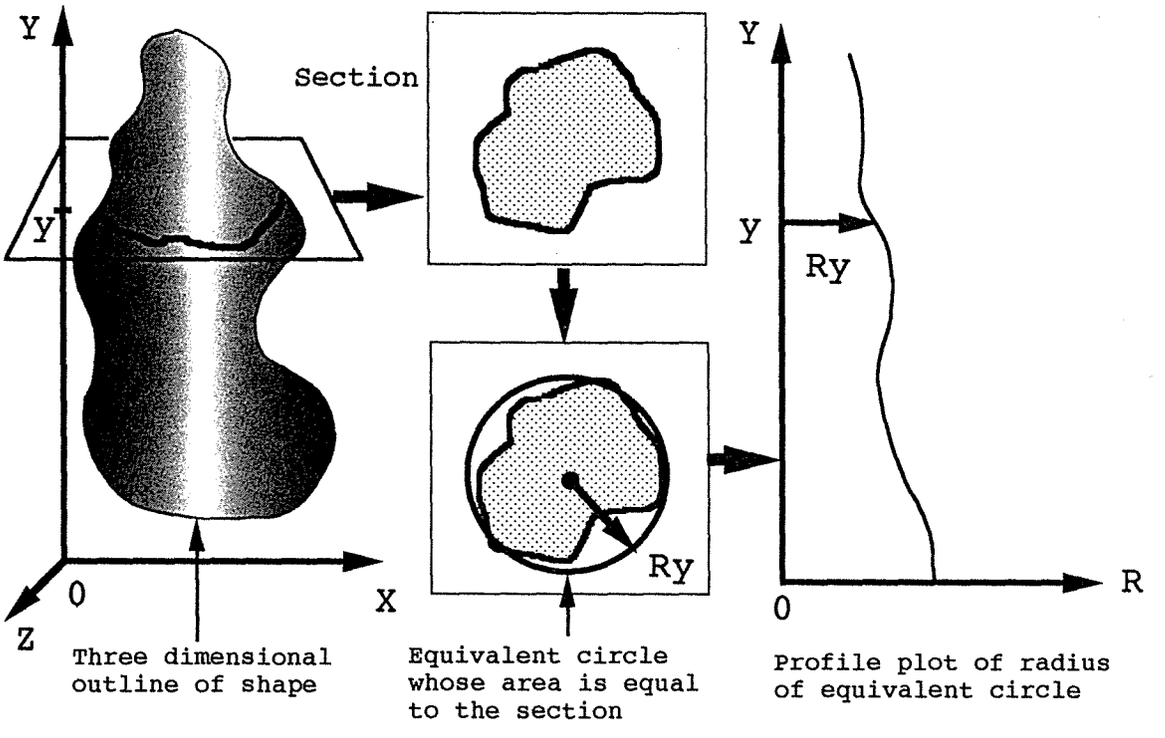
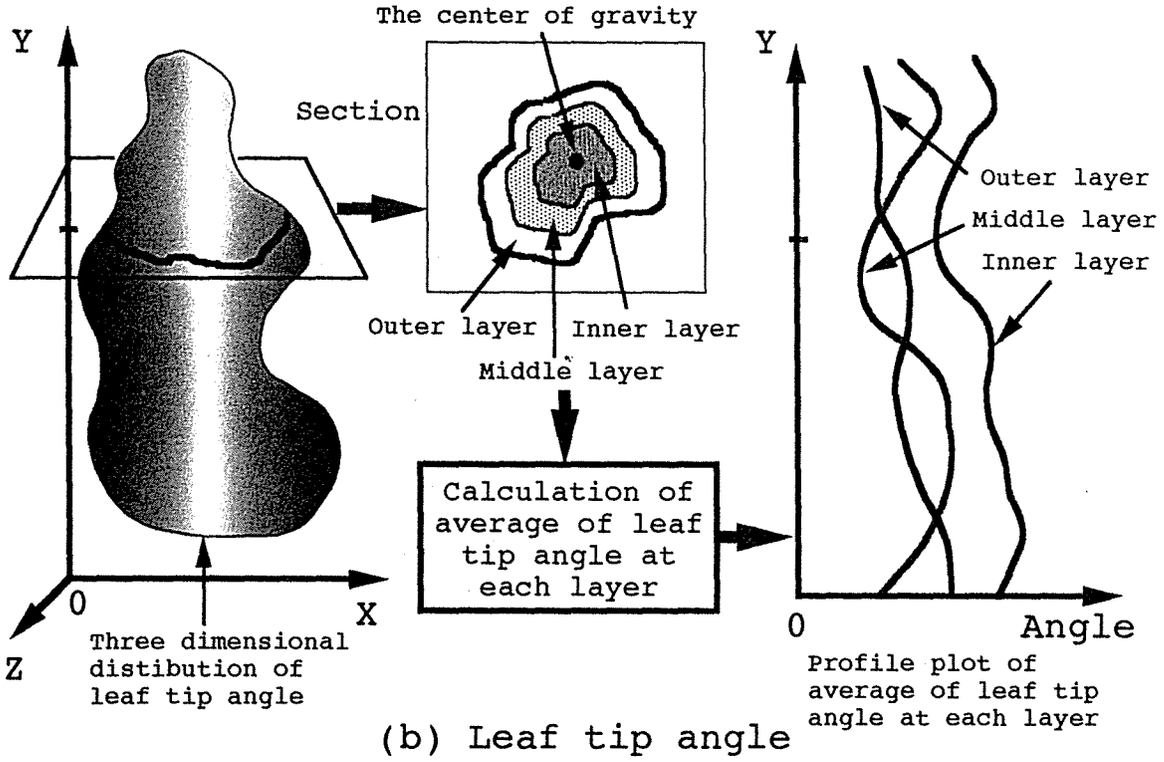


Fig. 2.2-12 Front images of all materials.

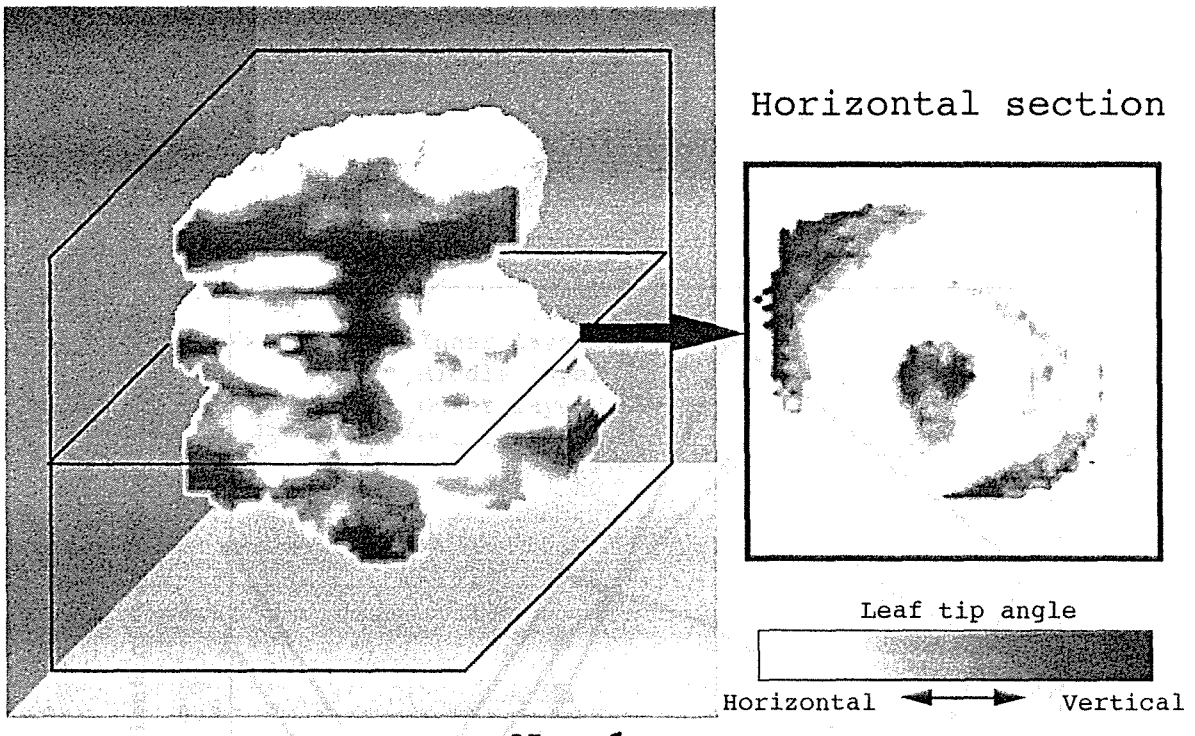


(a) Outline of shape



(b) Leaf tip angle

Fig. 2.2-13 Schematic view of simplification of three dimensional distribution of leaf tip angle and simplified three dimensional outline of shape.



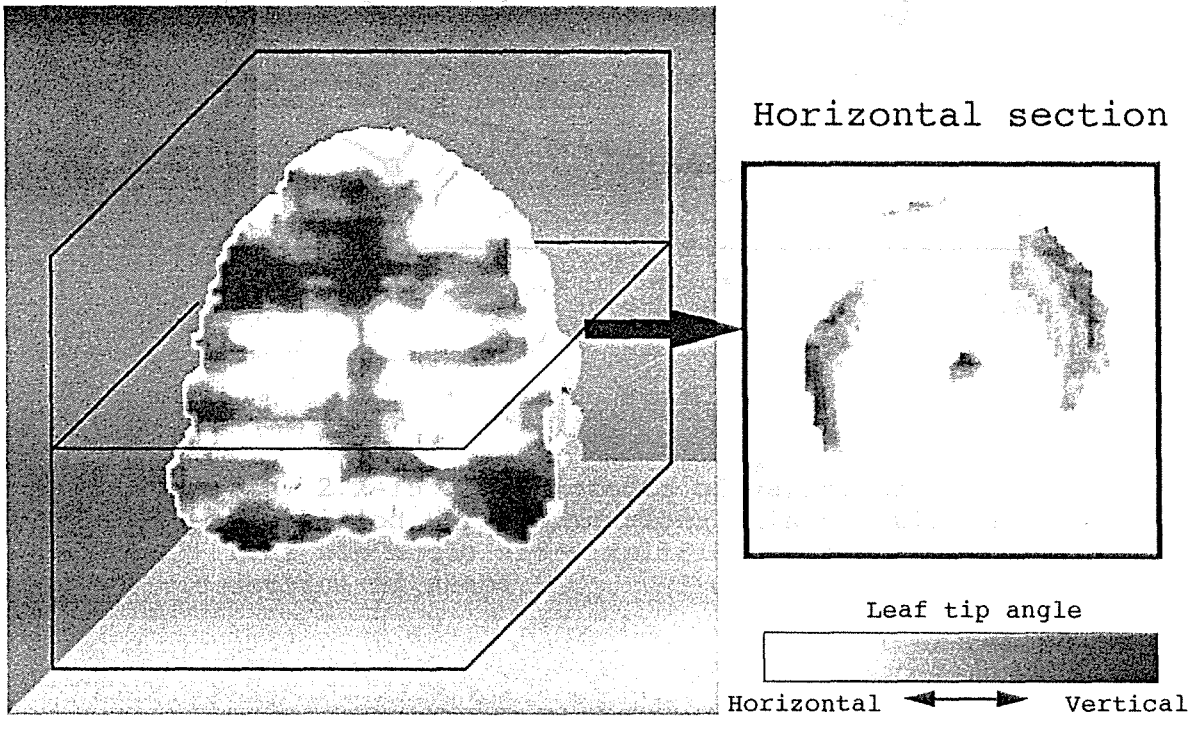


Fig. 2.2-14 Sample results of reconstruction of three dimensional distribution of leaf tip angle.

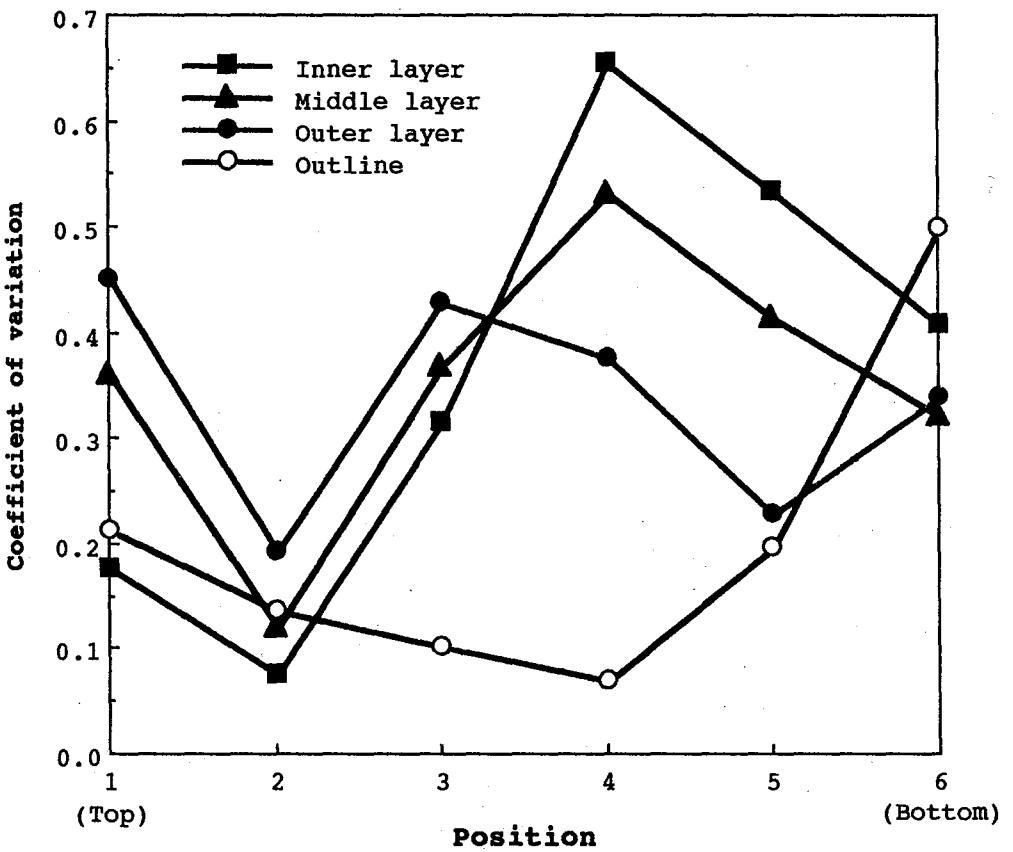
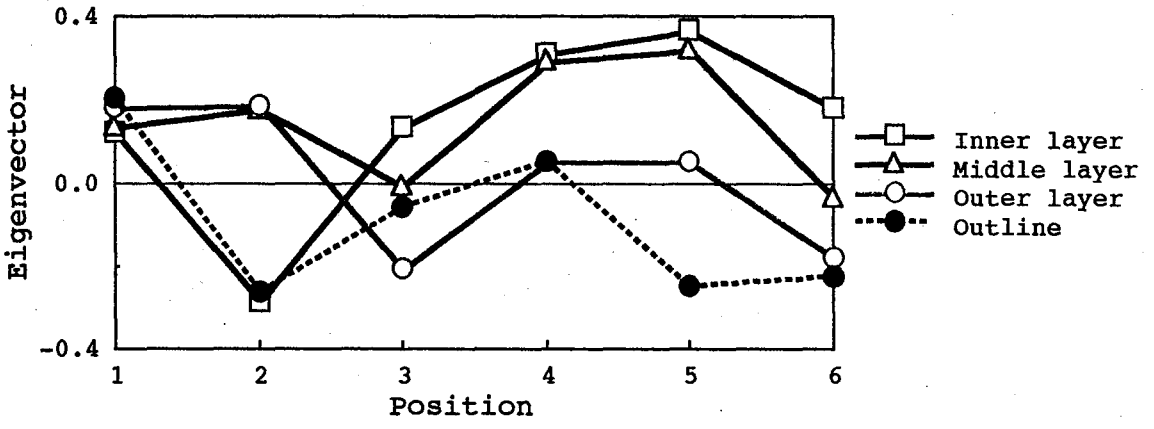
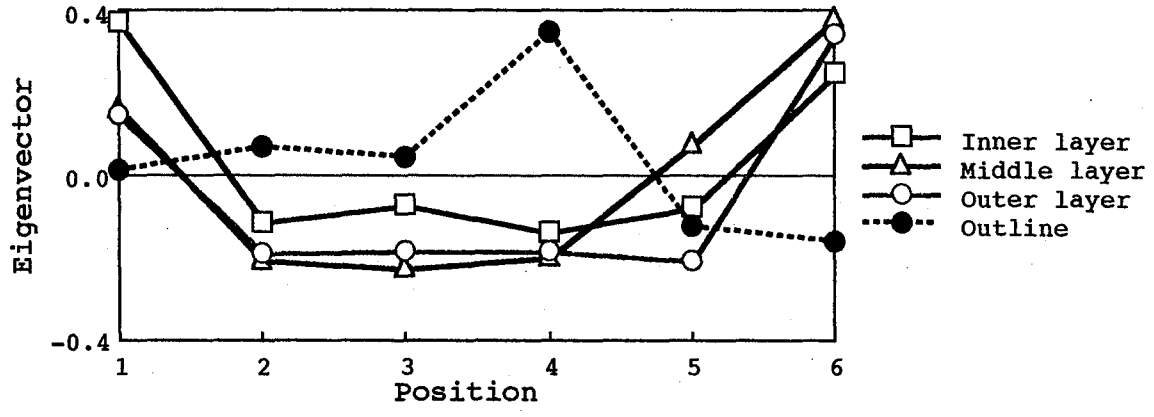


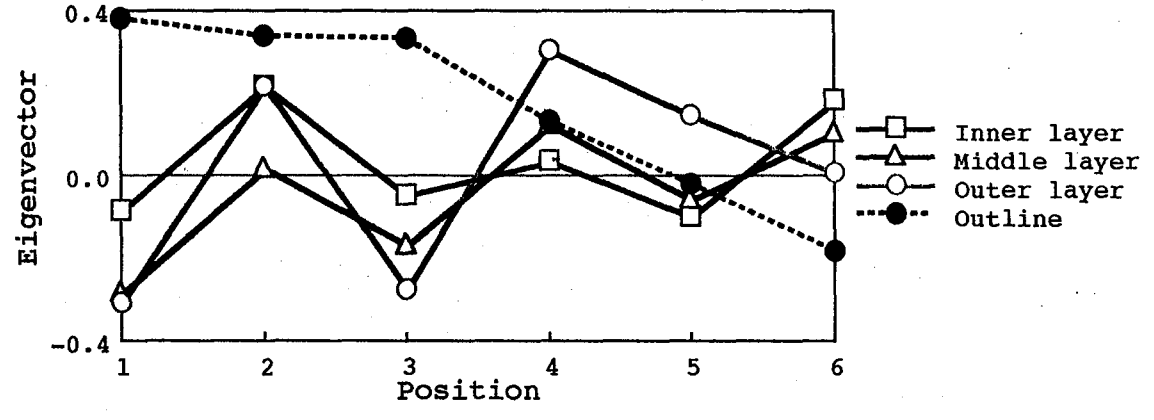
Fig. 2.2-15 Coefficients of variation of simplified three dimensional distribution of leaf tip angle and simplified three dimensional outline of shape.



(a) Principal component No.1



(b) Principal component No.2



(c) Principal component No.3

Fig. 2.2-16 Eigenvectors of Principal component No.1-5 derived from simplified three dimensional distribution of leaf tip angle and simplified three dimensional outline of shape.

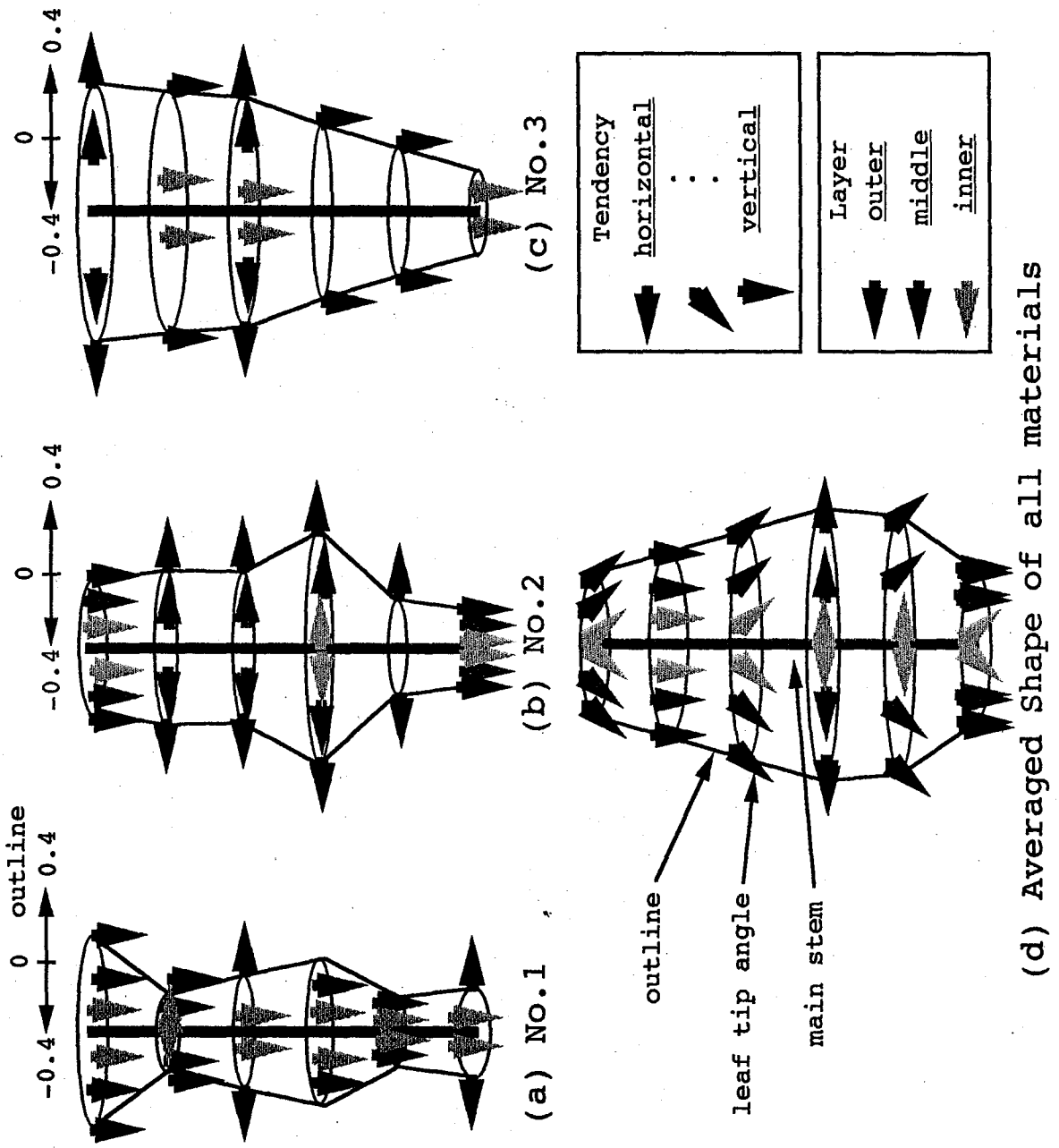


Fig. 2.2-17 Schematic illustrations of meanings of principal component No.1-3.

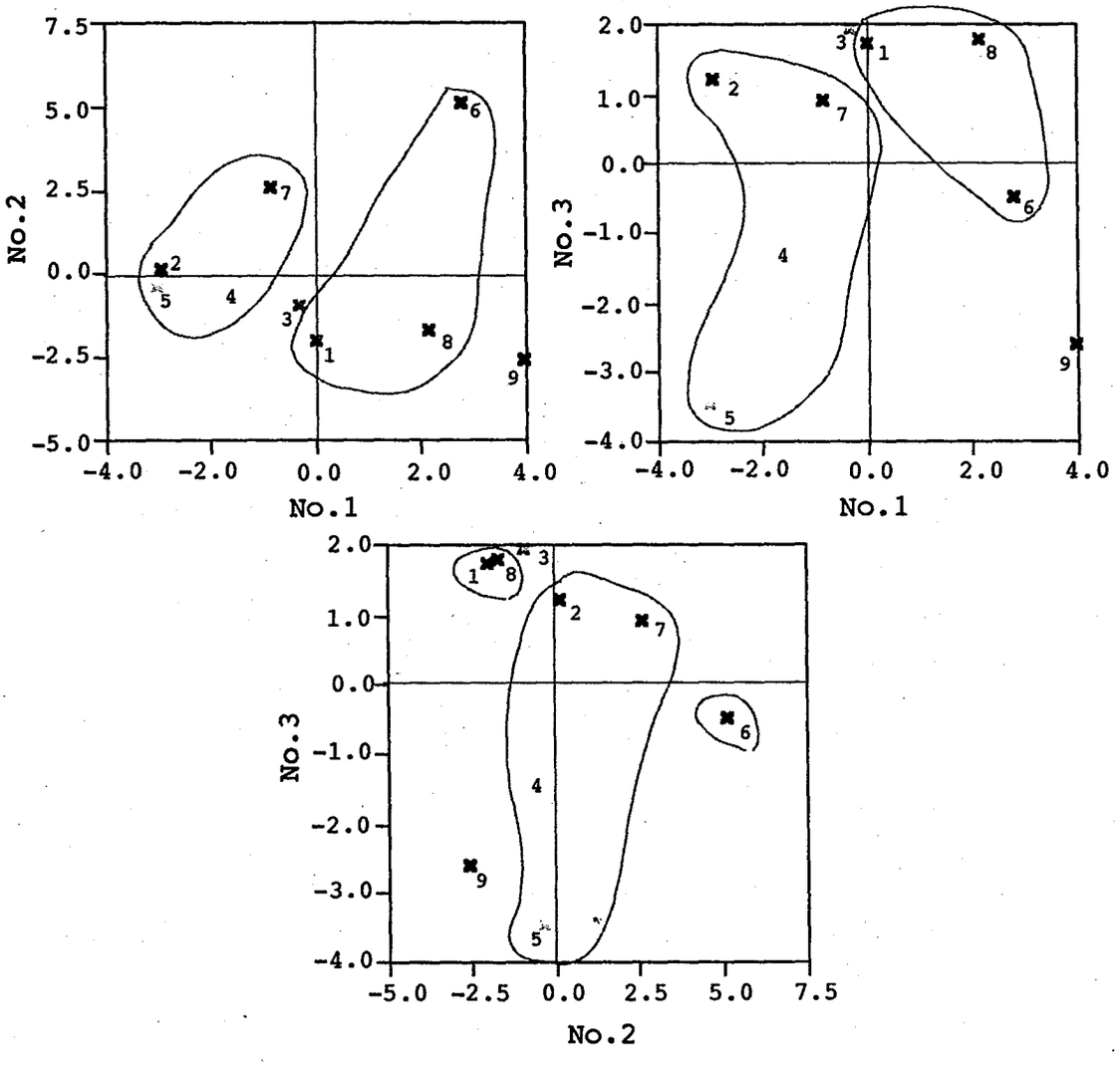


Fig. 2.2-18 Scatter plots of principal component No.1-3.

Table 2.2-1 Correlation coefficients between simplified three dimensional distribution of leaf tip angle and simplified three dimensional outline of shape.

	angle 1	angle 2	angle 3	angle 4	angle 5	angle 6	outline 1	outline 2	outline 3	outline 4	outline 5	outline 6
angle 1	1.000	-0.173	-0.352	-0.207	0.386	0.274	-0.162	-0.521	-0.366	0.444	-0.156	-0.183
angle 2	-0.173	1.000	-0.194	0.508	0.233	-0.391	0.515	0.077	0.039	-0.264	0.446	0.018
angle 3	-0.352	-0.194	1.000	0.087	-0.164	-0.373	-0.400	-0.238	-0.125	-0.557	0.060	0.297
angle 4	-0.207	0.508	0.087	1.000	0.734	-0.336	0.536	-0.366	0.231	-0.137	-0.429	-0.214
angle 5	0.386	0.233	-0.164	0.734	1.000	-0.334	0.293	-0.661	-0.028	0.030	-0.541	-0.292
angle 6	0.274	-0.391	-0.373	-0.336	-0.334	1.000	0.224	0.349	0.301	0.837	-0.257	-0.404
outline 1	-0.162	0.515	-0.400	0.536	0.293	0.224	1.000	0.194	0.237	0.169	-0.292	-0.765
outline 2	-0.521	0.077	-0.238	-0.366	-0.661	0.349	0.194	1.000	0.620	0.272	0.456	0.036
outline 3	-0.366	0.039	-0.125	0.231	-0.028	0.301	0.237	0.620	1.000	0.525	-0.007	0.052
outline 4	0.444	-0.264	-0.557	-0.137	0.030	0.837	0.169	0.272	0.525	1.000	-0.255	-0.267
outline 5	-0.156	0.446	0.060	-0.429	-0.541	-0.257	-0.292	0.456	-0.007	-0.255	1.000	0.577
outline 6	-0.183	0.018	0.297	-0.214	-0.292	-0.404	-0.765	0.036	0.052	-0.267	0.577	1.000

Table 2.2-2 Result of principal component analysis derived from simplified three dimensional distribution of leaf tip angle and simplified three dimensional outline of shape.

	No.1	No.2	No.3	No.4	No.5
EigenValue:	6.3259	5.9231	4.0458	3.2405	1.9915
Percent:	26.3578	24.6797	16.8577	13.5020	8.2977
CumPercent:	26.3578	51.0376	67.8953	81.3972	89.6949
Eigenvectors:					
inner layer 1	0.1242	0.3712	-0.0845	0.0482	-0.0391
inner layer 2	-0.2852	-0.1112	0.2179	0.1714	0.0159
inner layer 3	0.1376	-0.0687	-0.0479	-0.4600	-0.0490
inner layer 4	0.3143	-0.1369	0.0389	-0.2032	-0.0553
inner layer 5	0.3681	-0.0767	-0.0963	0.0345	-0.1339
inner layer 6	0.1814	0.2517	0.1853	-0.1760	0.2535
middle layer 1	0.1363	0.1628	-0.2816	0.3250	0.1318
middle layer 2	0.1792	-0.2067	0.0196	0.2349	0.3937
middle layer 3	-0.0084	-0.2218	-0.1659	-0.3752	0.1683
middle layer 4	0.2920	-0.1943	0.1233	-0.1363	0.1291
middle layer 5	0.3221	0.0772	-0.0553	0.2331	-0.0948
middle layer 6	-0.0333	0.3806	0.1091	-0.0819	0.1399
outer layer 1	0.1819	0.1490	-0.3080	0.2776	0.0278
outer layer 2	0.1859	-0.1853	0.2200	0.1521	0.3618
outer layer 3	-0.2058	-0.1801	-0.2766	-0.0832	-0.0022
outer layer 4	0.0510	-0.1791	0.3075	0.2949	-0.2380
outer layer 5	0.0514	-0.2050	0.1502	0.1917	-0.5030
outer layer 6	-0.1781	0.3451	0.0103	-0.0761	-0.0751
outline 1	0.2074	0.0133	0.3818	-0.0165	0.1236
outline 2	-0.2608	0.0711	0.3383	-0.0089	0.1224
outline 3	-0.0570	0.0481	0.3350	-0.0435	-0.1423
outline 4	0.0552	0.3475	0.1330	0.0692	-0.0094
outline 5	-0.2482	-0.1205	-0.0173	0.2171	0.4088
outline 6	-0.2243	-0.1590	-0.1824	0.1065	0.0035

2．計測プログラムのモジュール化

2．計測プログラムのモジュール化

2．1．計測プログラムの整理・統合化によるモジュール化の概要

既存プログラムは、基本設計において自動化を考慮していないため、計測を一回実行するために様々な個別プログラムの手作業による順次実行が必要であった。さらに、実行途中において不統一なフォーマットによる中間データを取り扱わなければならないなど、プログラム間の連携の非効率さが自動化（サーバー構築）を妨げる大きな要因であった。前年度ではこの点に鑑み、まずプログラム実行プラットフォームをＵｎｉｘ（具体的にはＰＣ上で動作可能なＬｉｎｕｘを採用）と定め、個別に存在するプログラム群を機能毎に整理した。さらに、整理した各プログラムを全面的に記述し直し、Ｕｎｉｘのシェルスクリプトなどの各種スクリプト・ＣＧＩ記述言語から容易に呼び出し可能となるようデータの入出力方法を標準化した。さらに、順次加工・処理される中間データは、Ｕｎｉｘには標準で用意されるいわゆる「パイプ」を用いた不可視なデータストリームとして各プログラム間で受け渡されるようにした。このため、スクリプト記述者は、初期の画像データのフォーマットに関する取り決めを守りさえすれば、中間結果のデータフォーマットなどに頓着することなく希望する計測結果を得ることが可能となり、将来の自動演算サーバーを構築することが容易となると考えられる。

今回、新規に開発したプログラムモジュールは以下の通りである。初期入力画像のフォーマットは汎用のＰＧＭフォーマットとしているが、変換プログラムを使用することにより、ＧＩＦ、ＢＭＰ、ＰＩＣＴなどのフォーマットにも対応可能である。各モジュールの連携方法などの詳細に関しては省略する。

a）２次元フーリエ変換計算に関するモジュール

fourier.exe　　fourierbatch.exe
fourierpipe.exe　　fourierbatchpipe.exe

b）テクスチャ特徴量計算および２次元マップ化に関するモジュール

fouriermap.exe　　fouriermappipe.exe

c）葉傾斜角の３次元分布計算に関するモジュール

3dscan.exe　　3dscanpipe.exe

d）葉傾斜角の３次元分布の１次元プロファイル化に関するモジュール

ringpipe.exe

各モジュールを用いて葉傾斜角の３次元分布を計算し、さらに１次元のプロファイルを求める手順を、利用ＯＳを標準的なＵｎｉｘとして説明する。

多方向から対象を撮影した白黒画像群が「test***.pgm」なるファイル名である場合、

```
% fourier.exe test***.pgm ¦ 3dscan.exe ¦ ringpipe.exe > PROFILE.DAT
```

のようにコマンド起動するだけで、「PROFILE.DAT」なるプロファイルデータを得ることが出来る。白黒画像のファイル名の指定方法は、「fourier.exe」、「fourierbatch.exe」、「fourierpipe.exe」、「fourierbatchpipe.exe」で異なり、使用者の都合の良い方法を選択できるが、ここでは説明を省略する。

２．２．プログラムソース

以下に、今回作成した全てのプログラムのソースを掲載する。プログラムはＡＮＳＩ準拠のＣ言語で記述されている。ＧＮＵ版Ｃコンパイラによりコンパイル可能な設計となっているため、いわゆるＰＣ版Ｕｎｉｘ、例えばＬｉｎｕｘなどの環境があれば即時コンパイル・実行が可能である。

ａ）２次元フーリエ変換計算に関するモジュール

（１）fourier.exe

本モジュールは「fft3d.c」、「fourier7.c」で構成される。

fft3d.c ：　２次元ＦＦＴ計算

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>

#define		MAX_POINTS		256
#define		MAX_ITERATION		10
#define		GRAY_SCALE		255
#define		LAMBDA			1.0
#define		STEP			0.1
#define		NITCH			1.0e-15
#define		NITCH2			1.0e-15

#define		CONTINUE		-1
#define		BREAK			-2

#define		NG			-1
#define		OK			0
#define		OPEN_ERR		1
#define		DATA_ERR		2
#define		WRITE_ERR		3
#define		MEM_ERR			4
#define		CALC_ERR		5

#define		NORMAL			1
#define		INVERSE			-1

#define		NOT_FOUND		0
#define		FOUND			1

#define		PI			3.1415927

double	sintbl[MAX_POINTS], sintbli[MAX_POINTS], costbl[MAX_POINTS];
int		first_call_n = 1, first_call_i = 1;
int		m[MAX_POINTS];

FFT(x, y, n, f)
double	*x, *y;
int		n;
int		f;
```

```
{
        if (f == INVERSE) {
                if (first_call_i) {
                        FFTinverse1(x, y, n);
                        first_call_i = 0;
                }
                else
                        FFTinverse(x, y, n);
        }
        else {
                if (first_call_n) {
                        FFTnormal1(x, y, n);
                        first_call_n = 0;
                }
                else
                        FFTnormal(x, y, n);
        }

        return OK;
}
FFTnormal1(x, y, n)
double  *x, *y;
int             n;
{
        int             i, i0, i1, j, l1, ns, n1, arg, arg2;
        double  s, c, sc, x1, y1, t;

        n1 = n/2;
        ns = n/2;
        sc = 2*PI/(double)n;

/*      fprintf(stderr, "FFTnormal: first call!¥n");    */

        for (i = 0; i < MAX_POINTS; i++)
                m[i] = 0;

        while (ns >= 1) {
                for (l1 = 0; l1 < n; l1 += (2*ns)) {
                        arg = m[l1]/2;
                        arg2 = arg + n1;

                        c = cos(sc*(double)arg);
                        s = sin(-1.0*sc*(double)arg);

                        if (arg < 0 || arg >= MAX_POINTS) {

                                fprintf(stderr, "!!! error !!! arg = %d ...¥n", arg);
                                exit(1);
                        }

                        sintbl[arg] = s;
                        costbl[arg] = c;

                        for (i0 = l1; i0 < l1 + ns; i0++) {
```

```
                        i1 = i0 + ns;
                        x1 = x[i1]*c - y[i1]*s;
                        y1 = y[i1]*c + x[i1]*s;
                        x[i1] = x[i0] - x1;
                        y[i1] = y[i0] - y1;
                        x[i0] = x[i0] + x1;
                        y[i0] = y[i0] + y1;
                        m[i0] = arg;
                        m[i1] = arg2;
                    }
                }
                ns = ns/2;
        }

        for (i = 0; i < n; i++)
                if ((j = m[i]) > i) {
                        t = x[i];
                        x[i] = x[j];
                        x[j] = t;
                        t = y[i];
                        y[i] = y[j];
                        y[j] = t;
                }

        return OK;
}
FFTnormal(x, y, n)
double  *x, *y;
int             n;
{
        int             i, i0, i1, j, l1, ns, n1, arg, arg2;
        double  s, c, sc, x1, y1, t;

        n1 = n/2;
        ns = n/2;
        sc = 2*PI/(double)n;

        for (i = 0; i < MAX_POINTS; i++)
                m[i] = 0;

        while (ns >= 1) {
                for (l1 = 0; l1 < n; l1 += (2*ns)) {
                        arg = m[l1]/2;
                        arg2 = arg + n1;

                        c = costbl[arg];
                        s = sintbl[arg];

                        for (i0 = l1; i0 < l1 + ns; i0++) {
                                i1 = i0 + ns;
                                x1 = x[i1]*c - y[i1]*s;
                                y1 = y[i1]*c + x[i1]*s;
                                x[i1] = x[i0] - x1;
                                y[i1] = y[i0] - y1;
```

```
                        x[i0] = x[i0] + x1;
                        y[i0] = y[i0] + y1;
                        m[i0] = arg;
                        m[i1] = arg2;
                    }
                }
                ns = ns/2;
        }

        for (i = 0; i < n; i++)
                if ((j = m[i]) > i) {
                        t = x[i];
                        x[i] = x[j];
                        x[j] = t;
                        t = y[i];
                        y[i] = y[j];
                        y[j] = t;
                }

        return OK;
}
FFTinverse1(x, y, n)
double  *x, *y;
int             n;
{
        int             i, i0, i1, j, l1, ns, n1, arg, arg2;
        double  s, c, sc, x1, y1, t;

        n1 = n/2;
        ns = n/2;
        sc = 2*PI/(double)n;

        fprintf(stderr, "FFTinverse: first call!¥n");

        for (i = 0; i < MAX_POINTS; i++)
                m[i] = 0;

        while (ns >= 1) {
                for (l1 = 0; l1 < n; l1 += (2*ns)) {
                        arg = m[l1]/2;
                        arg2 = arg + n1;

                        c = cos(sc*(double)arg);
                        s = sin(sc*(double)arg);

                        if (arg < 0 || arg >= MAX_POINTS) {

                                fprintf(stderr, "!!! error !!! arg = %d ...¥n", arg);
                                exit(1);
                        }

                        sintbli[arg] = s;
                        costbl[arg] = c;
```

```
                        for (i0 = l1; i0 < l1 + ns; i0++) {
                                i1 = i0 + ns;
                                x1 = x[i1]*c - y[i1]*s;
                                y1 = y[i1]*c + x[i1]*s;
                                x[i1] = x[i0] - x1;
                                y[i1] = y[i0] - y1;
                                x[i0] = x[i0] + x1;
                                y[i0] = y[i0] + y1;
                                m[i0] = arg;
                                m[i1] = arg2;
                        }
                }
                ns = ns/2;
        }

        for (i = 0; i < n; i++) {
                x[i] /= (double)n;
                y[i] /= (double)n;
        }

        for (i = 0; i < n; i++)
                if ((j = m[i]) > i) {
                        t = x[i];
                        x[i] = x[j];
                        x[j] = t;
                        t = y[i];
                        y[i] = y[j];
                        y[j] = t;
                }


        return OK;
}
FFTinverse(x, y, n)
double  *x, *y;
int             n;
{
        int             i, i0, i1, j, l1, ns, n1, arg, arg2;
        double  s, c, sc, x1, y1, t;

        n1 = n/2;
        ns = n/2;
        sc = 2*PI/(double)n;

        for (i = 0; i < MAX_POINTS; i++)
                m[i] = 0;

        while (ns >= 1) {
                for (l1 = 0; l1 < n; l1 += (2*ns)) {
                        arg = m[l1]/2;
                        arg2 = arg + n1;

                        c = costbl[arg];
                        s = sintbli[arg];
```

```
                        for (i0 = l1; i0 < l1 + ns; i0++) {
                                i1 = i0 + ns;
                                x1 = x[i1]*c - y[i1]*s;
                                y1 = y[i1]*c + x[i1]*s;
                                x[i1] = x[i0] - x1;
                                y[i1] = y[i0] - y1;
                                x[i0] = x[i0] + x1;
                                y[i0] = y[i0] + y1;
                                m[i0] = arg;
                                m[i1] = arg2;
                        }
                }
                ns = ns/2;
        }

        for (i = 0; i < n; i++) {
                x[i] /= (double)n;
                y[i] /= (double)n;
        }

        for (i = 0; i < n; i++)
                if ((j = m[i]) > i) {
                        t = x[i];
                        x[i] = x[j];
                        x[j] = t;
                        t = y[i];
                        y[i] = y[j];
                        y[j] = t;
                }


        return OK;
}
```

fourier7.c : パワースペクトラム法計算プログラム本体

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


#define         MAX_POINTS          1280
#define         MAX_FFT_NUMBER      512
#define         GRAY_SCALE          256
#define         MAX_DELTA           512
#define         DELTA               128
#define         SKIP_WEDGE          10
#define         SKIP                2
#define         SMOOTH                      2

#define         NORMAL                      1
#define         INVERSE                     -1
```

```
#define         NG                      -1
#define         OK                      0
#define         OPEN_ERR        1
#define         DATA_ERR        2
#define         WRITE_ERR       3
#define         MEM_ERR                 4

#define         ON                      1
#define         OFF                     0

#define         PI                      3.1415927

#define         MASK_FILE       "mask128.raw"

#define         REAL_EQ                 1

int                     gray_scale;
int                     kind_of_eq = REAL_EQ;
int                     maskuse = OFF;
int                     skip = 0;
int                     smooth = SMOOTH;
int                     normalize = ON;


int     main(int argc, char *argv[])
{

        FILE                    *fpo;
        char                    source[100], destination[100], maskfile[100];
        unsigned char   (*buffer)[MAX_DELTA][MAX_DELTA];
        unsigned char   (*original)[MAX_POINTS][MAX_POINTS];
        double                  (*area_x)[MAX_FFT_NUMBER][MAX_FFT_NUMBER];
        double                  (*area_y)[MAX_FFT_NUMBER][MAX_FFT_NUMBER];
        unsigned char   (*mask)[MAX_DELTA][MAX_DELTA];
        int                             len_x, len_y, skip_x, skip_y;
        int                             delta = MAX_DELTA;
        int                             base_x, base_y, serial;
        int                             offset_x = 0, offset_y = 0;
        int                             flag = OK;
        int                             mx, my;

        void                    equalize(), real_equalize();

/* initialization */

        if (argc != 3 && argc != 5) {
                fprintf(stderr, "Usage: %s source(P6/P5 format) destination [skip_x skip_y] ...¥n", argv[0]);
                exit(NG);
        }

        if ((buffer = (unsigned char (*)[MAX_DELTA][MAX_DELTA])
                                                malloc(
                                                        (size_t)MAX_DELTA *
```

```
                                        (size_t)MAX_DELTA *
                                        (size_t)sizeof(unsigned char)
                                    )) == NULL) {

                                    fprintf(stderr, "Error:Cannot get memory for buffer!¥n");
                                    exit(MEM_ERR);
}

if ((original = (unsigned char (*)[MAX_POINTS][MAX_POINTS])
                                    malloc(
                                        (size_t)MAX_POINTS *
                                        (size_t)MAX_POINTS *
                                        (size_t)sizeof(unsigned char)
                                    )) == NULL) {

                                    fprintf(stderr, "Error:Cannot get memory for original!¥n");
                                    exit(MEM_ERR);
}

if ((area_x = (double (*)[MAX_FFT_NUMBER][MAX_FFT_NUMBER])
                                    malloc(
                                        (size_t)MAX_FFT_NUMBER *
                                        (size_t)MAX_FFT_NUMBER *
                                        (size_t)sizeof(double)
                                    )) == NULL) {

                                    fprintf(stderr, "Error:Cannot get memory for area_x!¥n");
                                    exit(MEM_ERR);
}

if ((area_y = (double (*)[MAX_FFT_NUMBER][MAX_FFT_NUMBER])
                                    malloc(
                                        (size_t)MAX_FFT_NUMBER *
                                        (size_t)MAX_FFT_NUMBER *
                                        (size_t)sizeof(double)
                                    )) == NULL) {

                                    fprintf(stderr, "Error:Cannot get memory for area_x!¥n");
                                    exit(MEM_ERR);
}

if ((mask = (unsigned char (*)[MAX_DELTA][MAX_DELTA])
                                    malloc(
                                        (size_t)MAX_DELTA *
                                        (size_t)MAX_DELTA *
                                        (size_t)sizeof(double)
                                    )) == NULL) {

                                    fprintf(stderr, "Error:Cannot get memory for mask!¥n");
                                    exit(MEM_ERR);
}

delta = DELTA;
```

```
gray_scale = GRAY_SCALE;

kind_of_eq = 1;

skip = SKIP_WEDGE;

smooth = 0;

normalize = 0;

offset_x = 0;
offset_y = 0;

strcpy(maskfile, MASK_FILE);

if (make_mask(maskfile, (*mask), delta) != OK) {
        fprintf(stderr, "@No mask file specified.¥n");
}

strcpy(source, argv[1]);
strcpy(destination, argv[2]);

if (argc == 5) {
        skip_x = abs(atoi(argv[3]));
        skip_y = abs(atoi(argv[4]));
}
else
        skip_x = skip_y = DELTA / 2;

fprintf(stderr, "@skip_X  = %d,¥tskip_Y  = %d.¥n", skip_x, skip_y);
fprintf(stderr, "@Source file      = %s ...¥n", source);
fprintf(stderr, "@Destination file = %s ...¥n", destination);

if (load_ppm(source, (*original), &len_x, &len_y) != OK) {

        fprintf(stderr, "Error:Cannot load data from source file(%s)!¥n", source);
        exit(DATA_ERR);
}
else
        fprintf(stderr, "Ok... Data have been loaded normally.¥n");


if ((fpo = fopen(destination, "a")) == NULL) {

        fprintf(stderr, "Error:Cannot open destination file(%s)!¥n", destination);
        exit(OPEN_ERR);
}

mx = my = 0;

for (base_y = offset_y; base_y + delta - 1 < len_y; base_y += skip_y)
        my++;

for (base_x = offset_x; base_x + delta - 1  < len_x; base_x += skip_x)
```

```
                mx++;

        if (fprintf(fpo, "%d %d¥n", mx, my) < 0) {
                fclose(fpo);
                fprintf(stderr, "Error:Cannot write header on destination(%s)!¥n",
                        mx, my, destination);
                exit(WRITE_ERR);
        }

/* main routine */

        serial = 0;

        for (base_y = offset_y; base_y + delta - 1 < len_y; base_y += skip_y) {
                for (base_x = offset_x; base_x + delta - 1  < len_x; base_x += skip_x) {

                        if (kind_of_eq != 0)
                                real_equalize((*original), (*buffer), base_x, base_y, delta, gray_scale);
                        else
                                equalize((*original), (*buffer), base_x, base_y, delta, gray_scale);

                        FFT2D((*buffer), delta, (*area_x), (*area_y), (*mask));

                        adjust_matrix((*area_x), (*area_y), delta);

                                if (save_index(fpo, (*area_x), (*area_y), delta) != OK) {
                                fprintf(stderr, "Error:Cannot write results on destination file(%s) !¥n", destination);
                                exit(WRITE_ERR);
                        }

                        if (serial % 100 == 0) {
                                fprintf(stderr, ".");
                                fflush(stderr);
                        }

                        serial++;

                }
        }

        fclose(fpo);

        fprintf(stderr, "¥nOk... Calculation finished(%s > %s). Total squares = %d.¥n",
                                                        source, destination, serial);

        exit(OK);
}
make_mask(maskfile, mask, delta)
char    maskfile[];
unsigned char   mask[][MAX_DELTA];
int             delta;
{
        FILE    *fpi;
        int             c, x, y;
```

```
        if ((fpi = fopen(maskfile, "r")) == NULL) {
                for (y = 0; y < delta; y++)
                        for (x =  0; x < delta; x++)
                                mask[x][y] = 255;

                return OPEN_ERR;

        }
        else {
                for (y = 0; y < delta; y++)
                        for (x =  0; x < delta; x++)
                                if ((c = fgetc(fpi)) == EOF) {
                                        fprintf(stderr, "Error:In sufficient data(%s)!¥n", maskfile);
                                        fclose(fpi);
                                        return DATA_ERR;
                                }
                                else
                                        mask[x][y] = c;

                fclose(fpi);
                return OK;
        }

}
set_name(fpi, source, destination)
FILE    *fpi;
char    source[], destination[];
{
        if (fscanf(fpi, "%s %s", source, destination) == EOF)
                return EOF;
        else
                return OK;
}
load_ppm(source, data, len_x, len_y)
char                    source[];
unsigned char   data[][MAX_POINTS];
int                             *len_x, *len_y;
{
        FILE    *fpi;
        char    sign[100], lx[100], ly[100], bright[100];
        int             x, y, r, g, b;
        int             format;

        if ((fpi = fopen(source, "r")) == NULL) {

                fprintf(stderr, "Error:Cannot open %s!¥n", source);
                return OPEN_ERR;
        }

        if (fgets(sign, 98, fpi) == NULL) {
                fprintf(stderr, "Error:Cannot load header of %s!¥n", source);
                fclose(fpi);
                return DATA_ERR;
```

```
}

if (fgets(bright, 98, fpi) == NULL) {
        fprintf(stderr, "Error:Cannot load header of %s!¥n", source);
        fclose(fpi);
        return DATA_ERR;
}

sscanf(bright, "%s %s", lx, ly);

if (fgets(bright, 98, fpi) == NULL) {
        fprintf(stderr, "Error:Cannot load header of %s!¥n", source);
        fclose(fpi);
        return DATA_ERR;
}

fclose(fpi);

if (strcmp(sign, "P6¥n") == 0)                    /* ppm */
        format = 6;
else if (strcmp(sign, "P5¥n") == 0)  /* pgm */
        format = 5;
else {
        fprintf(stderr, "Error:%s is not ppm-file(P6/P5)!¥n", source);
        return DATA_ERR;
}

*len_x = abs(atoi(lx));
*len_y = abs(atoi(ly));

fprintf(stderr,"@Data length: x = %d, y = %d, Max brightness = %s",
                *len_x, *len_y, bright);


if ((fpi = fopen(source, "r")) == NULL) {

        fprintf(stderr, "Error:Cannot open %s!¥n", source);
        return OPEN_ERR;
}

r = 3;
while (r > 0) {
        if ((g = fgetc(fpi)) == EOF) {
                fprintf(stderr, "Error:Cannot skip header(%s)!¥n", source);
                fclose(fpi);
                return DATA_ERR;
        }
        if (g == '¥n')
                r--;
}


if (format == 6) {
        for (y = 0; y < *len_y; y++)
```

```
                        for (x = 0; x < *len_x; x++) {

                                if ((r = fgetc(fpi)) == EOF ||
                                        (g = fgetc(fpi)) == EOF ||
                                        (b = fgetc(fpi)) == EOF)
                                        return DATA_ERR;

                                data[x][y] = (unsigned char)g;

                        }
        }
        else    {
                for (y = 0; y < *len_y; y++)
                        for (x = 0; x < *len_x; x++) {

                                if ((g = fgetc(fpi)) == EOF)
                                        return DATA_ERR;

                                data[x][y] = (unsigned char)g;

                        }
        }

        fclose(fpi);

        return OK;
}
FFT2D(data, delta, area_x, area_y, mask)
unsigned char   data[][MAX_DELTA];
int                             delta;
double                  area_x[][MAX_FFT_NUMBER];
double                  area_y[][MAX_FFT_NUMBER];
unsigned char   mask[][MAX_DELTA];
{

        int             x, y;
        double  buffer_x[MAX_FFT_NUMBER], buffer_y[MAX_FFT_NUMBER];

/* initialization */

        for (y = 0; y < delta; y++) {

                for (x = 0; x < delta; x++) {
                        buffer_x[x] = data[x][y] * (double)mask[x][y] / 255.0;
                        buffer_y[x] = 0;
                }

                FFT(buffer_x, buffer_y, delta, NORMAL);

                for (x = 0; x < delta; x++) {
                        area_x[x][y] = buffer_x[x];
                        area_y[x][y] = buffer_y[x];
                }
```

```
        }

        for (x = 0; x < delta; x++) {

                for (y = 0; y < delta; y++) {
                        buffer_x[y] = area_x[x][y];
                        buffer_y[y] = area_y[x][y];
                }

                FFT(buffer_x, buffer_y, delta, NORMAL);

                for (y = 0; y < delta; y++) {
                        area_x[x][y] = buffer_x[y];
                        area_y[x][y] = buffer_y[y];
                }

        }


}
average(data, x, y, len_x, len_y, pixel_size)
unsigned char   data[][MAX_POINTS];
int                             x, y, pixel_size, len_x, len_y;
{
        double          sum = 0;
        double          points = 0;
        int                     xx, yy, av;

        if (pixel_size <= 1)
                return data[x][y];

        for (yy = y; yy < y + pixel_size; yy++)
                for (xx = x; xx < x + pixel_size; xx++) {

                        if (xx < 0 || xx >= len_x || yy < 0 || yy >= len_y)
                                continue;
                        else {
                                sum += data[xx][yy];
                                points++;
                        }
                }

        if (points > 0)
                av = (int)(sum / points);
        else
                av = 0;

        return av;
}
chop(value, lower, upper)
int             value, lower, upper;
{
        if (value < lower)
                return lower;
```

```
        else if (value > upper)
                return upper;
        else
                return value;
}
void equalize(org, data, base_x, base_y, delta, gs)
unsigned char   org[][MAX_POINTS];
unsigned char   data[][MAX_DELTA];
int                             base_x, base_y, delta, gs;
{
        int             x, y, c;
        double  v;
        int             min = GRAY_SCALE - 1, max = 0;

        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {

                        c = org[x + base_x][y + base_y];
                        if (c < min)
                                min = c;
                        if (c > max)
                                max = c;
                }
        }

        if (min == max)
                return;

        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {

                        v = (double)(org[x + base_x][y + base_y] - min) / (double)(max - min + 1) * (double)gs;
                        if (v >= gs)
                                data[x][y] = gs - 1;
                        else
                                data[x][y] = v;
                }
        }


        return;
}
void real_equalize(org, data, base_x, base_y, delta, gs)
unsigned char   org[][MAX_POINTS];
unsigned char   data[][MAX_DELTA];
int                             base_x, base_y, delta, gs;
{
        int             x, y, i, a, l, v;
        int             histgram[GRAY_SCALE];
        int             table[GRAY_SCALE];
        double  d;

        for (i = 0; i < GRAY_SCALE; i++)
                histgram[i] = table[i] = 0;
```

```
        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {

                        v = org[x + base_x][y + base_y];
                        i = chop(v, 0, GRAY_SCALE - 1);
                        histgram[i]++;

                }
        }

        d = (double)delta * (double)delta / (double)gs;

        a = l = 0;

        for (i = 0; i < GRAY_SCALE; i++) {

                a += histgram[i];

                if (a >= d) {
                        a = 0;
                        l++;
                }

                chop(l, 0, gs - 1);

                table[i] = l;

        }

        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {

                v = org[x + base_x][y + base_y];
                data[x][y] = (int)((double)table[v] * (double)gs / (double)(l + 1));

                }
        }


        return;
}
fprint_matrix(area_x, area_y, delta)
double  area_x[][MAX_FFT_NUMBER];
double  area_y[][MAX_FFT_NUMBER];
int             delta;
{
        FILE    *fp;
        int             x, y;

        fp = fopen("testfft_x", "w");

        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {
```

```
                        fprintf(fp, "%4.2f¥t", area_x[x][y]);

                }
                fprintf(fp, "¥n");
        }

        fclose(fp);

        fp = fopen("testfft_y", "w");

        for (y = 0; y < delta; y++) {
                for (x = 0; x < delta; x++) {

                        fprintf(fp, "%4.2f¥t", area_y[x][y]);

                }
                fprintf(fp, "¥n");
        }

        fclose(fp);
}
adjust_matrix(ax, ay, delta)
double  ax[][MAX_FFT_NUMBER];
double  ay[][MAX_FFT_NUMBER];
int             delta;
{
        int             center;
        int             x, y, px, py;

        center = delta / 2;

        for (y = 0; y < center; y++)
                for (x = 0; x < center; x++) {

                        px = x + center;
                        py = y + center;

                        swap(&ax[x][y], &ax[px][py]);
                        swap(&ay[x][y], &ay[px][py]);

                        swap(&ax[px][y], &ax[x][py]);
                        swap(&ay[px][y], &ay[x][py]);


                }

}
swap(x, y)
double  *x, *y;
{
        double  temp;

        temp = *y;
```

```
        *y = *x;
        *x = temp;
}
save_index(fpo, ax, ay, delta)
FILE    *fpo;
double  ax[][MAX_FFT_NUMBER];
double  ay[][MAX_FFT_NUMBER];
int             delta;
{
        double  pp;
        double  r;
        double  total, w_total;
        double  ring[MAX_FFT_NUMBER], wedge[MAX_FFT_NUMBER],
                        N_ring[MAX_FFT_NUMBER], N_wedge[MAX_FFT_NUMBER];
        int             x, y, center, px, py, wedge_number, ring_number;
        int             total_number;
        void    smooth_array();

        center = total_number = delta / 2;

        for (x = 0; x < total_number; x++)
                ring[x] = N_ring[x] = wedge[x] = N_wedge[x] = 0;

        for (y = 0; y < delta; y++)
                for (x = center; x < delta; x++) {
                        pp = sqrt((double)(ax[x][y] * ax[x][y] + ay[x][y] * ay[x][y]));

                        px = x - center;
                        py = y - center;

                        /* RINGS */

                        r = sqrt(px * px + py * py);
                        ring_number = chop((int)r, 0, (center - 1));

                        ring[ring_number] += pp;

                        /* WDGE */

                        if (r < skip)
                                continue;

                        if (px == 0) {
                                if (py > 0)
                                        wedge_number = center - 1;
                                else if (py <= 0)
                                        wedge_number = 0;
                        }
                        else {
                                wedge_number = (int)((atan((double)py / (double)px) + PI / 2.0) / PI * center);
                                wedge_number = chop(wedge_number, 0, center - 1);
                        }
```

```
                        wedge[wedge_number] += pp;
                }

        smooth_array(ring, smooth, delta);
        smooth_array(wedge, smooth, delta);

        total = w_total = 0;

        for (x = 0; x < total_number; x++) {
                total += ring[x];
                w_total += wedge[x];
        }

        if (total != 0)
                for (x = 0; x < total_number; x++)
                        N_ring[x] = ring[x] / total;

        if (w_total != 0)
                for (x = 0; x < total_number; x++)
                        N_wedge[x] = wedge[x] / w_total;

/*
        depth = calcdepth(N_ring, delta);

        slant = calcslant(N_wedge, delta);

        slantpeak = searchpeak(N_wedge, delta);

        if (fprintf(fpo, "%f¥t", depth) <= 0)
                return WRITE_ERR;

        if (fprintf(fpo, "%f¥t", slant) <= 0)
                return WRITE_ERR;

        if (fprintf(fpo, "%f¥t", slantpeak) <= 0)
                return WRITE_ERR;
*/

        if (normalize == ON) {
                for (x = 0; x < total_number; x++)
                        if (fprintf(fpo, "%f¥t", N_ring[x]) <= 0)
                                return WRITE_ERR;

                for (x = 0; x < total_number; x++)
                        if (fprintf(fpo, "%f¥t", N_wedge[x]) <= 0)
                                return WRITE_ERR;
        }
        else {
                for (x = 0; x < total_number; x++)
                        if (fprintf(fpo, "%f¥t", ring[x]) <= 0)
                                return WRITE_ERR;

                for (x = 0; x < total_number; x++)
                        if (fprintf(fpo, "%f¥t", wedge[x]) <= 0)
```

```
                return WRITE_ERR;
        }

        if (fprintf(fpo, "¥n") <= 0)
                return WRITE_ERR;

        return OK;

}
void smooth_array(N, span, delta)
double  N[];
int             span, delta;
{

        double  N2[MAX_FFT_NUMBER];
        double  v, sum;
        int             i, j, k, l, m;

        if (span == 0)
                return;

        m = delta / 2;

        for (i = 0; i < m; i++)
                N2[i] = N[i];

        for (i = 0; i < m; i++) {
                sum = 0;
                l = 0;
                for (j = -span; j <= span; j++) {

                        if ((k = i + j) < 0 || k >= m)
                                v = 0;
                        else {
                                v = N2[k];
                                l++;
                        }

                        sum += v;
                }

                if (l == 0)
                        N[i] = 0;
                else
                        N[i] = sum / l;

        }
}
```

（2）fourierbatch.exe
（3）fourierpipe.exe
（4）fourierbatchpipe.exe

以上のモジュールは（1）のパラメータ受け渡し方法を変更したものであるため、プログラムソースは省略する。

b）テクスチャ特徴量計算および２次元マップ化・１次元プロファイル化に関するモジュール
（１）fouriermap.exe

本モジュールは、「fft3d.c」（既出）、「fouriermap.c」で構成される。葉傾斜角の計測に直接関連しないため、ここではプログラムソースは省略する。

（２）fouriermappipe.exe

本モジュールは（１）のパラメータ受け渡し方法を変更したものであるため、プログラムソースは省略する。

c）葉傾斜角の３次元分布計算に関するモジュール

（１）3dscan.exe

本モジュールは、「3dscan3.c」で構成される。

3dscan3.c　：　葉傾斜角の２次元分布から３次元分布を再構成する。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>

#define		MAX_POINTS		64
#define		MAX_PAGES		8
#define		GRAY_SCALE		256
#define		DELTA			128
#define		INDEX			64

#define		STEP_ANGLE		5

#define		NG				-1
#define		OK				0
#define		OPEN_ERR		1
#define		DATA_ERR		2
#define		WRITE_ERR		3
#define		MEM_ERR			4

#define		ON				1
#define		OFF				0

#define		PI				3.1415927

#define		SKIP_ANGLE		40 / 180 * PI
#define		ITERATION		10

int			gray_scale;

/* USAGE: 3dscan.exe source.file(piled) 3d.file */

int main(int argc, char *argv[])
{
	char		source[100], destination[100];
	float		(*buffer3D)[MAX_POINTS][MAX_POINTS][INDEX];
	float		(*buffer2D)[MAX_PAGES][MAX_POINTS][MAX_POINTS][INDEX];
	double		skip_angle = SKIP_ANGLE, step_angle;
	int			iteration = ITERATION;
	int			len_x, len_y;
	int			start_z, end_z;
	int			delta;
	int			x, y, z, i, page;
	int			max;

/* initialization */
```

```
        if (argc != 3) {
                fprintf(stderr, "Usage: %s source.file destination(P2 format).file...¥n", argv[0]);
                exit(NG);
        }

        strcpy(source,argv[1]);
        strcpy(destination, argv[2]);

        if ((buffer3D = (float (*)[MAX_POINTS][MAX_POINTS][INDEX])
                                                malloc(
                                                        (size_t)MAX_POINTS *
                                                        (size_t)MAX_POINTS *
                                                        (size_t)INDEX *
                                                        (size_t)sizeof(float)
                                                )) == NULL) {

                                                fprintf(stderr, "Error:Cannot get memory for buffer(3D)!¥n");
                                                exit(MEM_ERR);
        }

        if ((buffer2D = (float (*)[MAX_PAGES][MAX_POINTS][MAX_POINTS][INDEX])
                                                malloc(
                                                        (size_t)MAX_PAGES *
                                                        (size_t)MAX_POINTS *
                                                        (size_t)MAX_POINTS *
                                                        (size_t)INDEX *
                                                        (size_t)sizeof(float)
                                                )) == NULL) {

                                                fprintf(stderr, "Error:Cannot get memory for buffer(2D)!¥n");
                                                exit(MEM_ERR);
        }

        delta = DELTA;

        iteration = ITERATION;

        step_angle = STEP_ANGLE / 180.0 * PI;

        max = 0;            /* 0 for accumulate, 1 for maxmize */

        for (page = 0; page < MAX_PAGES; page++)        /* clear buffers */

                for (x = 0; x < MAX_POINTS; x++)
                        for (y = 0; y < MAX_POINTS; y++)
                                for (z = 0; z < INDEX; z++)
                                        (*buffer2D)[page][x][y][z] = 0;

/* end of initialization */

/* Main routine */

        if (load_data(source, (*buffer2D), &len_x, &len_y, delta, &page) != OK) {
```

```
                fprintf(stderr, "Error:Cannot 2D data from %s!¥n", source);
                exit(NG);

        }
        else {
                fprintf(stderr, "@Total page number = %d. ", page);
                skip_angle = 2 * PI / page;
                fprintf(stderr, "@Skip angle = %f (degree).¥n", (double)(360 / page));
        }

        if (make_P2_header(destination, len_x, len_y) != OK) {
                fprintf(stderr, "Error:Cannot write header on destination file(%s)!¥n", destination);
                exit(NG);
        }

        /* iterated calculations */

        start_z = 0;

        end_z = len_y - 1;

        for (z = start_z; z <= end_z; z++) {

                fprintf(stderr, "<%3d>", z);

                for (i = 0; i < iteration; i++) {

                        /*      fprintf(stderr, "<%3d>", i); */

                        /* MAIN CALC. */
                        scan((*buffer2D), (*buffer3D), z, len_x, len_y, delta, page, skip_angle, step_angle, max);

                }

                if (save_result(destination, (*buffer3D), len_x, len_y, delta) != OK) {

                        fprintf(stderr, "¥nError:Cannot save result on %s!¥n", destination);
                        exit(NG);

                }
        }

        fprintf(stderr, "¥n");


        fprintf(stderr, "¥n@All finished!¥n");
        exit(OK);
}
make_P2_header(destination, len_x, len_y)
char    destination[];
int             len_x, len_y;
{
        FILE    *fpo;
```

```
        long    height;

        height = (long)len_x * (long)len_y;

        if ((fpo = fopen(destination, "a")) == NULL)
                return OPEN_ERR;

        if (fprintf(fpo, "P2¥n%d %d¥n255¥n", len_x, (long)height) < 0) {
                fclose(fpo);
                return WRITE_ERR;
        }

        fclose(fpo);

        return OK;

}
scan(buffer2D, buffer3D, z, len_x, len_y, delta, page, skip_angle, step_angle, max)
float           buffer2D[MAX_PAGES][MAX_POINTS][MAX_POINTS][INDEX];
float           buffer3D[MAX_POINTS][MAX_POINTS][INDEX];
int                     z, len_x, len_y, delta, page, max;
double          skip_angle, step_angle;
{
        int             x, y, p;
        int             abs_x, abs_y;
        int             center;
        int             screen, index;
        double  base_angle, scan_angle, angle;
        double  get_angle();
        double  r, value1;

        for (x = 0; x < len_x; x++)
                for (y = 0; y < len_y; y++)
                        for (index = 0; index < delta / 2; index++)
                                buffer3D[x][y][index] = 0;

        center = len_x / 2;


        for (y = 0; y < len_y; y++)
                for (x = 0; x < len_x; x++) {

                        abs_x = x - center;
                        abs_y = y - center;
                        base_angle = get_angle(abs_x, abs_y);
                        r = sqrt(abs_x * abs_x + abs_y * abs_y);

                        for (p = 0, scan_angle = 0; p < page; p++, scan_angle += skip_angle) {

                                for(angle = 0; angle < skip_angle; angle += step_angle) {

                                        screen = chop((int)(center + r * sin(base_angle - (scan_angle + angle))), 0,
(len_x - 1));
```

```
                                        for (index = 0; index < delta / 2; index++) {

                                                if (max == 0)
                                                        buffer3D[x][y][index] += buffer2D[p][screen][z][index];
                                                else {
                                                        if ((value1 = buffer2D[p][screen][z][index]) >
buffer3D[x][y][index])
                                                                buffer3D[x][y][index] = value1;
                                                }

                                        }

                                }
                        }

                }

        return OK;
}
double  get_angle(x, y)
int             x, y;
{
        double  angle;

        if (x == 0) {

                if (y == 0)
                        angle = 0;
                else if (y > 0)
                        angle = PI / 2.0;
                else
                        angle = - PI / 2.0;

                return angle;
        }

        angle = atan((double)y / (double)x);

        if (x < 0)
                angle += PI;

        return angle;
}
load_data(source, buffer2D, len_x, len_y, delta, page)
char            source[];
float           buffer2D[MAX_PAGES][MAX_POINTS][MAX_POINTS][INDEX];
int                     *len_x, *len_y, delta, *page;
{
        FILE            *fpi;
        int             x, y, z, flag = OK;
        int             lx, ly;
        double  v;

        *page = 0;
```

```
if ((fpi = fopen(source, "r")) == NULL) {
        fprintf(stderr, "Error:Cannot open source file(%s)!¥n", source);
        return OPEN_ERR;
}

while (*page < MAX_PAGES) {

        if (fscanf(fpi, "%d %d", &lx, &ly) < 0) {
                return DATA_ERR;
        }

        *len_x = lx;
        *len_y = ly;

        if (lx <= 0 ¦¦ lx > MAX_POINTS ¦¦ ly <= 0 ¦¦ ly > MAX_POINTS) {

                fprintf(stderr, "Error:Page sizes(%d, %d) exceed limits!¥n", lx, ly);
                fclose(fpi);
                return NG;
        }


        for (y = 0; y < ly; y++) {
                for (x = 0; x < lx; x++) {

                        for (z = 0; z < delta / 2; z++)
                                if (fscanf(fpi, "%lf", &v) == EOF) {
                                        flag = EOF;
                                        break;
                                }

                        if (flag != OK)
                                break;

                        for (z = 0; z < delta / 2; z++)
                                if (fscanf(fpi, "%lf", &v) == EOF) {
                                        flag = EOF;
                                        break;
                                }
                                else
                                        buffer2D[*page][x][y][z] = v;

                        if (flag != OK)
                                break;
                }
                if (flag != OK)
                        break;
        }

        if (flag != OK)
                break;

        (*page)++;
```

```
        }

        fclose(fpi);

        return OK;
}
chop(value, lower, upper)
int             value, lower, upper;
{
        if (value < lower)
                return lower;
        else if (value > upper)
                return upper;
        else
                return value;
}
swap(x, y)
double  *x, *y;
{
        double  temp;

        temp = *y;

        *y = *x;
        *x = temp;
}
save_result(destination, buffer3D, len_x, len_y, delta)
char    destination[];
float   buffer3D[MAX_POINTS][MAX_POINTS][INDEX];
int             len_x, len_y, delta;
{
        FILE    *fpo;
        int             x, y;
        int             flag = OK;
        double  slant, sum, average, square, absslant(), sumindex();
        double  max_slant, min_slant, min_depth, number;

        max_slant = -DBL_MAX;
        min_slant = DBL_MAX;

        if ((fpo = fopen(destination, "a")) == NULL) {

                fprintf(stderr, "Error:Cannot open %s!¥n", destination);
                return OPEN_ERR;
        }

        average = square = number = 0;

        for (y = 0; y < len_y; y++)             {
                for (x = 0; x < len_x; x++) {

                        slant = absslant(buffer3D, x, y, delta);
```

```
                    if (slant > max_slant)
                            max_slant = slant;
                    else if (slant < min_slant)
                            min_slant = slant;

                    sum = sumindex(buffer3D, x, y, delta);

                    average += sum;
                    square += sum * sum;
                    number++;
            }
}

if (number > 0)
            average /= number;
else
            average = 0;

if (number <= 1)
            min_depth = 0;
else
            min_depth = average - sqrt((square - number * average * average) / (number - 1));

for (y = 0; y < len_y; y++)             {
            for (x = 0; x < len_x; x++) {

                    slant = absslant(buffer3D, x, y, delta);
                    sum = sumindex(buffer3D, x, y, delta);

                    if (max_slant == min_slant || sum < min_depth)
                            slant = 0;
                    else  {
                            slant = (slant - min_slant) / (max_slant - min_slant) * (GRAY_SCALE - 2) + 1;

                            if (slant < 1)
                                    slant = 1;
                            else if (slant > GRAY_SCALE -1)
                                    slant = GRAY_SCALE - 1;
                    }

                    if (fprintf(fpo, "%d¥t", (int)slant) == EOF) {
                            flag = WRITE_ERR;
                            break;
                    }

            }

            if (fprintf(fpo, "¥n") == EOF) {
                            flag = WRITE_ERR;
            }

            if (flag != OK)
                    break;
```

```
        }

        fclose(fpo);

        return flag;

}
double  absslant(buffer, x, y, delta)
float   buffer[MAX_POINTS][MAX_POINTS][INDEX];
int             x, y, delta;
{
        double  total = 0, sum = 0;
        float   index [INDEX];
        int             i;

                for (i = 0; i < delta / 2; i++)
                        index[i] = buffer[x][y][i];

                for (i = 0; i < delta / 4; i++) {
                                total += index[i];
                }


                for (i = 0; i < delta / 4; i++) {
                                total += index[delta / 4 + i];
                                index[i] += index[delta / 2 - i - 1];
                }


                if (total != 0) {
                        for (i = 0; i < delta / 4; i++)
                                index[i] /= total;
                }

                sum = 0;

                for (i = 0; i < delta / 4; i++) {
                        sum += index[i] * i ;
                }

                return sum;
}
double  sumindex(buffer, x, y, delta)
float   buffer[MAX_POINTS][MAX_POINTS][INDEX];
int             x, y, delta;
{
        double  sum = 0;
        int             i;

                for (i = 0; i < delta / 2; i++)
                        sum += buffer[x][y][i];

                return sum;
}
```

```
insert_zeros(fpo, len_x)
FILE    *fpo;
int             len_x;
{
        int             flag = OK, i;

        for (i = 0; i < len_x; i++) {

                if (fprintf(fpo, "%f¥t", 0.0) == EOF) {
                        flag = WRITE_ERR;
                        break;
                }
        }

        if (fprintf(fpo, "¥n") == EOF)
                flag = WRITE_ERR;

        return flag;

}
reverse(s)
char    s[];
{
        int     c, i, j;

        for (i = 0, j = strlen(s) - 1; i < j; i++, j--) {

                c = s[i];
                s[i] = s[j];
                s[j] = c;
        }
}
itoa(n, s)
int             n;
char    s[];
{
        int             i, sign;

        if ((sign = n) < 0)
                n = -n;

        i = 0;

        do {

                s[i++] = n % 10 + '0';

        } while (( n /= 10) > 0);

        if (sign < 0)
                s[i++] = '-';

        s[i] = '¥0';
```

```
    reverse(s);
}
```

（2）3dscanpipe.exe

本モジュールは（1）のパラメータ受け渡し方法を変更したものであるため、プログラムソースは省略する。

d）葉傾斜角の3次元分布の1次元プロファイル化に関するモジュール
（1）ringpipe.exe

本モジュールは「ringpipe2.c」で構成される。

ringpipe2.c　：　葉傾斜角の3次元分布の1次元プロファイルを計算する。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>

#define		OK			0
#define		NG			1
#define		BREAK		-1
#define		CONTINUE	0

#define		MAP_SIZE	128
#define		RING		3
#define		HEIGHT			128

#define		PI			3.1415927

/* Usage: cat 3d.map(P2 format) ¦ ringpipe.exe > ringprofile.dat */

int main(int argc, char *argv[])
{
	int			max_ring;
	double		(*map)[MAP_SIZE][MAP_SIZE];

	if (argc != 1) {
		fprintf(stderr, "Usage: cat 3d.map(P2 format) ¦ %s > ringprofile.dat ...¥n", argv[0]);
		exit(NG);
	}

	if ((map = (double (*)[MAP_SIZE][MAP_SIZE])
				malloc((size_t)MAP_SIZE *
					(size_t)MAP_SIZE *
					(size_t)(sizeof(double)))) == NULL) {

		fprintf(stderr, "Error:Cannot get memory for map[%d][%d]!¥n", MAP_SIZE, MAP_SIZE);
		exit(NG);
	}

	max_ring = RING;

	if (make_profile(max_ring, (*map)) != OK) {
		fprintf(stderr, "Error:Couldn't calculate profile!¥n");
		exit(NG);
	}
	else
		fprintf(stderr, "@Finshed!¥n");
```

```
        exit(OK);
}
load_header(format, width, height)
int             *format, *width, *height;
{
        char    line1[100], line2[100], line3[100];
        int             lx, ly;

        if (fgets(line1, 98, stdin) == NULL) {
                fprintf(stderr, "Error:Cannot load header!¥n");
                return NG;
        }

        if (fgets(line2, 98, stdin) == NULL) {
                fprintf(stderr, "Error:Cannot load header!¥n");
                return NG;
        }

        sscanf(line2, "%d %d", &lx, &ly);

        if (fgets(line3, 98, stdin) == NULL) {
                fprintf(stderr, "Error:Cannot load header!¥n");
                return NG;
        }


        if (strcmp(line1, "P2¥n") == 0)                 /* pgm ascii */
                *format = 2;
        else if (strcmp(line1, "P5¥n") == 0)        /* pgm binary */
                *format = 5;
        else {
                fprintf(stderr, "Error:Not pgm-file(P2/P5)!¥n");
                return NG;
        }

        *width = lx;
        if (lx > 0)
                *height = ly / lx;
        else {
                fprintf(stderr, "Error:Illegal data file!¥n");
                return NG;
        }


        fprintf(stderr,"@Format = %d, Map size: x = %d, y = %d, Depth: %d ...¥n",
                        *format, *width, *width, *height);

        return OK;
}
load_map(format, width, map)
int             format, width;
double  map[][MAP_SIZE];
{
        int     x, y, value;
```

```
        if (format == 2) {
                for (y = 0; y < width; y++)
                        for (x = 0; x < width; x++) {

                                if (fscanf(stdin, "%d", &value) < 0)
                                        return EOF;

                                map[x][y] = value;

                        }
        }
        else    {
                for (y = 0; y < width; y++)
                        for (x = 0; x < width; x++) {

                                if ((value = fgetc(stdin)) == EOF)
                                        return EOF;

                                map[x][y] = value;

                        }
        }

        return OK;
}
make_profile(max_ring, map)
int             max_ring;
double  map[MAP_SIZE][MAP_SIZE];
{
        int             z, flag, i, j, ring;
        int             format, width, height;
        double  cx, cy, area, sum, max, deltaR, pr[HEIGHT], r;
        double  sum_ring[RING][HEIGHT], area_ring[RING][HEIGHT];
        double  average, square, stdev;

        if (load_header(&format, &width, &height) != OK)
                return NG;

        if (height > HEIGHT) {
                fprintf(stderr, "Error:Height(%d) exceeds limit(%d)!¥n", height, HEIGHT);
                return NG;
        }

        printf("3dprofile¥t%d¥t", height);

        for (z = 0; z < height; z++) {

                flag = CONTINUE;
                sum = 0;
                max = -DBL_MAX;

                cx = cy = sum = area = 0;
```

```
        if (load_map(format, width, map) == EOF)
                return OK;

        for (j = 0; j < width; j++)
                for (i = 0; i < width; i++) {
                        cx += i * fabs(map[i][j] != 0);
                        cy += j * fabs(map[i][j] != 0);
                        if (map[i][j] != 0)
                                area++;
                }

        if (area != 0) {
                cx /= area;
                cy /= area;
                pr[z] = sqrt(area / PI);
        }

        for (i = 0; i < RING; i++)
                sum_ring[i][z] = area_ring[i][z] = 0;

        deltaR = pr[z] / max_ring;

        fprintf(stderr, "@cx = %d, cy = %d, pr = %f, deltaR = %f ...¥n",
                        (int)cx, (int)cy, pr[z], deltaR);

        for (j = 0; j < width; j++)
                for (i = 0; i < width; i++) {

                        r = sqrt((j - cy) * (j - cy) + (i - cx) * (i - cx));

                        ring = (int)(r / deltaR);

                        if (ring >= max_ring || map[i][j] == 0)
                                continue;

                        sum_ring[ring][z] += map[i][j];
                        (area_ring[ring][z])++;

                }

        for (i = 0; i < max_ring; i++)
                if (area_ring[i][z] != 0)
                        sum_ring[i][z] /= area_ring[i][z];


}

/*              printf("%d¥t%d¥t%10.3f¥t%10.3f¥n", (int)cx, (int)cy, pr, deltaR); */

for (i = 0; i < max_ring; i++) {

        average = square = 0;

        for (z = 0; z < height; z++) {
```

```
                    average += sum_ring[i][z];
                    square += sum_ring[i][z] * sum_ring[i][z];

            }

            if (height != 0)
                    average /= height;

            if (height <= 1)
                    stdev = 0;
            else
                    stdev = sqrt((square - height * average * average) / (height - 1));

            printf("%f %f¥t", average, stdev);

            for (z = 0; z < height; z++)
                    printf("%f ", sum_ring[i][z]);

            printf("¥t");
    }

    average = square = 0;

    for (z = 0; z < height; z++) {
            average += pr[z];
            square += pr[z] * pr[z];

    }

    if (height != 0)
            average /= height;

    if (height <= 1)
            stdev = 0;
    else
            stdev = sqrt((square - height * average * average) / (height - 1));

    printf("%f %f¥t", average, stdev);

    for (z = 0; z < height; z++)

            printf("%f ", pr[z]);

    printf("¥n");

    return OK;
}
```

3．走査型２次元分光分析計の開発

3．走査型２次元分光分析計の開発

3．1．はじめに

農作業は筋肉労働だけではなく、人間の五感による観察・生育診断が極めて重要な作業となる。このため、人間が圃場を巡回する代わりに、五感のような機能を持つセンサー類を置き、コンピュータがそれから得た情報をデータを用いて様々な生育状況の判断を行うことが可能になれば、従来修得に長い年月と経験が必要とされた篤農家的専門技術は不要となり、結果として作物の生産性向上に大きな寄与をもたらすと期待される。作物の生育状態を反映する情報の中でも、人間の目を通して得られる視覚情報の重要性は極めて高い。視覚情報は主として色と形に関する情報であるが、とりわけ色情報から判断できることは、栄養状態の適否、収穫部位の充実度合い、病虫害や環境ストレスなどと極めて多岐に渡り、上記のような遠隔生育診断を行う上で極めて重要な情報であると言えよう。

ここで、色情報を用いた遠隔的生育診断を行う上で、色情報をいかにして正確に計測し、遠くへ伝達し、さらには再現するかという問題がある。なぜなら、人間が実際の作物を直接観察して認識する色は、現状で最も一般的な写真やビデオ機材等の機材により計測・再現される色とは決して同一ではあり得ないからである。すなわち、カメラやフィルムスキャナは、色を赤（R）・緑（G）・青（B）の３原色に分けて計測し、逆にカラーフィルムやブラウン管でその混合率に基づいて再現している。このようにして観測・再現された色調はあくまでも実際の色の近似色にすぎず、真の色と同じである保証は全くない。当然、色情報に含まれる植物の生育情報の多くの部分は観測・再現の段階でそぎ落とされる可能性が高い。

作物の色情報をあるがままに計測・再現するためには、色を連続した分光成分に分解・記録し、さらにこれらから逆に色情報に再現する手段を新たに用意しなければならない。そこで本研究では、まず作物の色を正確かつ定量的に計測することを目的とし、プリズムを用いた走査型２次元分光分析計を開発し、その有効性を検討した。本器の最大の特徴は、受光素子に市販の２次元白黒ＣＣＤカメラを用いることにより、１次元走査だけで作物の２次元分光分析を可能とした点にある。また、対物レンズには任意の焦点距離のレンズを交換・使用可能なため、対象物は、幼苗固体などのミクロなものから、群落さらには圃場全体など、マクロなものまで計測可能となっている。

3．2．走査型２次元分光分析計の構造

今回開発した走査型２次元分光分析計の概略を図１に示す。対物レンズ（A）から入った光はスリット上に結像したのち、コリメータ（B）により平行光線とされ、プリズムへ偏角51°24′で入射する。プリズムにより分光された光線は対物レンズ（C）によりＣＣＤカメラの２次元受光素子上に分光イメージとして結像される（図２）。スリットの幅は30μｍである。レンズA・B・Cはいずれも一眼レフカメラ用レンズであり、焦点距離はレンズAとCが３７ｍｍ、レンズBが５０ｍｍである。プリズムは重フリントガラス（ＳＦ２）製であり、頂角は60°となっている。

本体は主に合板（厚さ9mm）をネジで留めて作り、乱反射などによる迷光を防ぐために内側に黒い紙を貼り、さらにコリメーターとプリズムの間には黒のフェルトで覆った。

3．3．結果

以下の作物を対象に分光計測を実行し、本器の有効性を検討した。

1）ポトス：サトイモ科（*Scindapsus aureus*）

観葉植物の一種であり、白または黄色の斑が入る。ここでは白斑部分を切り取り、白斑と緑部の色調の差異を確かめ、次いで一枚の葉の全体を２次元走査撮影した。

a）１次元分光計測

斑の大きく入った葉を選び、白斑と緑部に切り分けた。葉片を色別に、白パネルに貼り、１次元分光した結果には各部位の色情報の差が明確である。（図３）

b）２次元分光計測

葉を白パネルに固定し、本器を５ｍｍ毎に水平１次元走査し、18枚の分光画像を得た。18枚の分光画像から、300～800nmまで20nmごとの分光画像を得た。白斑は１次元分光計測による計測で緑部位との差が大きかった720～600nm付近の画像で鮮明に見られる。

2）カエデ：カエデ科（*Acer palmatum*）

モミジとも呼ばれ、秋には真紅に紅葉する。紅葉期、深緑色から真紅まで、紅葉の段階を追って６枚、一

部分ずつ1次元分光計測した（図4）。各葉、目視で知覚した色はかなり異なるが、可視域の分光特性には顕著な差は認められない。最も緑の濃い葉で500～600nmの反射がやや強い。可視域より近赤外域に差異が認められるのは興味深い。

3）イチョウ：イチョウ属（*Ginkgo biloba*）
紅葉期、緑色から、徐々に黄色くなった葉を6枚、1次元分光計測した（図5）。約520～720nmの範囲で、緑色が衰える程反射が強くなっている。細胞の死に伴って光合成が行われなくなり、この範囲の光を吸収しなくなったためと思われる。

4）ユリノキ：モクレン科（*Liriodendron Tulipifera L.*）
紅葉期、緑色から徐々に茶褐色化を呈する5枚の葉を1次元分光分析した（図6）。茶褐色化するにつれ、700nm付近の谷が浅くなっている。

5）ハボタン ：アブラナ科（*Brassica oleracea var. acephala cvs.*）
内側が淡い黄色で外へ向かって緑が濃くなる対象物を選び2次元分光計測した。対象物を白パネルを背景として固定し、本器を5mm毎に水平1次元走査し、45枚の分光画像を得、さらに各画像から2次元分光画像を求めた（図7）。

3．4．今後の課題
本研究では、色の違いが肉眼で確かめられる材料のみを使用し、本器が人間の知覚と同様の能力を持っていることを確かめた。今後、さらに本器が人間の知覚以上の能力を有するか否かの確認を今後の課題として挙げたい。

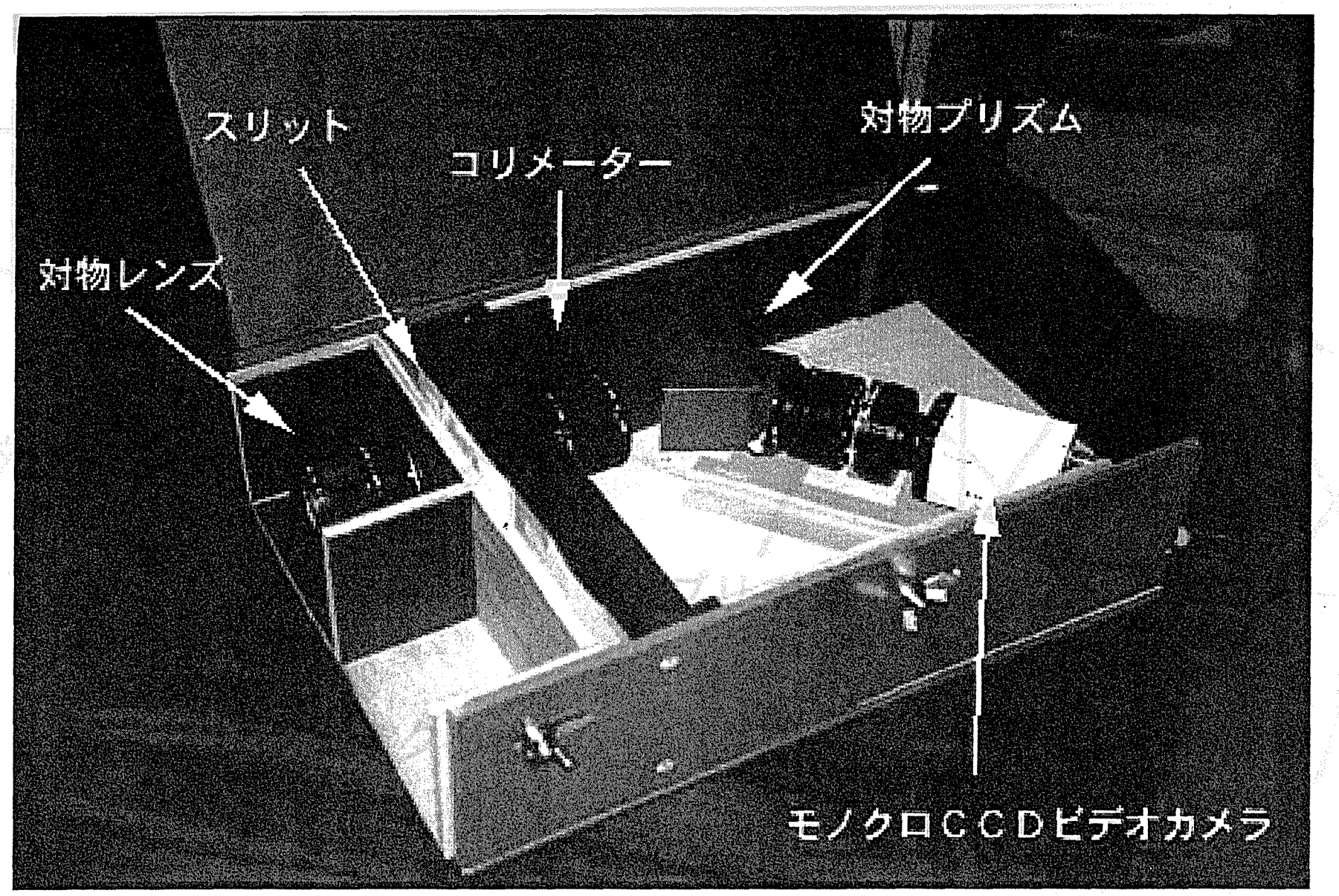


図１．走査型２次元分光分析計概略

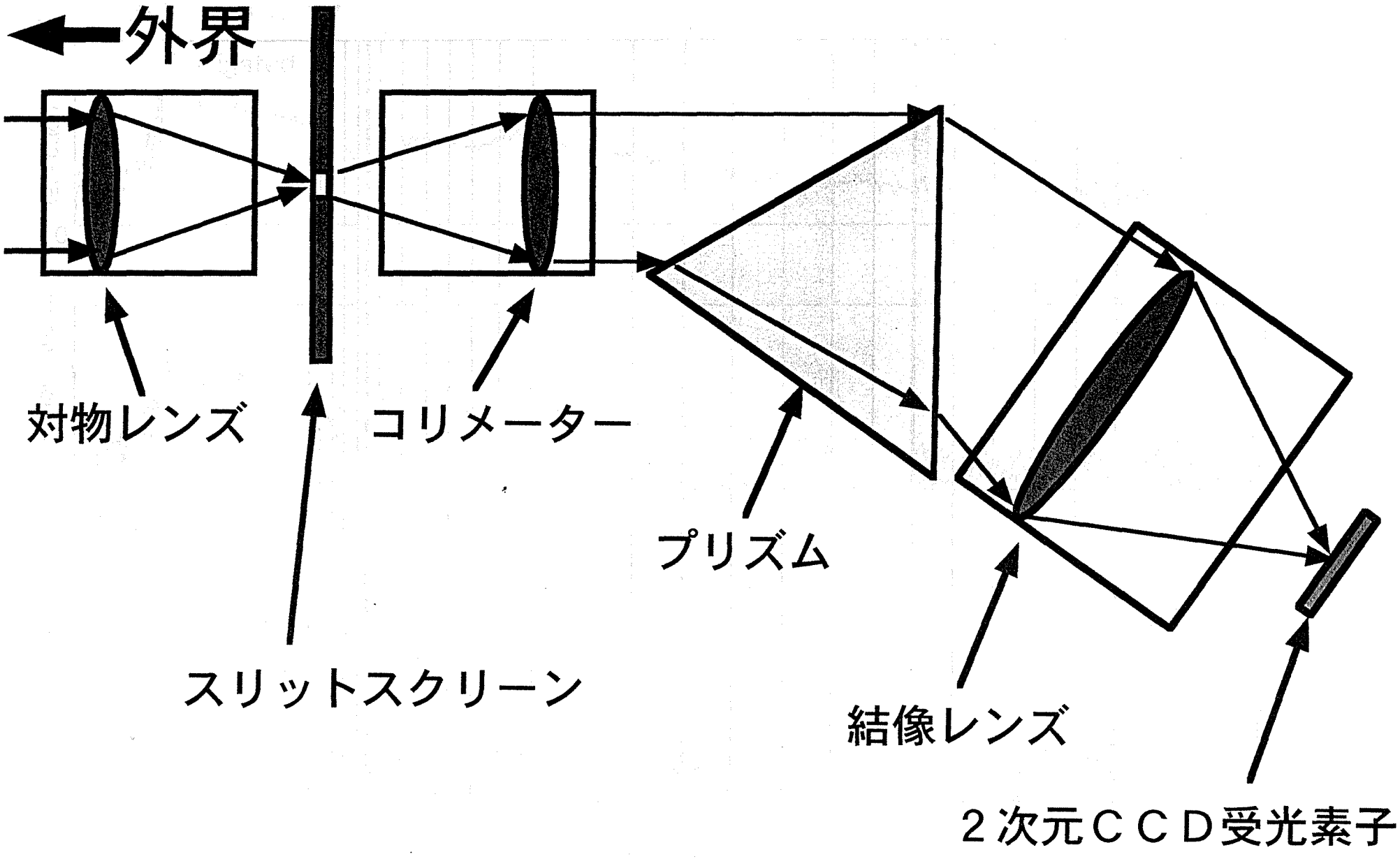


図２．走査型２次元分光分析計内部構造概略

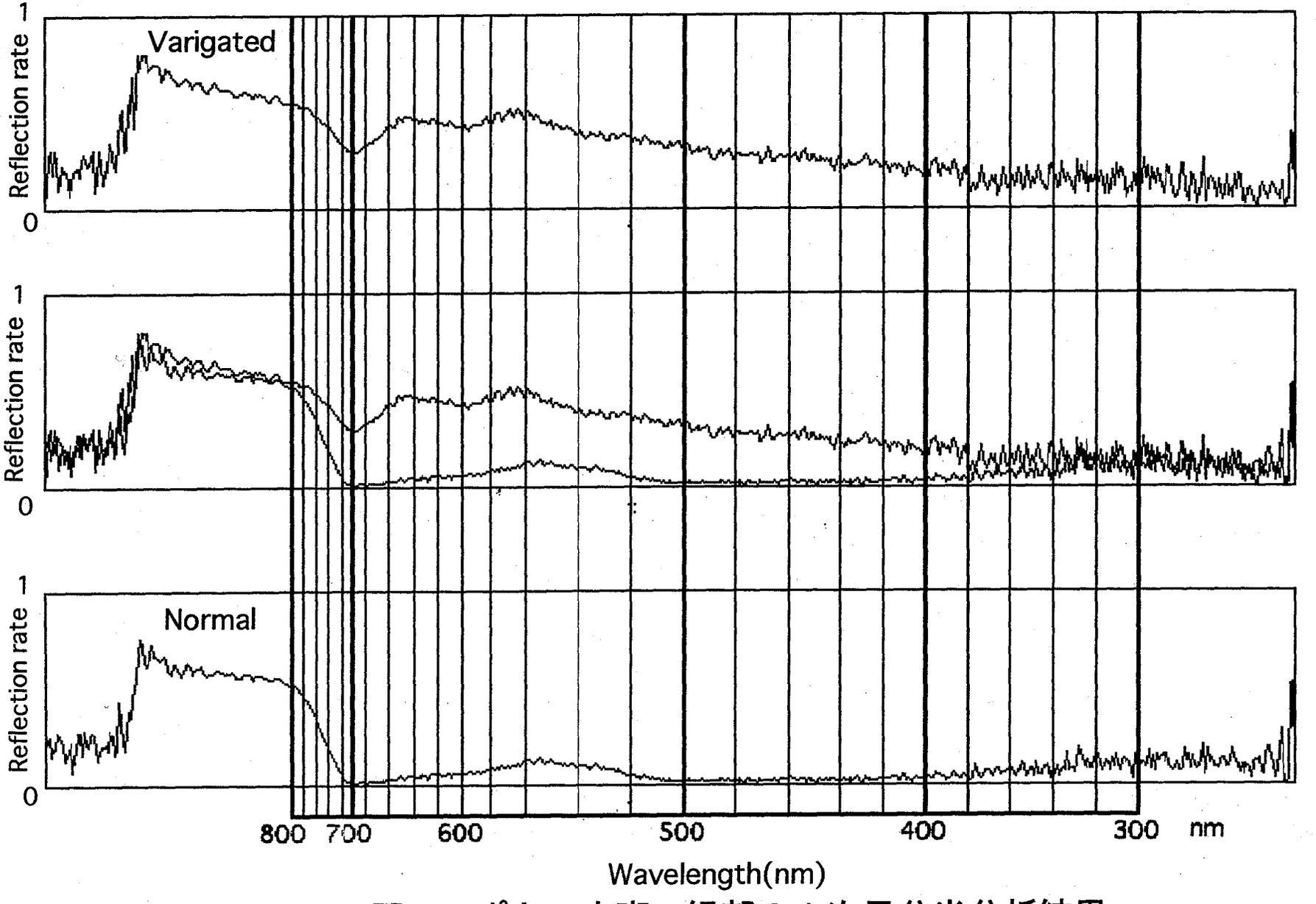


図３．ポトス白斑・緑部の１次元分光分析結果

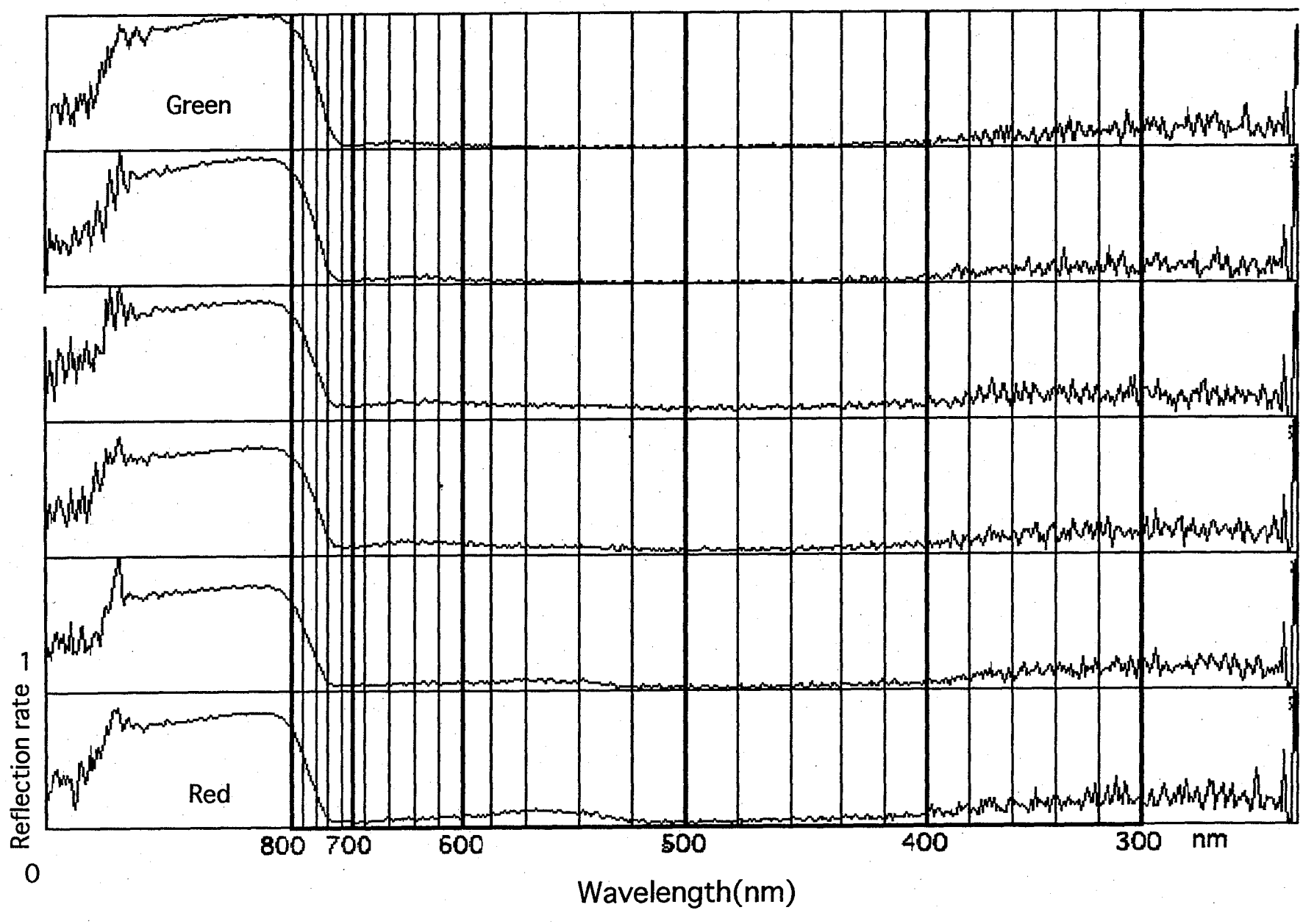


図４．カエデの１次元分光分析結果

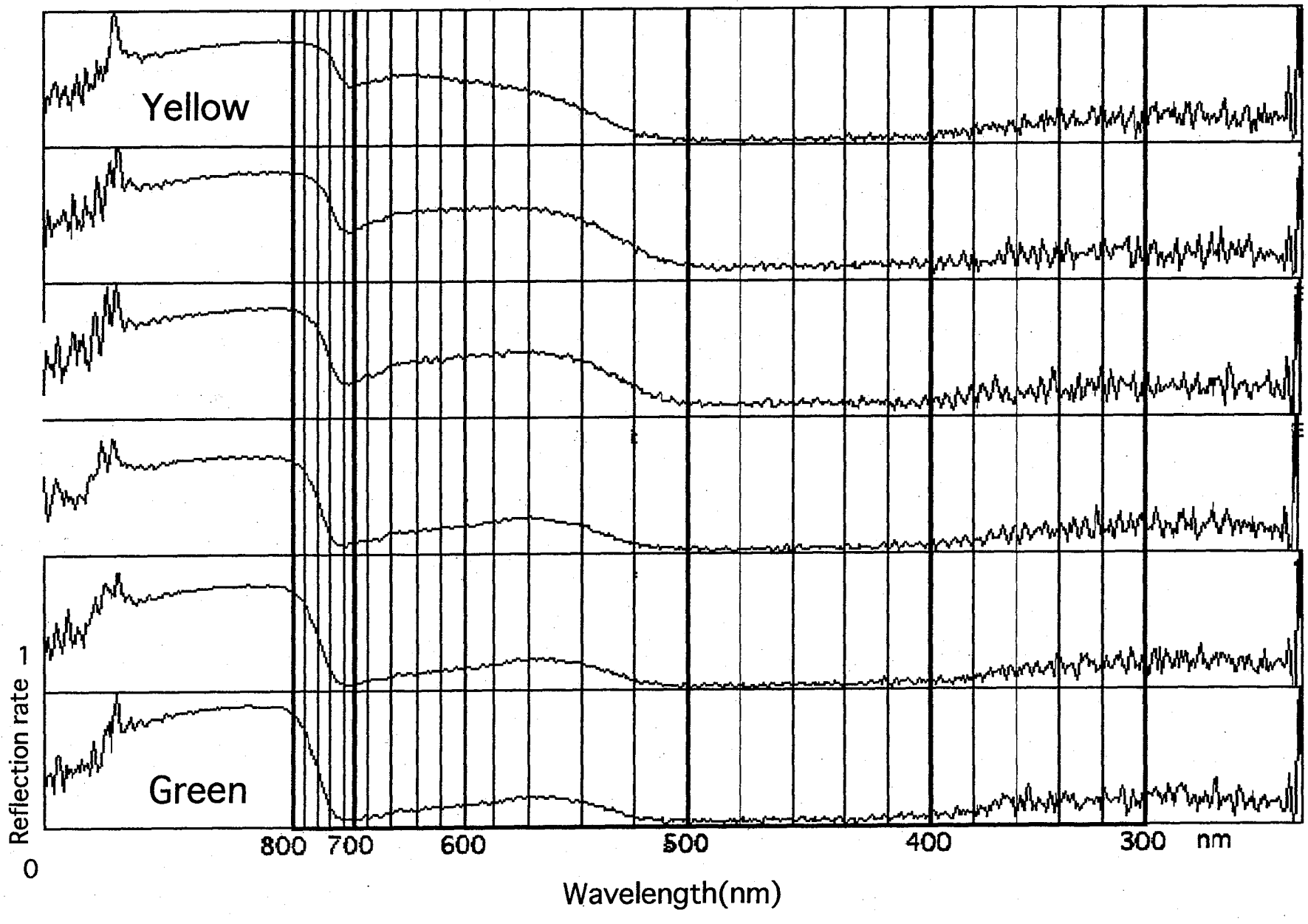


図５．イチョウの１次元分光分析結果

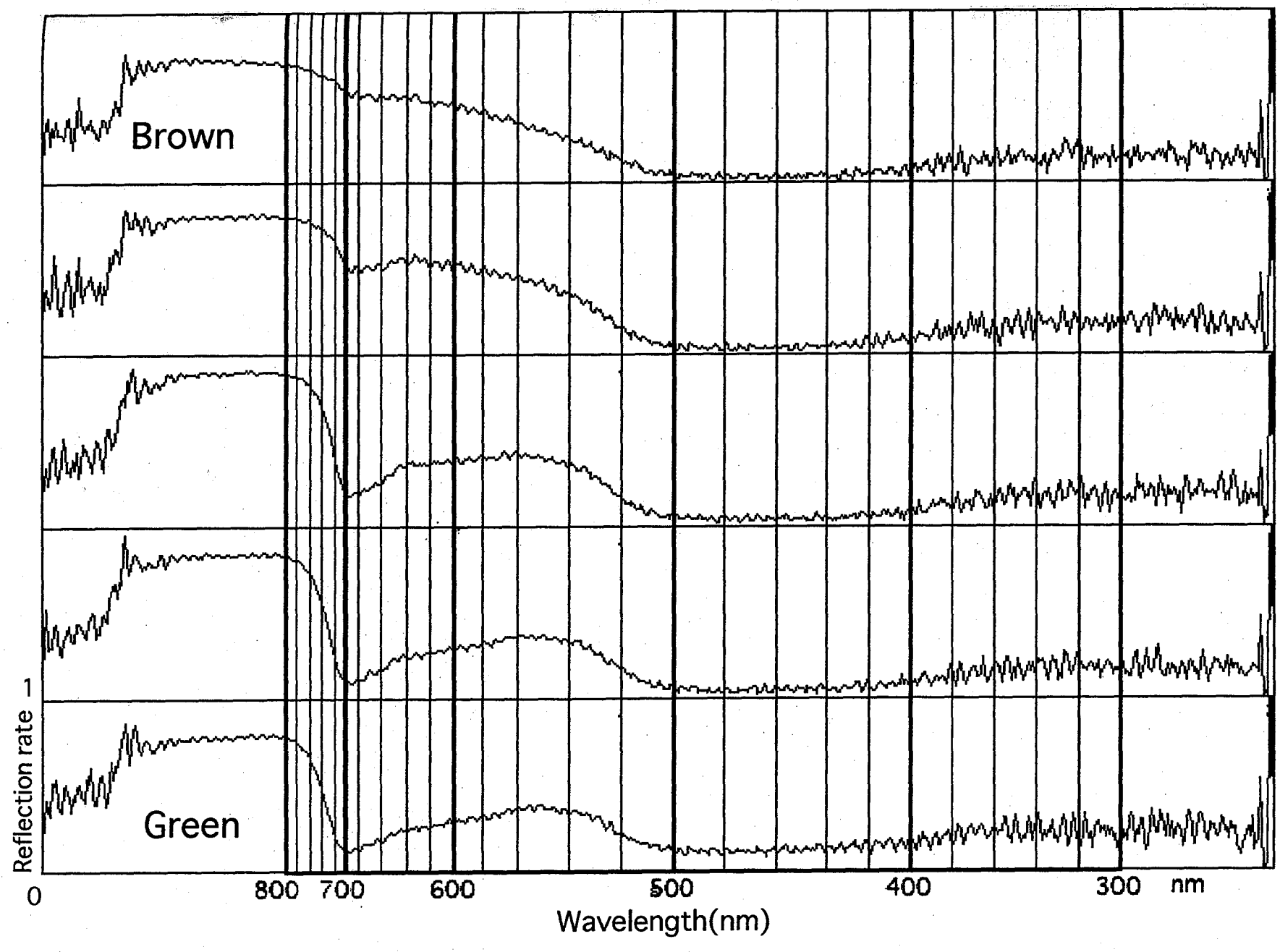
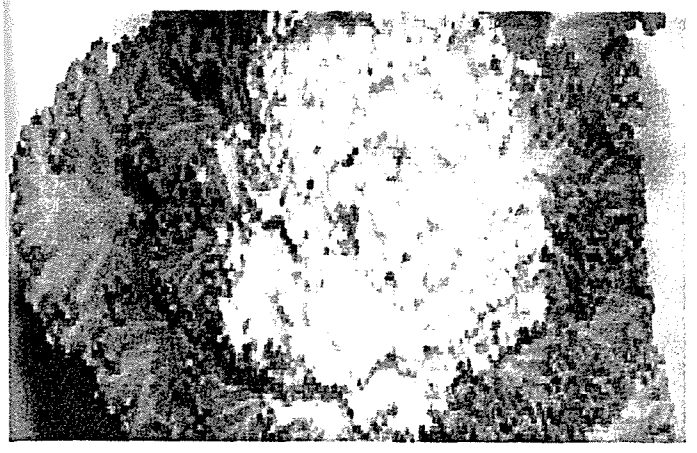
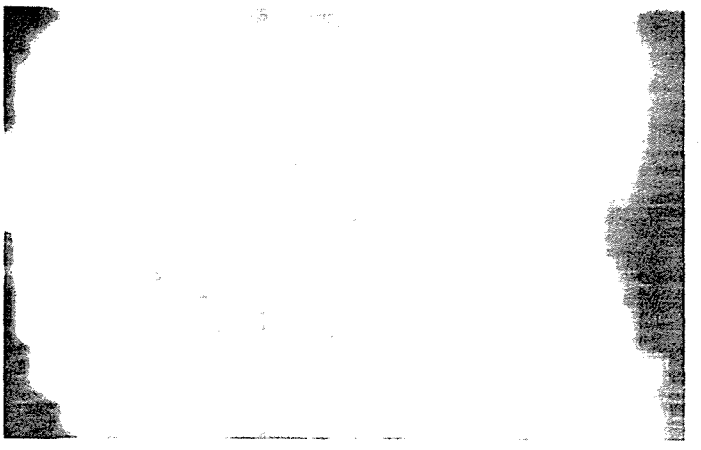


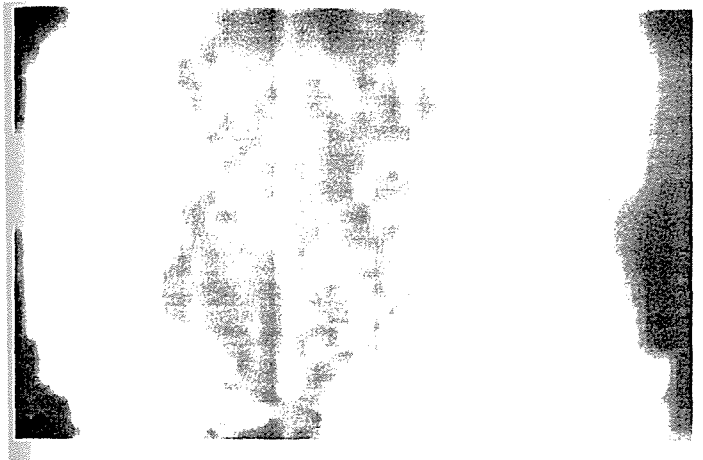
図６．ユリノキの１次元分光分析結果



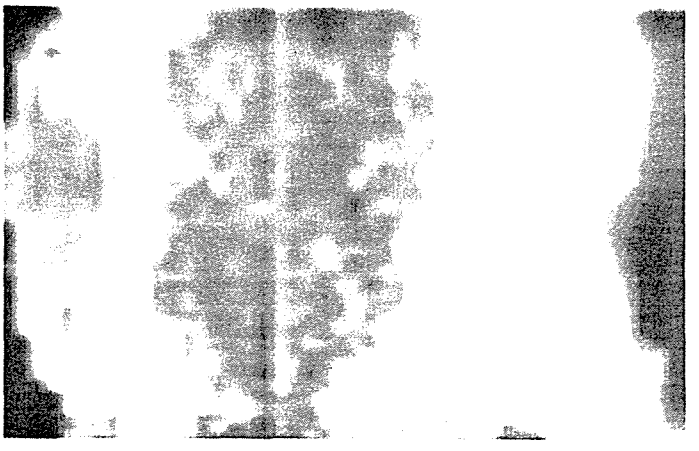
カラー画像



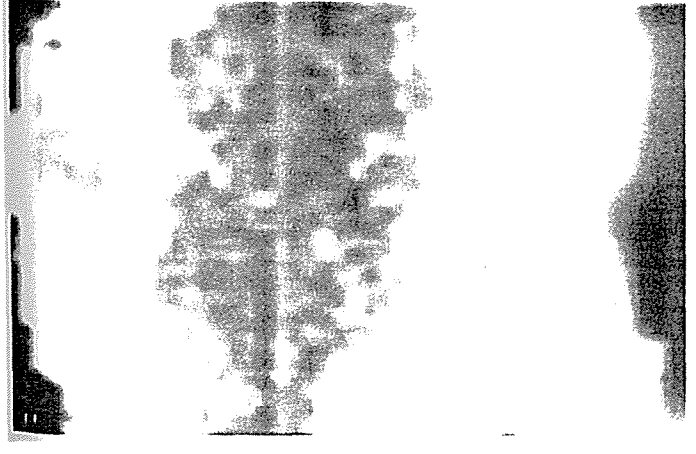
400nm



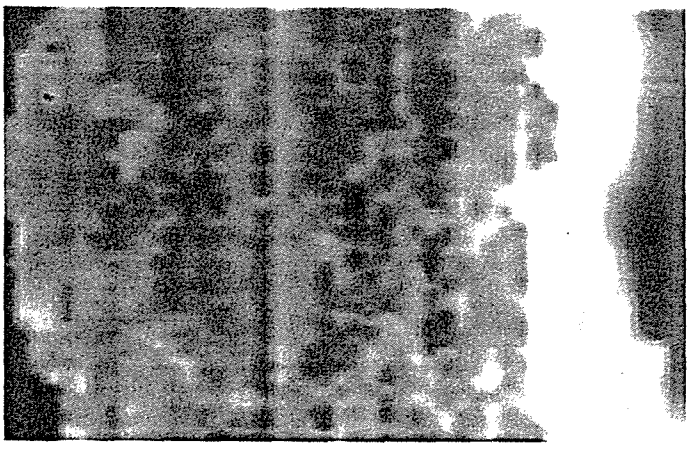
500nm



600nm



700nm



800nm

図７．ハボタンの２次元分光分析結果
（濃度が高いほど反射率が高い）

4．アンケート調査に基づく作物の形状認識および評価方法に関する解析

4．アンケート調査に基づく作物の形状認識および評価方法に関する解析

4．1．はじめに

植物の生育状態は日々刻々と変化するが、その中で農作物もまた様々な自然環境・栽培方法により毎日形態を変えそれが収量に影響してくる。一般に、このような変化を適宜察知・判断を行い、より良い対処をするためには、作物栽培に関する長年の経験を必要とする。同じ作物を何年も栽培し良好な成果を上げている熟練栽培者、いわゆる篤農家は作物の形状を見るだけで状態が良好か良好でないかを判断することができると言われている。この栽培者が作物を観察するときには、ある形態的特徴・情報（葉の傾き、個体全体の輪郭、葉の色など）を認識して作物に接していると考えられる。その際、熟練者がどの部分に着目し、何を基準に判断しているのかを知り、これを解析することで作物栽培に関する知識の少ない素人にでも明確な指針を与えることが可能になると考えられる。また、最近いたるところでオートメーション化による省力化が図られている。作物の栽培管理のオートメーションかを考えると、作物の生育状態を画像情報によって判断するためにも熟練者の知識が必要になると考えられる。

4．2．目的

本研究は、トマトの画像を用いて熟練栽培者と呼ばれる篤農家と、作物に関する詳しい知識を持っていない未熟練者による評価、分類の違いを比較する。また、画像計測に基づく値を用いた分類と未熟練者、熟練者の分類を合わせて比較解析することで、熟練者の作物に対する認識の仕方についての基礎的な知見を見いだす。

4．3．材料および方法

4．3．1．材料

トマト（品種、強力米寿、播種日　1995年3月2日）15個体をビニールポットで育苗した後、ワグネルポットに定植し（定植日　同年5月30日）1995年6月26日、6月28日に撮影したものを今回使用した。撮影は個体別に45°きざみ、8方向から撮影、画像データとしてコンピュータに取り込み、トマトだけの画像を抽出し、これをアンケート材料として用いた。

4．3．2．方　法・手　順

1）アンケートの材料作成

15個体のトマトを赤色の紙を背景に撮影し、フィルムスキャナ(NICON CoolScan)を用いてコンピュータにRGBデジタル画像データとして取り込み、背景の赤と葉群の緑の濃度差が異なることを利用して画像処理を行い、トマトだけの画像を抽出する。詳細は（工藤,1995を参照のこと。図.1にトマトの画像だけを抽出した例を示す。

2）1回目のアンケート方法と処理

トマトの栽培知識の少ない未熟練者（学生10人）に分類評価をしていただいた。1回目は個人が持っている作物に対する意識で判断してもらうため、こちらからは評価基準を指定せず、個人に決定してもらう。また、一個体の評価が全体で100％となるようなパーセンテージで3つのカテゴリーか5つのカテゴリーに分類してもらう。その後、良い評価をつけているものから順にGroupAからGroupEまで並び替えた。例を図.2に示す。同時に、未熟練者が作物に対してどの部分に着目して、どのような評価をつけているかを知るために、別に用意したトマト画像が載っているプリントに印を付けるよう指示した。アンケート結果については、付録1を参照のこと。

次に、各グループの評価に対するパーセントを比較しやすくするために評価のよい順にポイントを与える。良好な個体、GroupAから順に+5.0、+2.5、0.0、-2.5、-5.0である。これは5カテゴリーに分類したものに対してのポイントであるが、3カテゴリーに対しては+5.0、0.0、-5.0とした。そのポイントに各グループのパーセンテージを掛けたものを各個体ごと全部加算し、100で除算して計算した重心値をその個体に対する得点として求める。10人分の得点を求めた後、平均を求め、その値を未熟練者10人の1個体に対する評価とした。この値をもとに、全個体をカテゴリー化して未熟練者の作物の形状に対する認識および評価の状況を検討した。

3）熟練者による分類・評価

1回目のアンケートと同じトマトの画像を熟練栽培者（東北農業試験場　　小沢　聖氏）に見ていただいた。未熟練者に対するアンケートの仕方とは異なり、数値ではなく全体の形などから今までどのように生長してきたかを聞き取り調査し、さらに各個体を分類、評価するよう依頼した。また、各個体の栽培者として

の評価に対する短いコメントもを同時に記録した。この結果については、付録2を参照のこと。
さらに1回目のアンケートの未熟練者の結果と熟練栽培者の結果を比較し、作物の形態に対する認識の違いを検討した。

4）2回目アンケートの方法と処理
2回目のアンケートは、熟練栽培者のアンケート結果を利用した。熟練栽培者が分類した項目と基準を簡単にまとめた文章と模式図で表し（図.3）、2回目のアンケートを始める前に未熟練者に示し、内容を理解するように指示した。2回目のアンケートは、作物の個体数を減らし1回目に使用した個体から、私が選抜した13個体を対象とした。使用した個体は、表1を参照のこと。今回は、4つのカテゴリーはじめから与えておき、どのカテゴリーにそれぞれの個体が当てはまるか、分類してもらう方法をとった。アンケート用紙は、付録3参照のこと。4つのカテゴリーは、「現状は生育良好」「現状は良好でない」「しおれている」「苗時に問題があった」である。このアンケートは、それぞれのカテゴリーに含まれる人数をカウントして、その値をポイントとする。この結果を見て1回目のアンケートの結果に比べ、どれだけ作物形態に対する認識が変わるか、熟練栽培者の評価に近づくかを検討した。

5）画像計測による分類
画像計測による値を使って作物個体を分類するにあたり、ここで使用する値について述べる。値は、葉の傾斜角と、作物体のアウトラインを数値化したものである。以下にその概要を示すが、詳細な測定方法・概念は（庄野,1995）を参照されたい。
葉傾斜角は、パワースペクトル法を用いて求めた。1個体から得られた数値は、個体の高さを49要素に分けて求めたものである。1つの要素をOuter-layer Middle-layer Inner-layerの3つに分けて求め、3つの平均を求めたものである。この値を8要素ごとの平均に個体の上部から6要素に集約する。
作物体のアウトラインは、8方向から撮影した画像から逆投影法を用いて求めた数値である。この値も個体の高さを49要素に分けて求めたもので、これも葉傾斜角と同じく8要素ごとの平均に個体の上部より6要素に集約した。その集約した値を表2に示す。葉傾斜角、作物体のアウトラインの計測に関する概略図は図.4に示す。AaのAはアングル、OaのOはアウトラインをそれぞれ示す。aは特別な意味はない。この値をコンピューター統計処理ソフトのJMPのクラスター分析（K-means法）を用いて5グループ、3グループに分類する（表3(a)~(d)）。コンピューターによる分類が熟練者に近いか、未熟練者に近いかを比較検討した。

6）クラスター分析
クラスター分析は、変数間の関係についての情報から出発して、変数間に意味のあるまとまりを見つけだそうとするための多変量解析である。このようにして得られるメンバーのまとまりをクラスター（群、集団）という。

4．4．結果と考察

4．4．1．1回目のアンケートの結果
1回目は、各個人にトマトの評価基準を決定してもらい5つのカテゴリーか、3つのカテゴリーにそれぞれ分類させた。5つのカテゴリーに分類した人は6人、3つのカテゴリーに分類したのは4人であった。（付録1参照。）
得点化して5個のグループに分けた結果を図に示した（図.5）。飛び抜けて良い、悪いなどの極端な評価は、未熟練者では見られなかった。これは、判断に自信がないせいか評価は全体的に普通の評価に集中した。平均重心値で一番高い値を示したものでも+の1.73で、最低値は-1.61であった。このように+2.0から-2.0の狭い範囲以内に値がおさまった。（表4）

4．4．2．Group毎の特徴
Group1は、作物栽培に関する知識の少ないとされる素人が評価した際に良い評価を得たものである。下葉が繁茂しているものや、育苗時にやや栄養不足などで茎葉の生長が抑制され気味であっても生長するにつれて回復しつつあるものを良い評価とする傾向が見られた。
良い評価が得られなかったGroup5にはしおれている個体だけが集まった。同じ特徴を持つ個体だけでグループを構成しているものはこのグループだけであった。a-14、a-15、b-10は、その例である。
未熟練者にとって評価が”普通”のカテゴリーのGroup3は、全体の形を見ると三角形の輪郭を持つもので大部分をしめている。これらの中には、未熟練者では見分けられない形態的特徴を持つ個体がいくつかあった。例えば、形は良くても生長点が生育不良などと言った一目では気づきにくい特徴である。

Group2で見られるように素人は、下葉部でも上葉部でも過繁茂に見える個体を全体的によい評価のグループに分類する傾向が見られた。
Group4は、しおれ気味や、全体の輪郭が三角形型になっているものでグループを構成していた。
どのグループもやはり、異なった特徴の個体がいろいろ混ざった状態でグループを構成しているものが大部分であった。しかし、これは作物栽培に関する知識が少ないために複数の特徴を持つグループができたと考えられる。評価の良かったものに過繁茂型を選ぶ傾向があり、三角形型は普通の評価、しおれ型は評価が良くない、などの傾向が見られた。

4．4．3．熟練者の形状認識

熟練者のグループ分類評価を見てみると、5つのグループに分類して評価をしている（図6）。良好、過繁茂、三角形型、育苗時に問題があった、水欠乏（しおれている）に分類された。まずまずの評価をつけた個体の特徴としては、下部から上部にかけて等幅間隔で茎葉が発達しているもの、見た目に葉が茂りすぎていないものであった。過繁茂については下葉部上葉部に関係なく、良い評価はもらえなかった。三角形型については、地際に近いほどよく本葉が伸びたもの。すなわち苗時に栄養状態が良すぎたもので、生長後に上葉が水欠などで小さく収縮したり、垂れ下がっているものが対象となっていた。育苗時に問題があったと言われるものは、幼苗のうちにたくさんの水（肥料）を吸わせ過ぎて過繁茂になってしまい、現在もその傾向があるもの、または逆に肥料が少なく茎葉が抑制気味で生長していったものである。
熟練者から全体的に水管理が良くないと指摘を受けた。

4．4．4．画像計測のデータに基づく解析

画像計測による葉の傾斜角、個体のアウトラインを数値化したもの2つを利用したクラスター分析で、グループ分けを観察したところ、はっきりとわかる特徴で分類されているグループが、5つのグループ分けたクラスター分析で見られた（図.7）。下葉部が過繁茂気味で上葉部付近は水不足か他の要因で抑制気味になっている三角形型の個体であった。このことから画像計測による値で個体の輪郭、形状的なものなどは認識できると考えられる。また、未熟練者と熟練者が目視で、しおれている・過繁茂と言う個体の違いを認識できるものでも、画像計測による分類ではしおれも過繁茂も同一のグループに分類されてしまった。このことから画像計測の輪郭と葉傾斜角の２つの分析基準では、しおれと過繁茂の違いを認識しにくいことが分かった。他に分類されたグループは特に共通する特徴もなく、1グループに1個の個体となるような分類も見られた。3つのグループ分けでは、しおれ型、過繁茂型、三角形型、も全部同じグループ分けをされてしまった。
トマト栽培に関する知識がほとんどない状態の未熟練者アンケートの結果と熟練者とを比較した際に注目する点がある。しおれている個体をほぼ熟練者と同じように未熟練者が認識できることである。これは、未熟練者がトマトでなくても何らかの植物を今まで暮らしの中で育ててきたか、身近で観察してきた経験があるはずである。その時、植物のもっともわかりやすい日常的な現象（しおれ）を1度は見ているはずである。その知識が役にたっているのだと考えられる。また、過繁茂に関しても未熟練者は熟練者と違ってよい評価をつける傾向が見られた。数が多ければ、見た目が良ければと言う考えから過繁茂には、よい評価をつけたのではないだろうか。他に共通した特徴となるものは見あたらなかった。

4．4．5．2回目のアンケートの結果

熟練者の知識を文章と模式図で表して、これを2回目のアンケートを行う前に未熟練者に与えた結果、1回目のアンケートと同様にしおれに関する形状の認識は、ほぼ熟練者と一致する結果となった。また、１回目のアンケートで過繁茂に対する認識の仕方とアドバイスを与えたときでは、過繁茂は作物栽培において適切ではないと言うことを理解した人も約半数程度見られた。（図.８）学習することにより作物に対する認識の仕方が若干ながら変化することが分かった。育苗時に何らかの影響を受けた個体を判断することやハウス型、露地栽培型と言ったような詳しいことを形状から認識することは、未熟練者にとって難しいことが分かった。

4．5．考　察

1回目のアンケート、熟練者の結果それぞれを、画像計測のクラスター分析の結果と比較してみた。クラスター分析によるグループ化は、個体の輪郭を基準に分けた１グループと、見た目、黒い固まりのようにみえるしおれ、過繁茂を1グループとして分類している。結果としては、この2グループの中に大部分の個体が含まれた。他の1回目アンケート、熟練者のグループ分け評価は、少数のグループに片寄らずにいろいろな形状に関する認識が見られた。
画像計測に基づく形状認識と人間の形状認識はまだ共通する事柄は少ない。熟練者と未熟練者では各個体を分類することについては、似た性質を持っている。しかし分類が同じであってもそれを評価する知識、経

験が違うので評価に差が出てしまうと考えられる。

未熟練者と、熟練技術者の評価の仕方を比較してみると、しおれている、しおれていないの判断はほぼ熟練者と同様に判断できる結果となった。しおれているという評価は好ましくないと言うことを未熟練者も認識していて、栽培熟練者と呼ばれる篤農家の知識（水分状態に関すること）の一部を未熟練者も身につけていると言えるのであろう。また、過繁茂に対する評価は、はじめ高い評価を付けている未熟練者が多い。これは、ほとんど栽培に関する知識がなく、葉の枚数が多かったり、本葉の広がりの大きかったり左右対称に広がっているものを高く評価する傾向があるためと考えられる。世間一般的にどのようなものに対してでも、見た目がきれいなもの、数が多いものに対しては良い印象を受ける一般的傾向が存在することは否めない事実である。もし、収穫量、品質などを念頭に置きながら栽培に心がけると、ただ見た目だけを考えるのと違って熟練者なみの形状に対する認識が自然と身につくと考えられる。熟練者と未熟練者の着眼点が同じであっても、そこから作物を判断するための経験、知識、基準に差があるため評価に違う結果が出たと考えられる。グループ分けを見ても、少しではあるが熟練者、未熟練者とも同じ分類をしている個体がいくつか見られた。

トマトの画像を被験者に見せるにあたって、画像の大きさによって評価が変わる印象があった。本研究全体を通して見ると、作物の生育に対する評価、分類をアンケートでおこなうにあたって、項目をはっきり絞り、形式を統一したり、集計が容易にできるような方法が望ましいことが分かった。また、画像も静止画像だけではなく、動きのある映像や、アンケートの調査人数をもっと増やす等、さらに改善、工夫する必要があると考えられる。

引用文献

1.古谷野亘　多変量解析ガイド　調査データのまとめかた　川島書店
2.中村正　例解　多変量解析入門　日刊工業
3.庄野浩資　1996　トマト作物体を対象とした葉傾斜角と概略的形状の3次元画像計測　生物環境調節 34:p75-85
4.工藤忠之　1996　画像計測手法の応用によるトマト作物体の形状解析　岩手大学農学部卒業論文

参考文献

1.有馬哲　石村貞夫　多変量解析の話　東京図書
2.菅民郎　やさしい統計学の本　まなぶ　現代数学社
3.野菜園芸大百科　2　トマト　社団法人農山漁村文化協会
4.庄野浩資　岡田益己　樋口誠一郎　1994　近接圃場画像に対するテクスチャ解析手法の有効性　農業気象 49(4):p227-235
5.本條毅　庄野浩資　高辻正基　1993　植物形状の測定と可視化　植物工場学会誌　第4巻　第2号

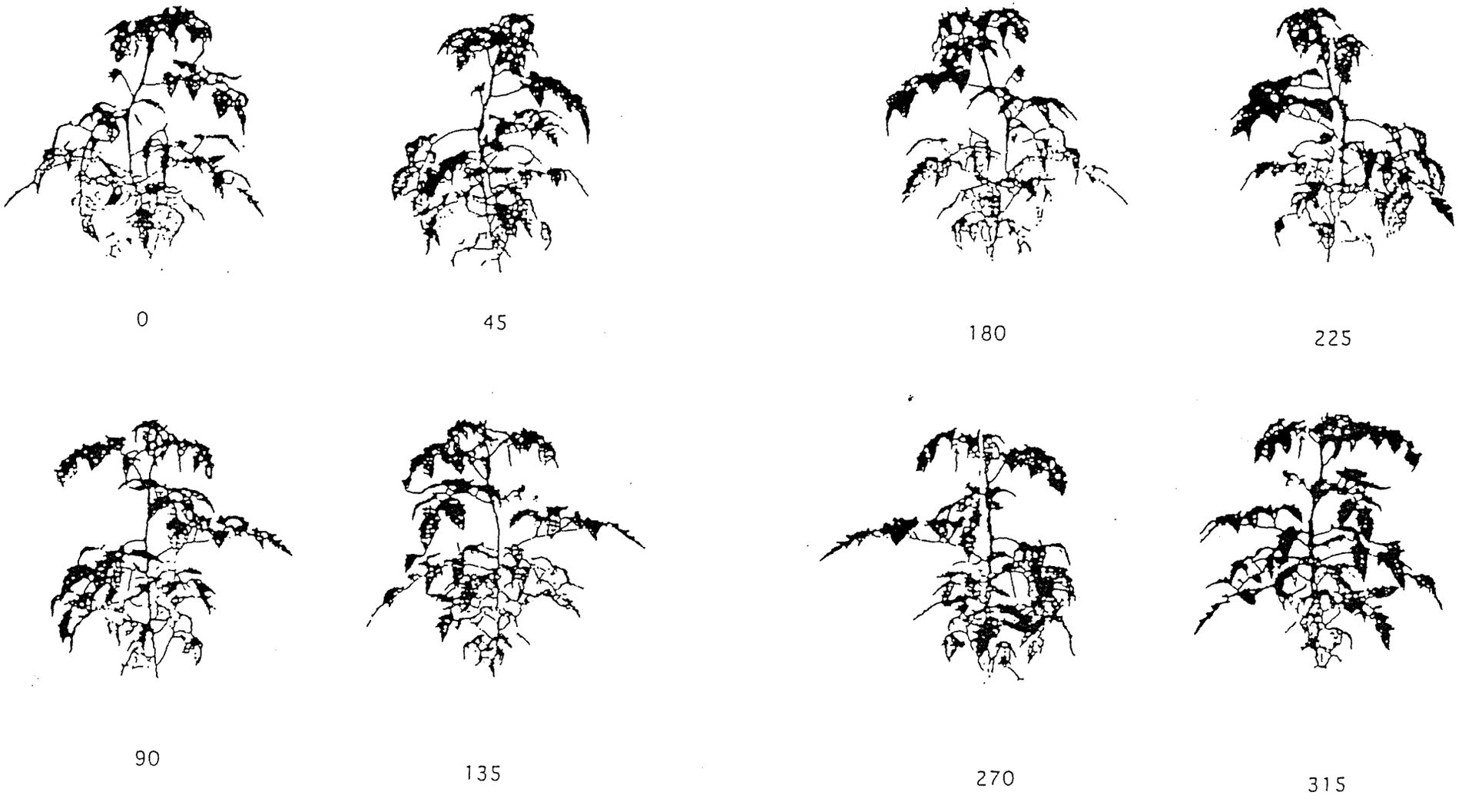
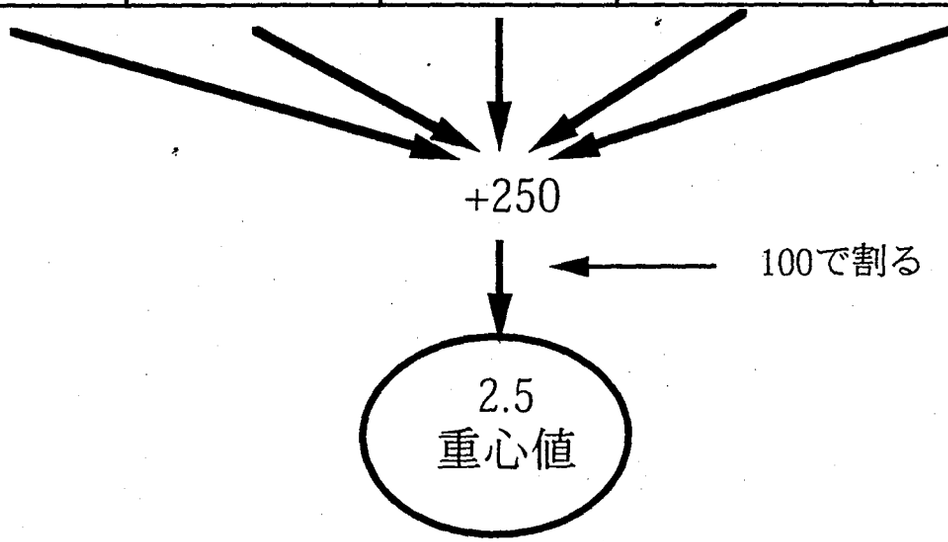


図.１　８方向から撮影したトマトの抽出画像

（例）F-2の得点（重心値）

Good ← → Bad

	Group A	Group B	Group C	Group D	Group E
%	50%	20%	15%	10%	5%
ポイント	+5.0	+2.5	0.0	-2.5	-5.0
ポイント×%	+5.0×50	+2.5×20	0.0×15	-2.5×10	-5.0×5
得点	+250	+50	0	-25	-25



図．2　アンケート形式および処理方法

《トマトの生育に関する情報》

Ⅰ　水・肥料が不足すると、生育は抑制される。

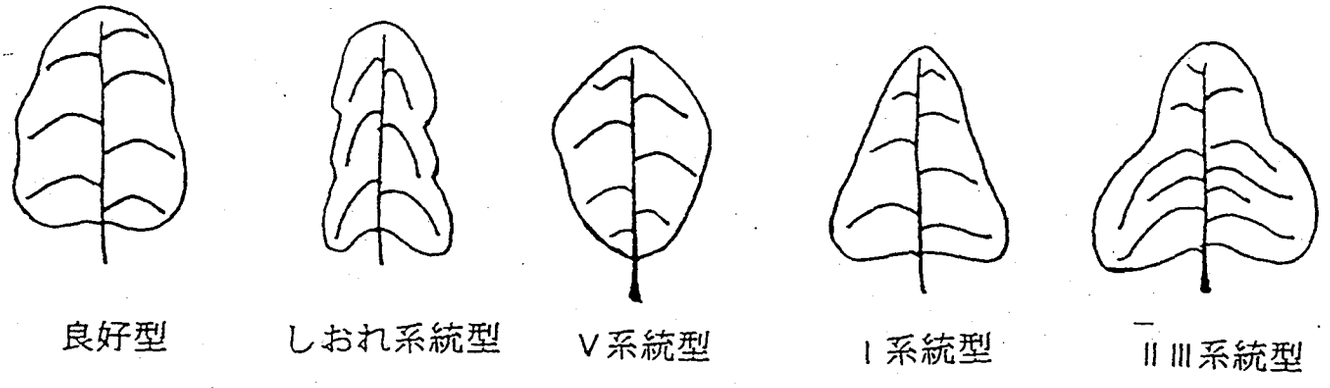
Ⅱ　苗の時に栄養状態が良好すぎると、過繁茂になる。
　　さらに好条件状態が続くとあばれだす。

Ⅲ　過繁茂になると、栄養の分配がうまく行かなくなる傾向がある。

Ⅳ　過繁茂になると、着果、果実の肥大が抑制される。

Ⅴ　苗の生育が不十分で、定植後も回復できないと、根いたみなどがおきる。

Ⅵ　地上部の生育が抑制ぎみになると根の潜在力が強くなる。



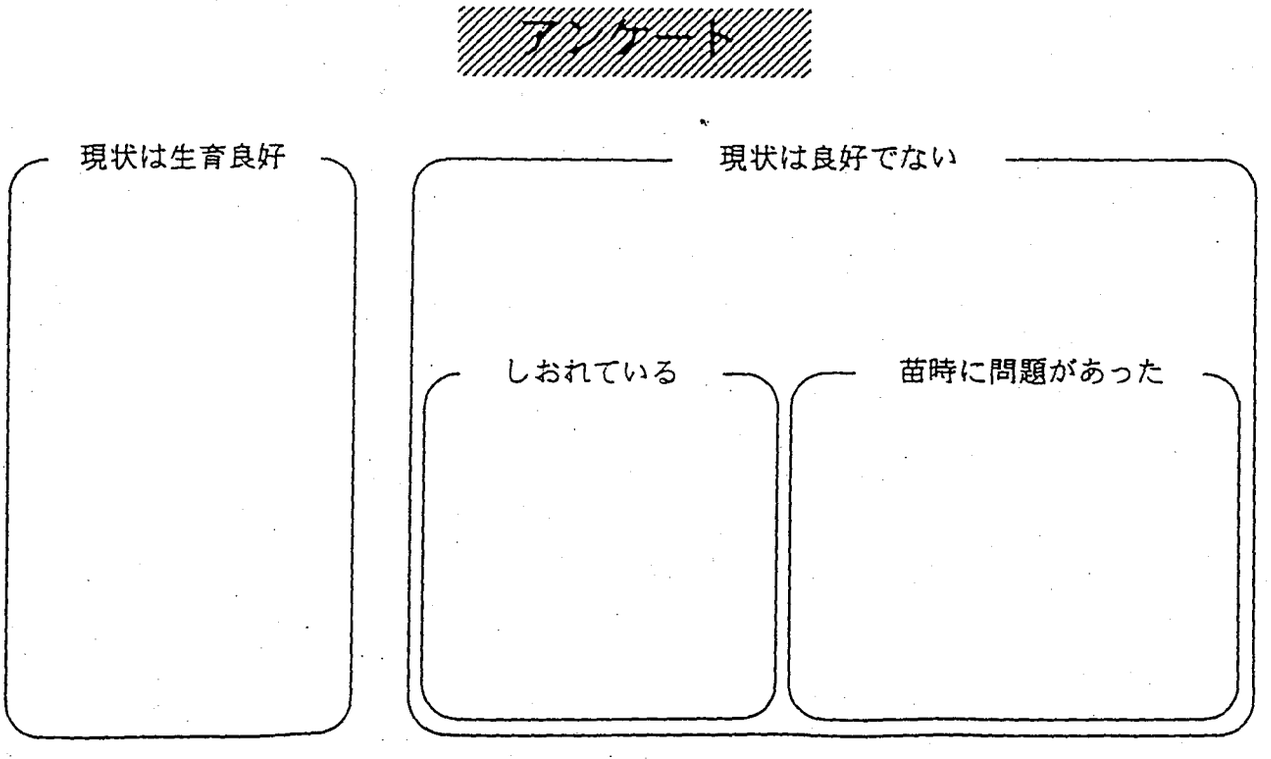


図. 3　2回目のアンケート形式

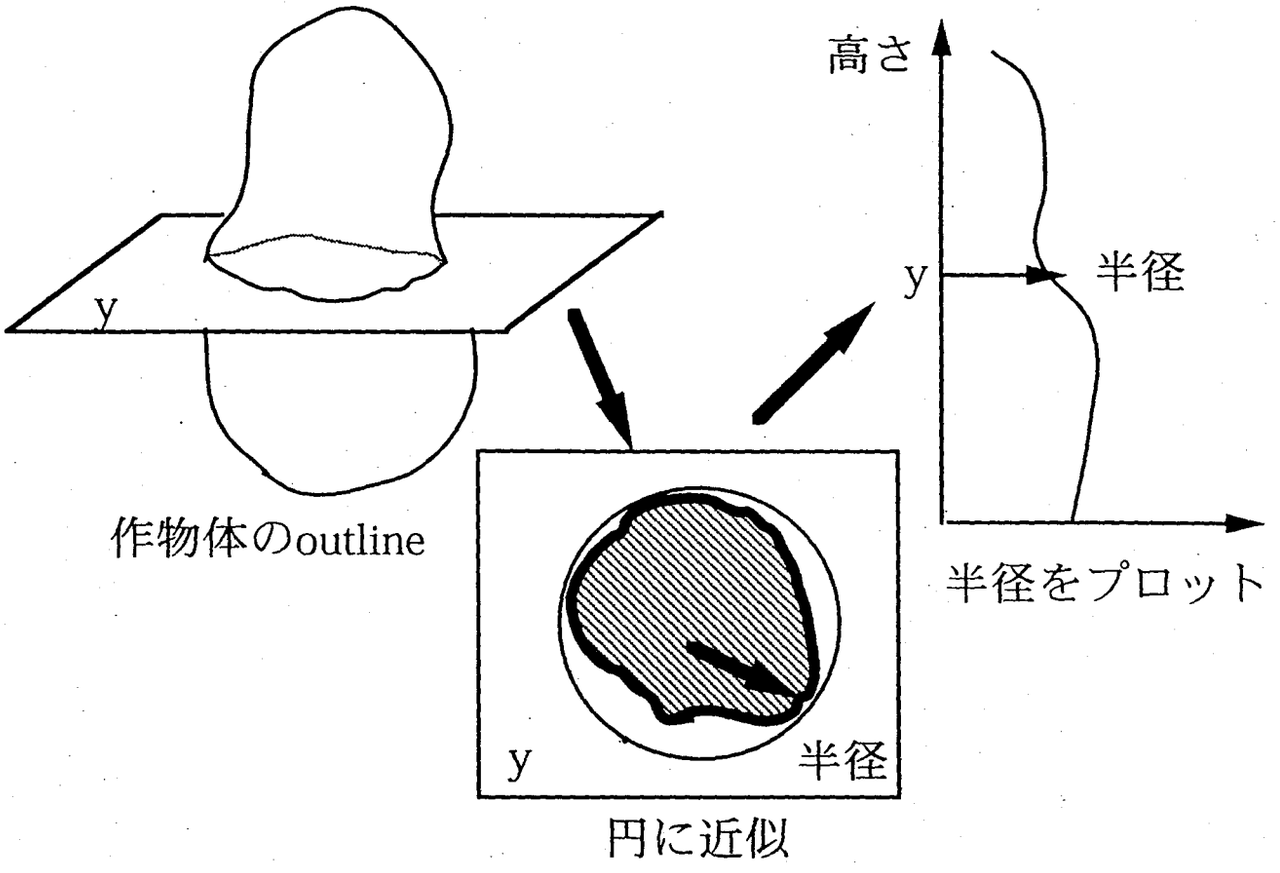


図.4　葉傾斜角と作物体のアウトラインの計測

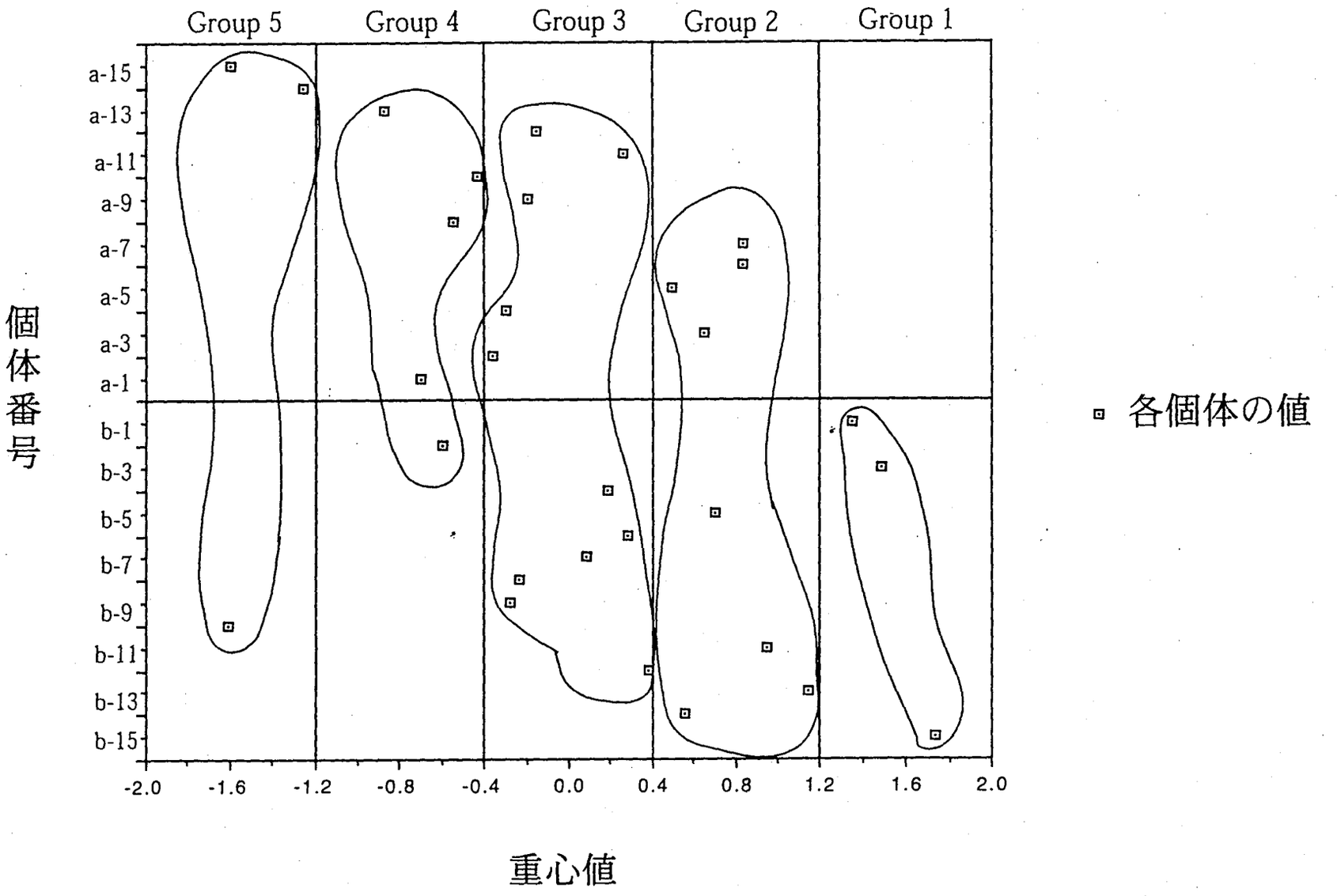
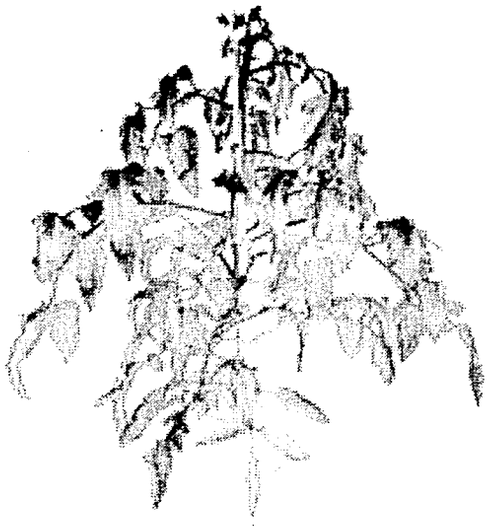


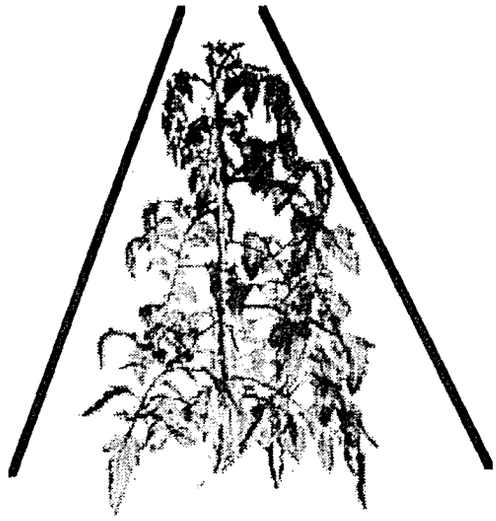
図. 5　１回目アンケートによる各個体の評価と分類

しおれている、どうしようもない

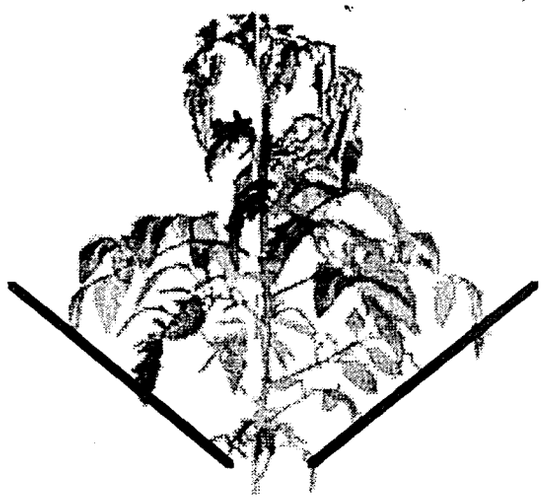
良好



過繁茂



三角形型



育苗時に問題があった

図．6　熟練者による分類例

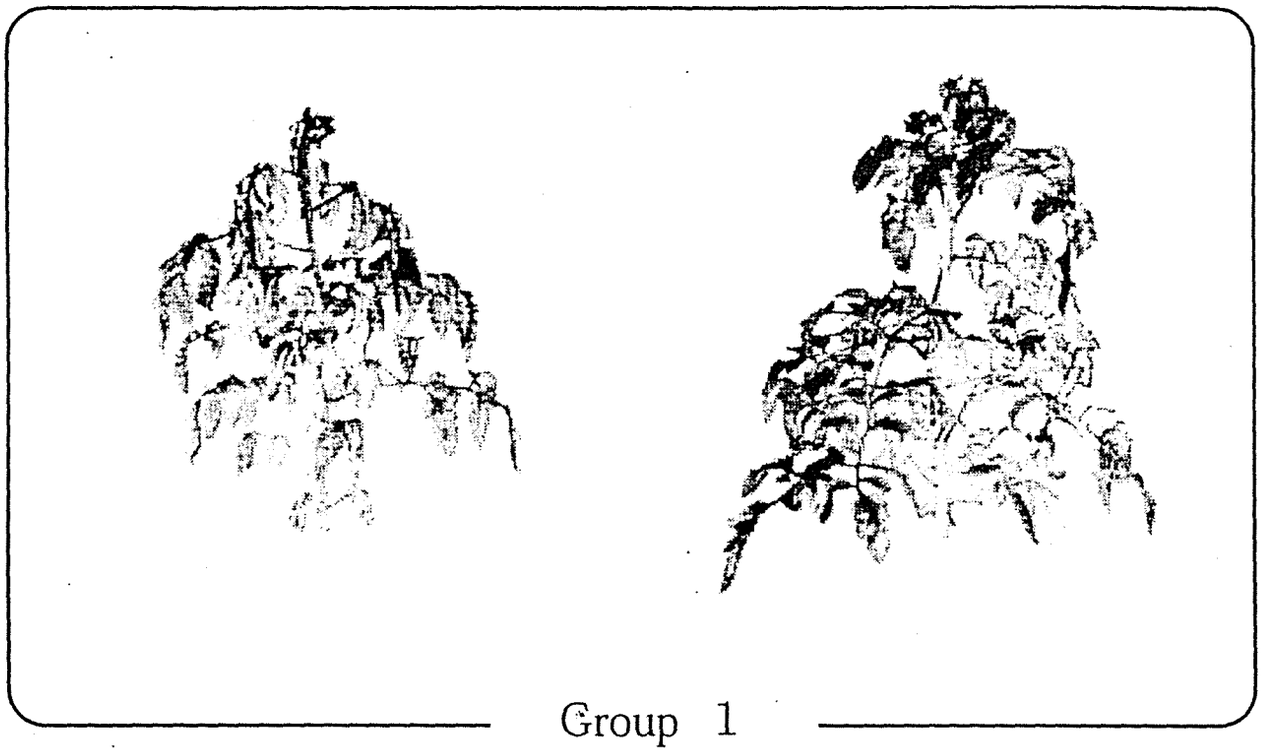
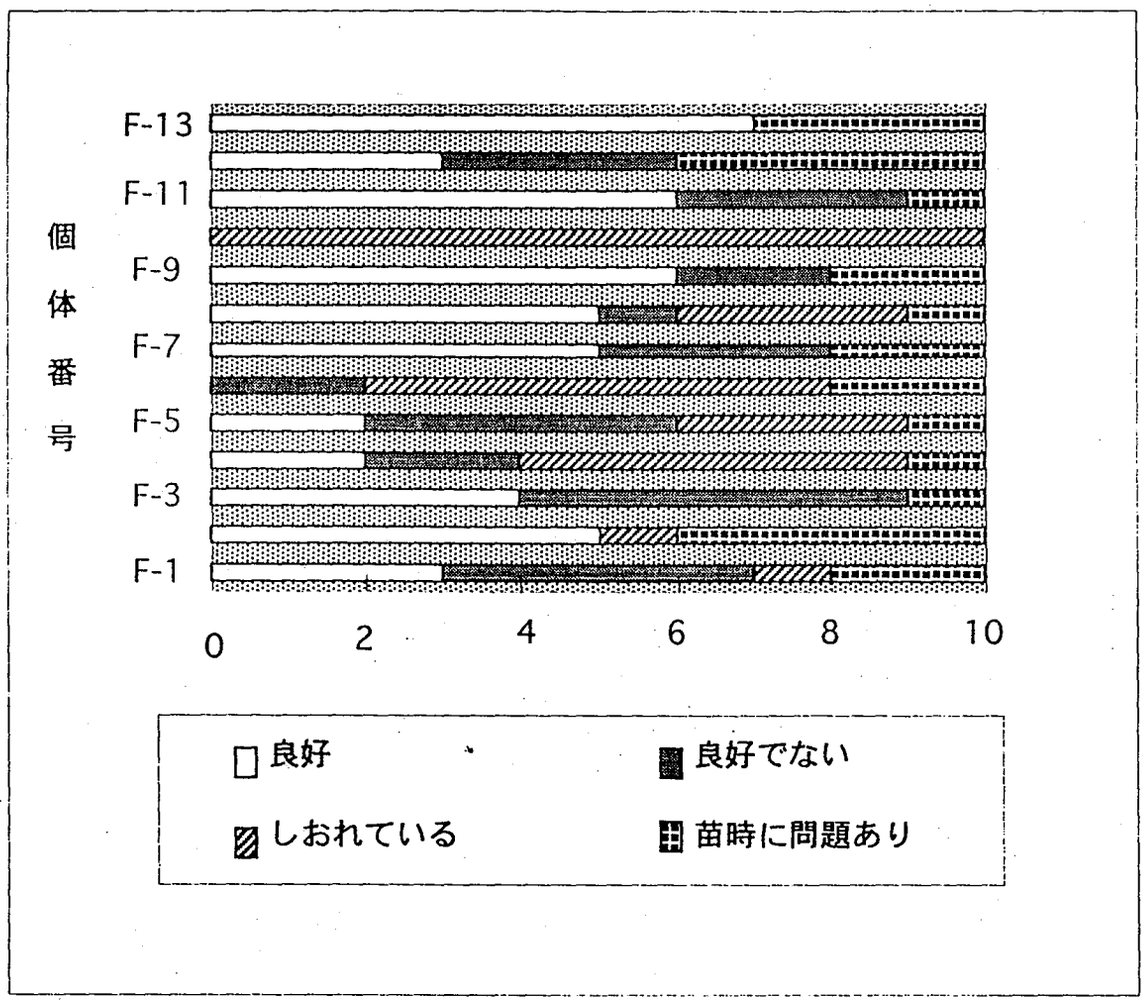


図.7　コンピューターによる分類の例



個体番号	良好	良好でない	しおれている	苗時に問題あり
F-1	3	4	1	2
F-2	5	0	1	4
F-3	4	5	0	1
F-4	2	2	5	1
F-5	2	4	3	1
F-6	0	2	6	2
F-7	5	3	0	2
F-8	5	1	3	1
F-9	6	2	0	2
F-10	0	0	10	0
F-11	6	3	0	1
F-12	3	3	0	4
F-13	7	0	0	3

単位（人）

図．8　２回目のアンケート結果

表.1　２回目のアンケートに使用した個体番号対応表

日付	２回目アンケート番号	１回目アンケートの番号
6月26日	F-1	No.2
6月26日	F-2	No.4
6月26日	F-3	No.6
6月26日	F-4	No.8
6月26日	F-5	No.10
6月26日	F-6	No.15
6月28日	F-7	No.1
6月28日	F-8	No.3
6月28日	F-9	No.4
6月28日	F-10	No.10
6月28日	F-11	No.12
6月28日	F-12	No.13
6月28日	F-13	No.14

表. 2　葉傾斜角と作物体のアウトラインの数値化

Rows	Num.	Day1	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6
1	1	26	22.037	88.2168	37.9844	14.3144	29.5458	15.144	4.297	11.923	13.4926	20.362	20.8252	3.09
2	2	26	0	76.4266	34.12499	25.1112	26.45801	15.386	5.917	9.41202	13.2568	18.6238	17.5278	2.034
3	3	26	22.892	49.49741	5.61824	29.4124	33.96	16.687	5.944	11.7766	17.2696	21.814	22.9462	1.693
4	4	26	31.337	79.34799	49.10619	43.36241	56.35481	12.986	6.232	12.105	14.7632	17.7234	12.8852	1.871
5	5	26	12.736	87.2908	10.17486	9.70472	22.67241	14.551	5.078	12.601	11.4552	20.187	20.1528	1.493
6	6	26	21.234	52.95039	86.02139	10.00671	33.62021	15.505	5.323	12.268	16.4144	19.7114	21.9484	10.69
7	7	26	12.053	83.2732	32.1178	7.55094	0.07092	15.753	5.67	13.5332	15.7366	21.3008	19.0546	4.068
8	8	26	4.346	62.10579	24.38581	5.66515	6.90627	16.71	5.698	10.31824	13.6502	16.4264	18.397	4.478
9	9	26	0	87.8256	40.16059	14.3386	23.69941	16.486	5.232	10.45472	13.3234	17.173	17.8858	0.977
10	10	26	0.2332	57.143	24.5566	5.95023	13.6724	15.407	4.297	12.578	16.5944	19.4648	20.0646	1.128
11	11	26	32.738	59.9488	20.2326	45.7694	35.46481	13.366	4.82	10.522	17.864	21.0632	9.0436	1.382
12	12	26	0	55.5214	27.072	20.45	45.12381	14.737	5.754	8.22988	12.8342	15.5948	15.9228	2.326
13	13	26	0	84.9848	34.929	54.6928	74.767	16.853	5.557	9.36958	17.4386	21.7902	21.0368	1.382
14	14	26	23.541	47.9496	32.4856	51.213	88.1244	14.398	5.046	9.27804	17.834	19.5856	12.143	5.293
15	15	26	35.071	72.2082	34.97079	53.274	63.84721	14.099	3.785	9.12744	16.538	22.3082	15.5596	1.262
16	1	28	9.0801	78.3932	23.19841	25.08581	21.96	90	8.722	14.288	14.429	17.6306	19.4194	1.262
17	2	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	4.853	8.909	11.8218	15.6538	16.4944	4.853
18	3	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	5.293	12.9836	16.2992	19.3918	20.1176	1.596
19	4	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.232	12.8186	13.474	18.8616	14.6926	1.693
20	5	28	0	60.8826	19.197	23.5932	64.16601	33.323	4.82	12.8352	12.352	18.2256	19.6576	0.798
21	6	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.614	10.0614	14.0732	17.3666	20.6804	12.425
22	7	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	3.09	9.8868	14.0866	18.0396	17.088	1.128
23	8	28	0	80.8668	29.9064	54.8998	37.3584	35.949	3.785	9.81604	12.7274	15.3804	15.9388	1.954
24	9	28	9.0801	78.3932	23.19841	25.08581	21.96	90	4.184	9.78344	12.2688	15.1846	15.9202	2.185
25	10	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	5.202	9.0554	12.8262	15.063	15.6858	4.754
26	11	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	6.628	12.04	14.49	21.0064	10.1786	0.564
27	12	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.18	10.3892	11.8858	14.9284	14.7372	5.109
28	13	28	0	60.8826	19.197	23.5932	64.16601	33.323	5.441	14.0832	14.9164	20.6608	22.0862	2.7066
29	14	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.499	14.116	20.1982	15.4138	10.3524	2.111
30	15	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	7.485	11.6518	17.2392	19.8428	13.5806	1.69379

表. 3 (a)　クラスター分析による分類（５）

Rows	Num.	Day1	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6	Cluster	Distance
1	1	28	9.0801	78.3932	23.19841	25.08581	21.96	90	8.722	14.288	14.429	17.6306	19.4194	1.262	1	2.464974
2	11	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	6.628	12.04	14.49	21.0064	10.1786	0.564	1	2.637013
3	15	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	7.485	11.6518	17.2392	19.8428	13.5806	1.69379	1	2.46891
4	2	26	0	76.4266	34.12499	25.1112	26.45801	15.386	5.917	9.41202	13.2568	18.6238	17.5278	2.034	2	1.869325
5	8	26	4.346	62.10579	24.38581	5.66515	6.90627	16.71	5.698	10.31824	13.6502	16.4264	18.397	4.478	2	2.110471
6	9	26	0	87.8256	40.16059	14.3386	23.69941	16.486	5.232	10.45472	13.3234	17.173	17.8858	0.977	2	1.945927
7	12	26	0	55.5214	27.072	20.45	45.12381	14.737	5.754	8.22988	12.8342	15.5948	15.9228	2.326	2	1.997509
8	2	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	4.853	8.909	11.8218	15.6538	16.4944	4.853	2	1.884026
9	7	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	3.09	9.8868	14.0866	18.0396	17.088	1.128	2	3.017209
10	8	28	0	80.8668	29.9064	54.8998	37.3584	35.949	3.785	9.81604	12.7274	15.3804	15.9388	1.954	2	2.595867
11	9	28	9.0801	78.3932	23.19841	25.08581	21.96	90	4.184	9.78344	12.2688	15.1846	15.9202	2.185	2	2.675569
12	10	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	5.202	9.0554	12.8262	15.063	15.6858	4.754	2	1.865946
13	12	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.18	10.3892	11.8858	14.9284	14.7372	5.109	2	2.88427
14	1	26	22.037	88.2168	37.9844	14.3144	29.5458	15.144	4.297	11.923	13.4926	20.362	20.8252	3.09	3	2.531038
15	3	26	22.892	49.49741	5.61824	29.4124	33.96	16.687	5.944	11.7766	17.2696	21.814	22.9462	1.693	3	2.718496
16	4	26	31.337	79.34799	49.10619	43.36241	56.35481	12.986	6.232	12.105	14.7632	17.7234	12.8852	1.871	3	3.183782
17	5	26	12.736	87.2908	10.17486	9.70472	22.67241	14.551	5.078	12.601	11.4552	20.187	20.1528	1.493	3	3.060098
18	7	26	12.053	83.2732	32.1178	7.55094	0.07092	15.753	5.67	13.5332	15.7366	21.3008	19.0546	4.068	3	3.122901
19	10	26	0.2332	57.143	24.5566	5.95023	13.6724	15.407	4.297	12.578	16.5944	19.4648	20.0646	1.128	3	2.792024
20	11	26	32.738	59.9488	20.2326	45.7694	35.46481	13.366	4.82	10.522	17.864	21.0632	9.0436	1.382	3	3.450442
21	13	26	0	84.9848	34.929	54.6928	74.767	16.853	5.557	9.36958	17.4386	21.7902	21.0368	1.382	3	3.455294
22	14	26	23.541	47.9496	32.4856	51.213	88.1244	14.398	5.046	9.27804	17.834	19.5856	12.143	5.293	3	3.973813
23	15	26	35.071	72.2082	34.97079	53.274	63.84721	14.099	3.785	9.12744	16.538	22.3082	15.5596	1.262	3	3.518609
24	3	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	5.293	12.9836	16.2992	19.3918	20.1176	1.596	3	2.840766
25	4	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.232	12.8186	13.474	18.8616	14.6926	1.693	3	2.793473
26	5	28	0	60.8826	19.197	23.5932	64.16601	33.323	4.82	12.8352	12.352	18.2256	19.6576	0.798	3	2.653426
27	13	28	0	60.8826	19.197	23.5932	64.16601	33.323	5.441	14.0832	14.9164	20.6608	22.0862	2.7066	3	2.568669
28	14	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.499	14.116	20.1982	15.4138	10.3524	2.111	4	0
29	6	26	21.234	52.95039	86.02139	10.00671	33.62021	15.505	5.323	12.268	16.4144	19.7114	21.9484	10.69	5	2.382755
30	6	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.614	10.0614	14.0732	17.3666	20.6804	12.425	5	2.382755

表. 3 (b)　クラスター分析による平均値、標準偏差（５）

K-Means Clustering Control

- [x] Standardize data by Std Dev
- [x] Color while clustering
- [ ] Shift distances using sampling rates
- [ ] Use within-cluster std deviations

Go　Step　Help　N of Clusters...

Cluster Summary

Step　2

Cluster	Count	Max Dist	Prior Dist
1	3	2.63701307	7.4360334
2	10	3.01720921	7.28679409
3	14	3.97381296	7.24236297
4	1	0	7.54911839
5	2	2.38275455	7.48114544

Cluster Means

Cluster	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6
1	10.3734333	73.10246	23.7991667	25.4954667	30.86313	64.2393333	7.61166667	12.6599333	15.3860667	19.4932667	14.3928667	1.17326333
2	5.27201	73.313438	26.88679	18.121136	31.382491	35.48338	4.9895	9.625474	12.86812	16.20678	16.55978	2.9798
3	14.1111714	66.057955	27.2519907	29.9765507	43.4037843	20.5239857	5.17942857	11.82389	15.4305571	20.1956429	17.8761143	2.10397143
4	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.499	14.116	20.1982	15.4138	10.3524	2.111
5	21.6645	51.179495	54.16712	6.689765	27.927105	27.082	5.4685	11.1647	15.2438	18.539	21.3144	11.5575

Cluster Standard Deviations

Cluster	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6
1	6.20384695	20.0004207	18.063739	11.3024744	14.4287611	24.8911326	1.05273089	1.42324475	1.60515034	1.71482844	4.67364194	0.57009824
2	6.34522317	15.1922462	9.55917799	15.8597241	12.0431171	26.3841307	1.01073815	0.72356951	0.73758820	1.3137554	1.13725406	1.62894293
3	13.5634756	16.1241612	12.4092075	17.5312836	25.4360201	13.9818476	0.73658158	1.62001328	2.07764945	1.39139966	4.23328671	1.26477726
4	0	0	0	0	0	0	0	0	0	0	0	0
5	0.60881894	2.50442373	45.0487407	4.6908686	8.0512663	16.3723504	0.20576807	1.56030182	1.6554784	1.65802398	0.89661140	1.22683027

表. 3 (c)　クラスター分析による分類（3）

Rows	Num.	Day1	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6	Cluster	Distance
1	1	28	9.0801	78.3932	23.19841	25.08581	21.96	90	8.722	14.288	14.429	17.6306	19.4194	1.262	1	3.04819
2	3	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	5.293	12.9836	16.2992	19.3918	20.1176	1.596	1	2.554487
3	11	28	4.9182	50.98858	42.15579	36.9972	23.1188	62.398	6.628	12.04	14.49	21.0064	10.1786	0.564	1	2.538021
4	14	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.499	14.116	20.1982	15.4138	10.3524	2.111	1	3.42908
5	15	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	7.485	11.6518	17.2392	19.8428	13.5806	1.69379	1	2.850461
6	6	26	21.234	52.95039	86.02139	10.00671	33.62021	15.505	5.323	12.268	16.4144	19.7114	21.9484	10.69	2	2.382755
7	6	28	22.095	49.4086	22.31285	3.37282	22.234	38.659	5.614	10.0614	14.0732	17.3666	20.6804	12.425	2	2.382755
8	1	26	22.037	88.2168	37.9844	14.3144	29.5458	15.144	4.297	11.923	13.4926	20.362	20.8252	3.09	3	2.514179
9	2	26	0	76.4266	34.12499	25.1112	26.45801	15.386	5.917	9.41202	13.2568	18.6238	17.5278	2.034	3	1.827397
10	3	26	22.892	49.49741	5.61824	29.4124	33.96	16.687	5.944	11.7766	17.2696	21.814	22.9462	1.693	3	3.478339
11	4	26	31.337	79.34799	49.10619	43.36241	56.35481	12.986	6.232	12.105	14.7632	17.7234	12.8852	1.871	3	3.360029
12	5	26	12.736	87.2908	10.17486	9.70472	22.67241	14.551	5.078	12.601	11.4552	20.187	20.1528	1.493	3	2.807888
13	7	26	12.053	83.2732	32.1178	7.55094	0.07092	15.753	5.67	13.5332	15.7366	21.3008	19.0546	4.068	3	3.305647
14	8	26	4.346	62.10579	24.38581	5.66515	6.90627	16.71	5.698	10.31824	13.6502	16.4264	18.397	4.478	3	2.531811
15	9	26	0	87.8256	40.16059	14.3386	23.69941	16.486	5.232	10.45472	13.3234	17.173	17.8858	0.977	3	2.221661
16	10	26	0.2332	57.143	24.5566	5.95023	13.6724	15.407	4.297	12.578	16.5944	19.4648	20.0646	1.128	3	2.847936
17	11	26	32.738	59.9488	20.2326	45.7694	35.46481	13.366	4.82	10.522	17.864	21.0632	9.0436	1.382	3	3.896507
18	12	26	0	55.5214	27.072	20.45	45.12381	14.737	5.754	8.22988	12.8342	15.5948	15.9228	2.326	3	2.585083
19	13	26	0	84.9848	34.929	54.6928	74.767	16.853	5.557	9.36958	17.4386	21.7902	21.0368	1.382	3	3.833916
20	14	26	23.541	47.9496	32.4856	51.213	88.1244	14.398	5.046	9.27804	17.834	19.5856	12.143	5.293	3	4.298455
21	15	26	35.071	72.2082	34.97079	53.274	63.84721	14.099	3.785	9.12744	16.538	22.3082	15.5596	1.262	3	4.022342
22	2	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	4.853	8.909	11.8218	15.6538	16.4944	4.853	3	3.063265
23	4	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.232	12.8186	13.474	18.8616	14.6926	1.693	3	2.771224
24	5	28	0	60.8826	19.197	23.5932	64.16601	33.323	4.82	12.8352	12.352	18.2256	19.6576	0.798	3	2.470306
25	7	28	17.122	89.9256	6.0433	14.40339	47.51059	40.32	3.09	9.8868	14.0866	18.0396	17.088	1.128	3	2.900277
26	8	28	0	80.8668	29.9064	54.8998	37.3584	35.949	3.785	9.81604	12.7274	15.3804	15.9388	1.954	3	3.00274
27	9	28	9.0801	78.3932	23.19841	25.08581	21.96	90	4.184	9.78344	12.2688	15.1846	15.9202	2.185	3	3.603028
28	10	28	11.086	79.9362	32.5872	0.5068	33.54301	58.099	5.202	9.0554	12.8262	15.063	15.6858	4.754	3	3.022817
29	12	28	0	42.19699	18.802	20.24381	37.7224	9.0478	6.18	10.3892	11.8858	14.9284	14.7372	5.109	3	3.250513
30	13	28	0	60.8826	19.197	23.5932	64.16601	33.323	5.441	14.0832	14.9164	20.6608	22.0862	2.7066	3	3.055736

表. 3 (d)　クラスター分析による平均値、標準偏差（3）

K-Means Clustering Control

- [x] Standardize data by Std Dev
- [x] Color while clustering
- [ ] Shift distances using sampling rates
- [ ] Use within-cluster std deviations

Go　Step　Help　N of Clusters...

Cluster Summary

Step 3

Cluster	Count	Max Dist	Prior Dist
1	5	3.42907996	7.40673671
2	2	2.38275455	7.51243392
3	23	4.29845541	7.20799861

Cluster Means

Cluster	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6
1	11.6267	63.940912	27.173228	23.371284	27.588438	58.755	6.7254	13.01588	16.53112	18.65708	14.72972	1.445358
2	21.6645	51.179495	54.16712	6.689765	27.927105	27.082	5.4685	11.1647	15.2438	18.539	21.3144	11.5575
3	10.6677522	69.867703	26.4452165	24.516777	39.0590909	25.2074609	5.09191304	10.8176348	14.278687	18.4963043	17.2063391	2.50685217

Cluster Standard Deviations

Cluster	Aa.1	Aa.2	Aa.3	Aa.4	Aa.5	Aa.6	Oa.1	Oa.2	Oa.3	Oa.4	Oa.5	Oa.6
1	7.68564804	18.9128696	15.2876636	14.6170488	11.148955	20.895522	1.42554526	1.18771328	2.37632696	2.18160355	4.80146982	0.57833875
2	0.60881894	2.50442373	45.0487407	4.6908686	8.0512663	16.3723504	0.20576807	1.56030182	1.6554784	1.65802398	0.89661140	1.22683027
3	12.0189659	15.6997375	10.847058	17.7480901	21.6539862	19.984535	0.86390182	1.6858569	2.08241849	2.4564138	3.3479625	1.48256873

表.4　　個人のトマトに対する得点

	小林	竹田	小野寺	高松	藤井	高橋	薄衣	上野	関	田替	平均	分散	標準偏差
a-1	-0.25	-1.00	-1.00	-0.50	-2.38	-2.50	-1.13	-0.50	1.25	1.00	-0.70	1.34	1.16
a-2	-2	-2.00	-2.50	-1.00	-1.25	-1.38	1.75	2.75	0.25	1.75	-0.36	3.11	1.76
a-3	-0.5	1.50	-3.00	0.50	-0.25	0.50	1.00	2.50	1.50	2.75	0.65	2.49	1.58
a-4	-0.5	-0.50	-3.00	-1.00	-0.50	2.50	0.75	-0.25	-2.25	1.75	-0.30	2.49	1.58
a-5	0.25	0.00	-2.50	1.50	-1.00	2.00	-0.25	2.00	1.63	1.25	0.49	1.95	1.39
a-6	1	0.50	1.00	2.50	-0.75	-1.00	1.63	0.88	1.88	0.75	0.84	1.05	1.03
a-7	0.5	0.50	0.00	2.50	0.00	1.88	0.50	0.00	1.50	1.00	0.84	0.68	0.82
a-8	-1.5	-1.50	0.00	-0.50	-0.88	-2.75	0.63	-0.13	0.88	0.25	-0.55	1.13	1.06
a-9	-1	-1.00	2.00	1.00	-1.00	-3.00	1.63	-1.13	0.50	0.00	-0.20	2.08	1.44
a-10	-0.25	-0.50	-3.00	-0.50	-1.00	-2.50	0.63	1.63	1.25	0.00	-0.43	1.96	1.40
a-11	0.5	0.50	3.00	1.50	-0.75	-0.25	-1.00	-1.13	0.00	0.25	0.26	1.40	1.18
a-12	-0.5	-2.50	0.50	-1.00	1.63	0.25	1.25	-1.50	-0.88	1.25	-0.15	1.64	1.28
a-13	-1.25	1.00	-3.00	3.00	-2.00	-1.63	0.25	-2.50	-1.13	-1.50	-0.88	2.94	1.71
a-14	-0.5	-2.00	0.00	-0.50	-2.00	-2.25	0.38	-3.00	0.25	-3.00	-1.26	1.58	1.26
a-15	-1.25	-1.00	-2.50	-1.50	-1.63	-3.13	0.25	-2.63	0.13	-2.75	-1.60	1.24	1.11
b-1	0	0.00	2.00	0.50	1.25	2.75	1.50	2.25	1.75	1.50	1.35	0.78	0.88
b-2	-2	0.00	-2.00	-2.50	-1.75	-3.50	2.38	1.38	1.00	1.00	-0.60	3.55	1.88
b-3	0.75	0.50	2.50	4.00	-1.75	2.50	3.13	0.75	1.00	1.38	1.48	2.38	1.54
b-4	0.5	0.50	-2.50	0.00	0.50	0.75	0.63	-1.13	1.13	1.50	0.19	1.24	1.11
b-5	0.25	1.00	0.50	2.50	-0.25	0.63	-0.63	0.00	1.50	1.50	0.70	0.80	0.89
b-6	0	1.00	2.50	1.50	-1.00	-3.25	2.00	-1.00	0.75	0.25	0.28	2.59	1.61
b-7	-0.5	0.50	2.50	2.00	0.00	-2.00	2.50	-2.50	0.75	-2.50	0.08	3.40	1.84
b-8	-0.25	-0.50	0.50	1.00	0.38	-1.25	1.63	-1.00	-0.38	-2.50	-0.24	1.27	1.13
b-9	-1.25	0.00	1.50	0.50	-0.25	-1.50	2.75	-2.00	0.25	-2.75	-0.28	2.48	1.58
b-10	-2	-1.00	-1.50	0.00	-2.00	-4.50	1.13	-3.00	-0.25	-3.00	-1.61	2.48	1.58
b-11	0.5	-1.00	-2.00	2.50	0.50	3.75	-0.50	1.75	2.00	2.00	0.95	2.81	1.68
b-12	-1.25	-2.00	1.50	-0.50	1.13	0.00	1.63	1.00	0.50	1.75	0.38	1.49	1.22
b-13	-0.5	1.00	-1.00	4.00	-0.25	0.88	0.25	2.38	2.13	2.50	1.14	2.26	1.50
b-14	0.25	-0.50	-0.50	0.00	1.75	-0.25	0.88	0.88	1.38	1.75	0.56	0.70	0.84
b-15	2	1.00	2.00	3.00	1.25	2.00	1.88	2.50	0.38	1.25	1.73	0.53	0.73