



Osaka Gakuin University Repository

Title	ベジエ曲線によるダイコンの形状記述とその応用 Shape Description of Radish by Bézier Curve and its Application
Author(s)	淡 誠一郎 (Seiichiro Dan) 淡 裕美子 (Yumiko Dan) 吉田 康子 (Yasuko Yoshida)
Citation	大阪学院大学 人文自然論叢 (THE BULLETIN OF THE CULTURAL AND NATURAL SCIENCES IN OSAKA GAKUIN UNIVERSITY), 85-86 : 27-57
Issue Date	2023.03.31
Resource Type	Article/ 論説
Resource Version	
URL	
Right	
Additional Information	

ベジエ曲線によるダイコンの形状記述とその応用

淡 誠 一 郎¹・淡 裕 美 子²・吉 田 康 子³

Shape Description of Radish by Bézier Curve and its Application

Seiichiro Dan¹・Yumiko Dan²・Yasuko Yoshida³

はじめに

果物や野菜の形状は人の嗜好や収穫・加工の効率などに大きく影響する。形状は土壌や気候、栽培過程などの環境要因にも影響されるが、遺伝的要因によるところが大きい。本研究で扱うダイコンの形状にも様々な遺伝子が関わる事が予想され、ダイコンの効率的な育種を行ううえでは根形に関わる遺伝的メカニズムの解明が期待される。本来、量的形質である根形に関わる遺伝子の同定には、形状の量的な測定が必要である。しかし、従来研究では、楕円フーリエ記述子を用いた形状計測方法^[1,2]が量的な手法として挙げられるぐらいであり、ほとんどの研究では根形は質的にしか評価されてこなかった。

図形の形状記述法としてはさまざまな方法が考えられる。楕円フーリエ記述子を含む各種フーリエ記述子を使ったダイコンの形状記述とその問題点についてはすでに報告した^[2,3]。本報告では、コンピュータで滑らかな曲線を扱う際に用いられるパラメトリック曲線のうち、その代表であるベジエ曲線を用いた形状記述とその応用例について述べる。

フーリエ記述子は、図形のシルエット全体の特徴を少数のパラメータで捉えることができるため、形状の定量評価や分類のための強力なツールである（例えば SHAPE^[4]）が、曲がりを持った個体の長さや幅の計測、軸に沿った幅の変化の様子を捉えるなどといった

1 大阪学院大学 情報学部 Faculty of Informatics, Osaka Gakuin University

2 神戸大学大学院 農学研究科 Graduate School of Agricultural Science, Kobe University

3 神戸大学大学院 農学研究科附属食資源教育研究センター Food Resources Education and Research Center, Graduate School of Agricultural Science, Kobe University

目的には不向きである。そのような目的には、任意の曲線を少数の制御点で記述でき、曲線上の位置を1つのパラメータで連続的に表せるパラメトリック曲線の方が適する。本稿で解説するベジエ曲線を用いた形状記述法は、ダイコンの多様性評価のための形状解析や計測に利用しており、その成果については [5,6]ですすでに報告済みである。これらの報告は農学的な視点での成果報告であり、利用している画像処理プログラムの詳細については説明されていないので、ここに報告する次第である。

2. 対象画像と前処理

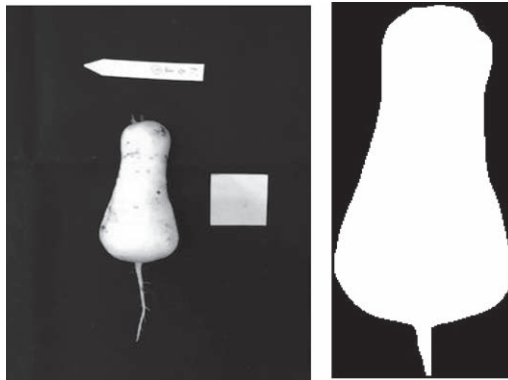


図1 入力画像と記述対象のシルエット画像の例

図1は本研究で対象とする実画像の一例と記述の対象となるシルエット画像である。実際に形状記述の対象とするのは右のシルエット画像の輪郭上に等間隔に取った標本点の列である。実画像からシルエット画像、シルエット画像から輪郭の標本点の列を得る手順は先の研究ノート[2]で述べた。

以下では標本点の列

$$[(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)]$$

を対象物の形状あるいはシルエットと同一視する。 $(x_k, y_k), k = 1, 2, \dots, M$ はシルエットの輪郭線から等間隔でサンプリングされた点の座標である。ここではデジタル画像を前提としているので、各座標成分は整数で表される。点列はほぼ等間隔に取られるが、デジタル画像であるのでサブピクセル単位の量子化誤差を含む。

3. ベジエ曲線によるシルエット記述についての基本考察

3.1 ベジエ曲線とは

ベジエ曲線はコンピュータグラフィックスにおいて滑らかな曲線を描くための代表的な

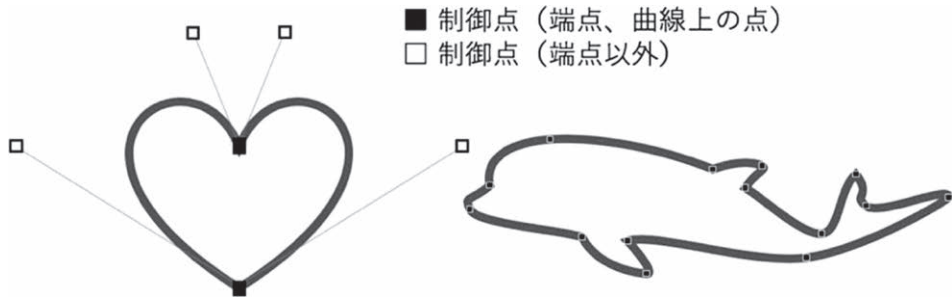


図2 ベジエ曲線で描かれた図形の例

手法である。図2にベジエ曲線で描かれた図形の例を示す。ベジエ曲線はいくつかの制御点によって区分的な曲線を定義する。複数の区分曲線をつなぎ合わせていくことで、いくらでも複雑な曲線を表現できる。

ベジエ曲線上の各点は1つのパラメータ変数の関数として表現される。そのような曲線表現はパラメトリック曲線とよばれる。先の研究ノート[2]で扱ったフーリエ記述子では経路長をパラメータとして形状を記述したが、それもパラメトリック曲線の一種である。ベジエ曲線を拡張したパラメトリック曲線として、B-Spline 曲線や有理ベジエ曲線、さらにそれらの拡張である NURBUS 曲線など、ベジエ曲線より表現力の高い手法もあるが、表現力が高いものほど曲線を定義するための項目が増える。イラストやCG制作が目的であるならば、表現力や自由度は高いほどよいので、ベジエ曲線よりも B-Spline、B-Spline よりも NURBUS 曲線の方が優れていると言えるが、解析や識別が目的であれば、表現能力が必要十分な範囲で、記述が簡潔な方が好ましい。そこで、ここでは制御点を用いたパラメトリック曲線の中でもっともシンプルな記述であるベジエ曲線を取り上げる。

ベジエ曲線の定義

有限個の制御点 B_0, B_1, \dots, B_N と $[0,1]$ を定義域とするパラメータ t により、次式で定義される曲線を N 次のベジエ曲線とよぶ。

$$B(t) = \sum_{n=0}^N b_{N,n}(t) B_n \quad \dots(1)$$

ここで、

$$b_{N,n}(t) = {}_N C_n (1-t)^{N-n} t^n \quad \dots(2)$$

はバーンスタイン基底関数とよばれる。 ${}_N C_n$ は N 個から n 個を選ぶ組み合わせの数である。

バーンスタイン基底関数に関して、

$$\sum_{n=0}^N b_{N,n}(t) = \sum_{n=0}^N {}_N C_n (1-t)^{N-n} t^n = (t + (1-t))^N = 1 \quad \dots(3)$$

が成り立つ。すなわち、バーンスタイン基底関数 $b_{N,n}(t), n=0,1,\dots,N$ の合計は常に 1 であり、ベジエ曲線においては制御点の混合比率を表していると解釈できるので、ブレンディング関数とも呼ばれる。

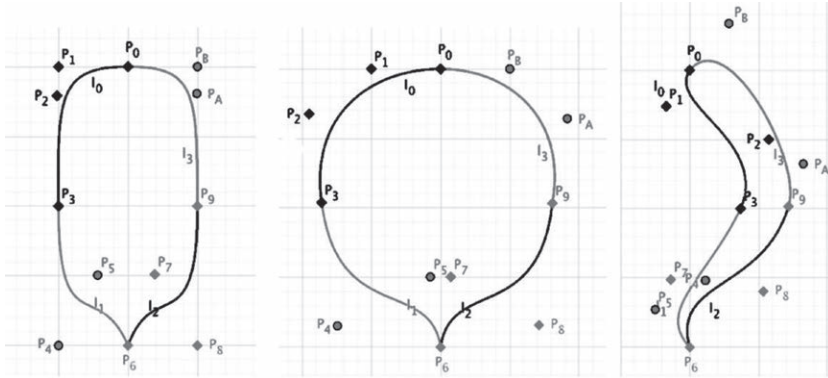


図3 ベジエ曲線をつないで描かれたダイコンふうの図形

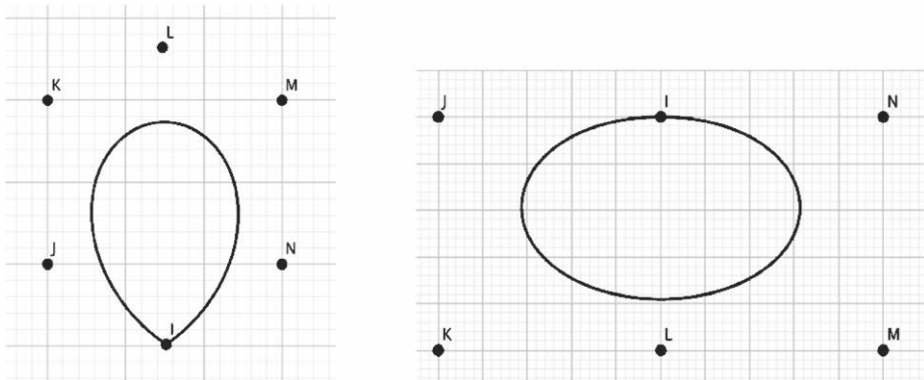


図4 6次ベジエ曲線で描かれた閉曲線

ベジエ曲線をつなぎ合わせることで、複雑な図形を形作ることができる。例えば、図3の3つのダイコンふうの図形は、それぞれが4つの3次ベジエ曲線をつなぎ合わせて描かれている。制御点はそれぞれ計12であり、この12点の配置だけで形状が定まる。次数を上げてうまく制御点を配置すれば、1つのベジエ曲線で閉曲線を形作ることもできる。図4の2つの図形は共にI,J,K,L,M,N,Iを制御点とする6次ベジエ曲線である。

この様に、ベジエ曲線で形状を近似すれば少数の制御点だけで連続的に変化する輪郭形状を表現することができる。分割数あるいは制御点の数を増やせばいくらかでも複雑な形状を表現することができる。また、ベジエ曲線は曲線上に同じ x 座標を持つ点が複数存在するような曲線でも扱えるという点も重要である。そのような曲線には多項式近似は適用できないが、ベジエ曲線による近似は可能である。実際、本研究で扱うダイコンの輪郭の全

周はもちろんのこと、2分割した場合でも曲線上の座標値の成分同士が一意対応となるとは限らない。

ベジエ曲線の制御点の位置は、図形をスケーリング、回転、平行移動させると変化する。しかし、楕円フーリエ記述子などと同様に、楕円近似の長軸を基準として正規化すれば、ベジエ曲線の制御点は形状を規定する不変特徴とみなせるので、フーリエ記述子同様、形状の分類や識別に応用できる可能性もある。

3.2 点列へのベジエ曲線のあてはめ

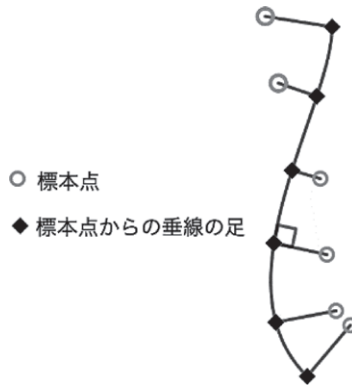


図5 曲線当てはめの誤差

輪郭線をベジエ曲線で近似するということは、標本点の集合 $\{P_k = (x_k, y_k): k = 1, 2, \dots, M\}$ にもっとも当てはまるベジエ曲線の制御点を見つけるということである。あるベジエ曲線と与えられた標本点とのずれを、次の式で定義し、この値が最小となる曲線を求めることを考える。

$$E = \frac{1}{2} \sum_{k=1}^M |P_k - B(t_k)|^2 \quad \dots(4)$$

ただし、 $B(t_k)$ はベジエ曲線上で標本点 $P_k = (x_k, y_k)$ に対応づけられる点で、 t_k はその点に対応するベジエ曲線のパラメータである。 $B(t_k)$ は理想的には標本点 P_k と一致すべきであるが、当然誤差は生じる。曲線の方を固定した場合、(4)式を最小化するためには、 $B(t_k)$ を P_k に最も近い点、すなわち P_k から曲線に下ろした垂線の足に一致させればよい(図5)。 t_k の値は求めたいベジエ曲線に依存して決まるものであるが、一旦それは忘れてこの値が既知であると仮定し、最小二乗法を用いて曲線あてはめを行うことを考える。具体的な計算方法は以下の通りである。

当てはめに用いるベジエ曲線の次数を N 、制御点を $\{B_n = (u_n, v_n): n = 0, 1, \dots, N\}$ とする。誤差関数 E は非負であり、制御点の座標の連続関数であるのだから、誤差が最小となる場合があるはずであり、そのとき、

$$\begin{cases} \frac{\partial E}{\partial u_n} = 0, & n = 0, 1, \dots, N \\ \frac{\partial E}{\partial v_n} = 0, & n = 0, 1, \dots, N \end{cases} \quad \dots(5)$$

が成立するはずである。そこで、この連立方程式を解くことで、誤差最小の制御点の組み合わせ、もっとも標本点にフィットするベジエ曲線を求める。

ベジエ曲線の式を成分に分解すると、

$$\mathbf{B}(t) = (B_x(t), B_y(t)) = \left(\sum_{n=0}^N b_{N,n}(t)u_n, \sum_{n=0}^N b_{N,n}(t)v_n \right) \quad \dots(6)$$

(5)の第一式に(4)式、(6)式を順に代入して式変形していくと、

$$\begin{aligned} \frac{\partial E}{\partial u_n} &= \sum_{k=1}^M \left((B_x(t_k) - x_k) \frac{\partial B_x(t_k)}{\partial u_n} \right) = \sum_{k=1}^M \left((B_x(t_k) - x_k) b_{N,n}(t_k) \right) \\ &= \sum_{k=1}^M \left(\left(\sum_{i=0}^N b_{N,i}(t_k)u_i - x_k \right) b_{N,n}(t_k) \right) \\ &= \sum_{i=0}^N \left(\sum_{k=1}^M b_{N,i}(t_k)b_{N,n}(t_k) \right) u_i - \sum_{k=1}^M x_k b_{N,n}(t_k) = 0 \end{aligned} \quad \dots(7)$$

が得られる。この式は

$$A_{n,0}u_0 + A_{n,1}u_1 + \dots + A_{n,N}u_N = B_n \quad \dots(8)$$

ただし、

$$A_{n,i} = \sum_{k=1}^M b_{N,i}(t_k)b_{N,n}(t_k), \quad B_n = \sum_{k=1}^M x_k b_{N,n}(t_k) \quad \dots(9)$$

と書ける。

パラメータ $t_k: k = 1, 2, \dots, M$ を固定した場合、 $A_{n,i}, B_n$ は定数となるので、(8)式は $N+1$ 個の未知数 u_0, u_1, \dots, u_N の線形方程式である。この方程式は $n = 0, 1, \dots, N$ について、計 $N+1$ 個作れるので、それらをすべて連立させて解けば、仮のパラメータ $\{t_k\}$ の下ではあるが、標本点への最相当てはめのベジエ曲線を得ることができる。(5)の第二式 (y 成分) についてもまったく同様である。

3.3 漸近的ベジエ近似アルゴリズム

上で述べた手順は最小二乗法による曲線回帰の常套的な枠組みをベジエ曲線に適用したにすぎない。そうするために各標本点に対応するパラメータ t_k を定数として扱ったが、実際にはあてはめ前に t_k はわかっていない。高次の N 次ベジエ曲線に対して、各標本点

から降ろした垂線の足の座標を制御点と標本点の座標により陽に表現するのは困難であるし、かといって、 t_k まで未知数として扱うと、線形方程式に帰着できなくなる。

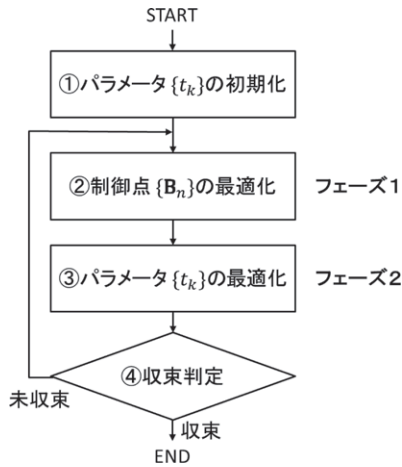


図6 漸近的ベジエ近似アルゴリズム

そこで、図6に示すように、

フェーズ1 暫定的なパラメータ $\{t_k\}$ の下で制御点 $B_n: n = 0, 1, \dots, N$ を最適化

フェーズ2 暫定的な制御点 $\{B_n\}$ の下でパラメータ $t_k: k = 1, 2, \dots, M$ を最適化

という2つのフェーズを残差が収束するまで繰り返すことで、漸近的に標本点に近似ベジエ曲線をフィッティングさせていくようなアルゴリズムを考える。

処理の流れをもう少し詳しく説明する。

処理の流れ

- ① パラメータの初期化。区間 $[0,1]$ を標本点の数で等分割した値を各標本点に対応付ける。すなわち、 $P_k: k = 1, 2, \dots, M$ に対するパラメータを $t_k = (k-1)/(M-1)$ とする。
- ② パラメータ $\{t_k\}$ を固定し、あてはめ誤差が最小となるように制御点の集合 $\{B_n: n = 0, 1, \dots, N\}$ を最適化する。(フェーズ1)
- ③ 制御点 $\{B_n\}$ できまるベジエ曲線に対し、各標本点に対して距離が最小となる点のパラメータを求め、パラメータ $t_k: k = 1, 2, \dots, M$ を更新する。(フェーズ2)
- ④ 収束条件を満たしていれば終了。さもなければ、②に戻り、処理を繰り返す。

3.4 アルゴリズムの実装

アルゴリズムを、表1に示す4種の方法 fit0, fit1, fitT0, fitT1 で実装してみた。違いは、上で説明した流れの②と③の実装法の違いである。②の制御点の決定に関して、fit0, fit1, fitT1 は最小二乗法を用い、fitT0 は勾配法を用いる。③のパラメータの決定に関して、fit0

表1 漸近的ベジエ近似アルゴリズムの実装方法

実装法名称	②制御点 $\{B_n\}$ の最適化方法	③パラメータ $\{t_k\}$ の最適化方法
fit0	最小二乗法 (3.2で説明した計算法)	①で初期化した値のままで固定
fit1	最小二乗法 (3.2で説明した計算法)	パラメータ区間[0,1]を細かく区切って暫定的な近似曲線上に密に候補点を設定し*, 各標本点 $P_k: k=1, \dots, M$ に対して距離最小の候補点を見つけ, そのパラメータを t_k とする.
fitT0	勾配法 (Tensorflow, Adam) 暫定的な $\{t_k\}$ の下で, 前の繰り返しで求めた $\{B_n\}$ を初期値とし, 勾配法による $\{B_n\}$ の更新を mloop 回繰り返す. (default: mloop=3)	勾配法 (Tensorflow, Adam) 暫定的な $\{B_n\}$ の下で, 前の繰り返しで求めた $\{t_k\}$ を初期値とし, 勾配法で $\{t_k\}$ を更新する.
fitT1	最小二乗法 (3.2で説明した計算法)	上に同じ

*隣接する候補点間の距離が1画素以内に収まるような間隔で候補点を設定.

は固定して更新しない, fit1は原始的に最寄り点を探して更新する, fitT0, fitT1は勾配法を用いて徐々に更新するという違いである. なお, ベジエパラメータ $\{t_k\}$ の初期値は区間[0,1]を均等分割した値とし, 制御点の初期値はfit0を1度実行して求めた値とする. fitT0は②③ともに勾配法を用いるが, ②の更新 mloop 回につき, ③の更新を1回実行する. 本稿の実験では, 経験的に mloop=3とした.

実装に用いたプログラミング言語はpythonであり, numpy, sympy, Tensorflowなどのライブラリを用いた. 最小二乗法の実装には数値解析ライブラリ numpy の線形連立方程式のソルバー linalg.solve()関数を用いた. fitT0, fitT1で用いる勾配法は機械学習ライブラリ Tensorflow を用いて実装し, 勾配法の最適化アルゴリズムとしては, さまざまな機械学習問題で広く使われ, 定評を得ている Adam 法を採用した.

Tensorflow には Adam 以外にも多くのオプティマイザが用意されている. 本質ではないので詳細は割愛するが, そのうち SGD, RMSprop, Adam, Adamax, Adagrad, Adadelta, Nadam, Ftrl を用い, さまざまな条件で比較実験した結果, この問題には Adam オプティマイザが安定的に優秀であることがわかっている. 比較実験の結果の一例を図7に示す.

Adam オプティマイザにはハイパーパラメータがたくさんあり, 処理時間を左右する. 本研究ではもっとも重要なハイパーパラメータである学習率 learning_rate のみをカスタマイズし, 他のパラメータはデフォルト値で実験を行う. 学習率は, ベジエパラメータの最適化では0.001, 制御点の最適化では30と定めた. この値は, ハイパーパラメータの最適化ツール optuna を用い, 実装法 fitT0で, 後述する実験対象画像のいずれにおいても解が得られ, かつ, 平均的に処理ステップ数が少なくなるような値を試行錯誤的に探して決めた値である. あくまでも実験してみた範囲であるが, 収束可能な学習率の範囲はfitT0よ

り fitT1の方が広く、fitT0が収束する値なら fitT1も収束することを確認している。
実験環境は以下に示すとおりである。

CPU : ThinkCentre M75q Gen 2 (タイプ 11JJ)

AMD Ryzen 7 PRO 4750GE 3.10 GHz (8 core)

RAM 16GB オンボード GPU

Python 環境 : Python 3.9(Anaconda) + Tensorflow 2.6, numpy 1.21.2, sympy 1.9

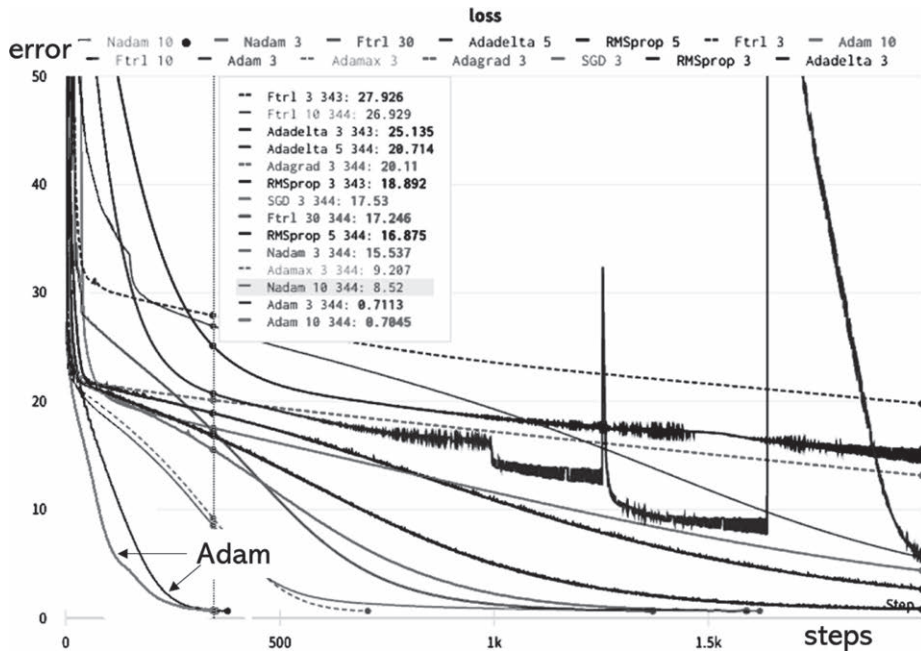


図7 最適化アルゴリズムの比較結果例
(横軸はステップ数、縦軸は残差平方平均)

3.5 オーバフィッティング対策とオーバフィッティングの判定

① オーバフィッティング対策

一般に曲線の自由度が必要以上に高いとオーバフィッティングが発生しやすい。点列への曲線あてはめでは標本点間に対しては何も制約がないため、標本点にはフィットするが標本点間で曲線とシルエットが大きくずれるという可能性は常にある。オーバフィッティングの例を図8に示す。

オーバフィッティング対策としては、標本点を増やす方法と曲線の自由度を下げるようなペナルティ項を目的関数に追加する方法が考えられる。標本点を増やすのは単純で効果的であるが、計算量とのトレードオフとなる。経験上もっともずれの発生しやすいのは両端部であることがわかっている。アドホックではあるが両端とその隣接標本点の中間点を

フィッティング対象に加えるとオーバフィッティングはかなり抑制できる。

標本点間での無用な湾曲を抑制するために、標本点における2次微係数の平方や接線方向と標本点間を結ぶ方向の不一致度などをペナルティ項として追加するのも一定の効果がある。ただし評価項が増えると項の重みの最適化という新しい問題が発生する。

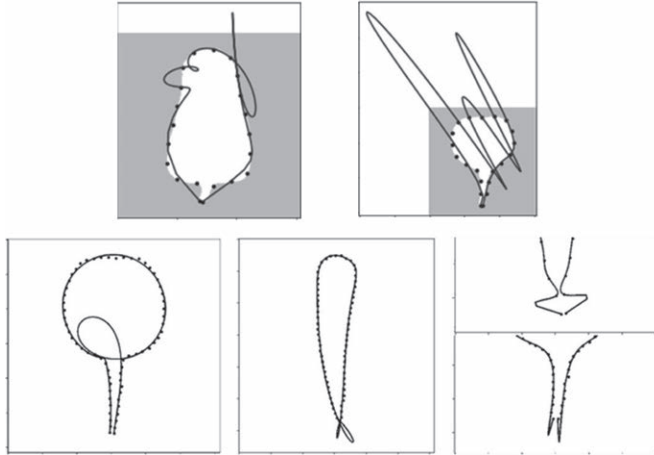


図8 オーバフィッティングの例

② オーバフィッティング判定

オーバフィッティングかどうかは程度の問題であり、判定するとすれば何らかの尺度を設けてしきい値で判定せざるをえない。基本的には近似曲線と輪郭との距離を尺度とすればよいが、どこを測るかが問題である。図9(a)に示すように、すべての輪郭点の近くを通過していても輪郭から逸脱する曲線はありえ、このようなケースは輪郭点列側からの最短距離だけを測っても見逃されてしまう。このケースは、近似曲線の方を十分細かく刻み、経路全体にわたって輪郭点までの距離を測ればよいが、距離だけの判定では図9(b)のケースのような順序関係の矛盾を見逃してしまう。このように厳密に考えても、しきい値の設定や近似曲線の刻み間隔次第で判定結果は左右される。

重要なのは、個人差の生じる人の主観的な判断ではなく、何らかの客観的基準を設けることである。本稿の以下の実験では次のような簡易的な判定方法を採用することにした。

- i) 輪郭点列を標本点の位置で分割し、区間ごとの点列を長さ基準で4等分するような位置を求め、5点(区間の始点、終点+3つの4分位点)で代表させる。
- ii) 近似曲線をi)で決めた区間に対応する区分曲線に分割し、i)同様に4等分し、5点で代表させる。
- iii) 区間ごとに、輪郭点側と近似曲線側の代表点同士の距離を求め、5つの距離の標準偏差があらかじめ設定したしきい値を超える区間が一か所でもあればオーバフィッ

ティングと判定する。

標準偏差では代表点5点すべてが同程度にずれているとオーバーフィッティングと判定されないことになるが、5点のうち2点は標本点であるのでフィッティング条件成立下では距離は小さいはずであり、5点の標準偏差が大きいということは残りの3点でのずれが大きいということになる。判定基準を平均値（誤差の合計としても同じ）や最大値とすることもできる。図8に示したような、目視できらかにオーバーフィッティングが生じていると感じるようなケースはいずれの指標でも検出できる。

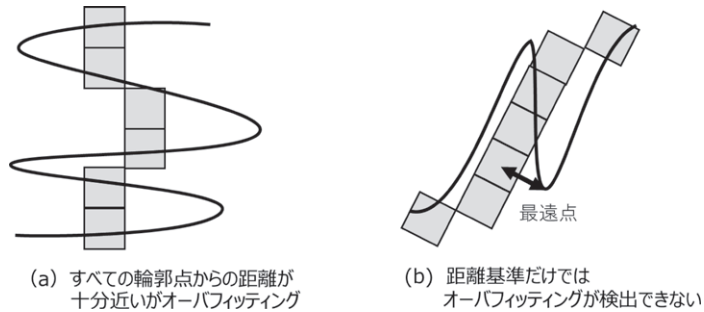


図9 オーバフィッティング判定の難しさ

4. ベジエ曲線によるダイコンのシルエット形状記述

実際のダイコンのシルエット全体、あるいは、左右に分割された輪郭線のそれぞれの記述に必要なベジエ曲線の次数や標本点の数、実装法の優劣などを実験的に調べていく。

4.1 フィッティングの評価尺度

近似曲線の次数を上げていけば画素レベルの微細な凹凸にまでフィットさせることができるが、ひげ根や不自然な切断による凸凹などのノイズにまでフィットさせるのは意味がない。欲しいのは、ノイズは無視して、個体のシルエットに自然な形で沿う滑らかな曲線である。ただそのような抽象的な要求を評価するのは困難であるので、ここでは3章で説明した曲線あてはめ法におけるあてはめ誤差、すなわち標本点と近似曲線上の対応点の距離の平方の平均値をフィッティングの尺度とする。

曲線あてはめの結果に影響を与える項目としては、

- 対象画像（画像サイズとシルエットのサイズ）
- 勾配法の収束判定基準と最適化アルゴリズム
- 標本点の数
- ベジエ曲線の次数
- 許容誤差

●実装法とその設定（最小二乗法／勾配法，オーバフィッティング対策）

などが挙げられる．本報告では，対象とするシルエット，画像のサイズと個体のサイズ，勾配法の収束判定基準，実装法の設定は以下で述べるように固定し，標本点の数，ベジェ曲線の次数，許容誤差，実装法の違いで結果がどう変化するかを調査することにする．

① 対象とするシルエット

図10に示す9個のシルエットを実験対象とする．これらは2017年に兵庫県加西市にある神戸大学大学院農学研究科附属食資源教育研究センターの圃場にて栽培し，それぞれの適期に収穫後に撮影した実写真をもとに，雑音除去・平滑化などの画像処理を施して得られたシルエットである．供試された品種は，「だるま大根」（中原採種場），「平安すしらず聖護院」（タキイ種苗），「方領」（トーホク），「カザフ辛味大根」（そ生研），「マコトちゃん」（トーホク），「ねずみ」（日本タネセンター），「おふくろ」（タキイ種苗），「夏つかさ」（トーホク）である．「ねずみ」については，当センター内の粘質の土壌の影響から品種本来の形に栽培できていない可能性があったため，栃木県宇都宮市にある株式会社トーホクの清原育種農場で栽培，収穫したものも使用した．これらのダイコンを9つのカテゴリー：逆三角形，横楕円形，角（つ）の形，三角形，丸形，円筒形，だるま形，楕円形，先流れに分類した．カテゴリーは，ダイコンの根形を網羅するように，日本国内で使用されている「農林水産省 農林水産植物種類別審査基準」^[7]，「農業生物資源遺伝バンク 特性評価マニュアル」^[8]，および国際的な基準である「IBPGR descriptors for Brassica

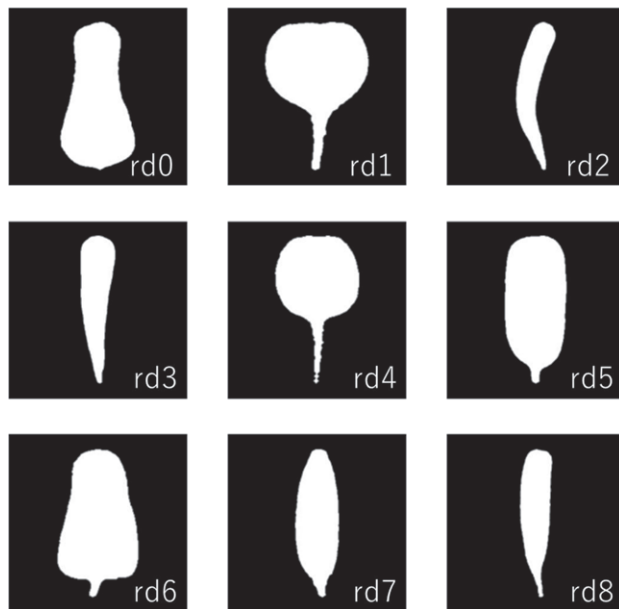


図10 実験対象とするダイコンのシルエット画像

and Raphanus」^[9], 「UPOV Guidelines for the conduct of tests for distinctness, uniformity and stability. Radish; Black radish」^[10,11]に記載されている根形の categorie を参考に設定した。

② 画像のサイズと個体のサイズ

シルエットのバウンディングボックスの長径が256となるようにリサイズし、308×308画素のキャンバス中央に配置した白黒画像を処理対象画像とする。この画像から画像処理ライブラリ OpenCV の findContour() 関数を用いて得られる8連結の輪郭点列が曲線あてはめの対象である。この輪郭を末端部の曲率最大点で開いて開曲線化したものを全周輪郭、それをさらに上端部で2分割したものを左右分割輪郭とよぶことにする。

③ 収束判定基準

- i) あてはめ誤差の平方平均（以下「あてはめ誤差」と省略する）がしきい値 err_th 未満
- ii) あてはめ誤差の減少量の移動平均（直近3回分）が、あてはめ誤差としきい値 err_th の差の1/100,000未満
- iii) 1,000回連続で誤差の最小値が更新されない
- iv) 繰り返し回数が3,000回を超えた

のいずれかの基準が満たされた場合に繰り返しを終了する。いずれのケースでも処理過程であてはめ誤差がもっとも小さかった時点の曲線をあてはめ結果とする。

④ 実装法の設定

実装法のうち、勾配法の最適化アルゴリズムは上述したように Adam 法を用い、学習率は統一する。オーバーフィッティング対策としては、両端部の標本点の中間点計2点を対象に加える方法を fit0 に適用する。どの実装法でも制御点の初期値の決定には fit0 を使っているため、結果的に全実装法にオーバーフィッティング対策を施したことになる。

4.2 全周輪郭のベジエ近似

シルエット全周を1つのベジエ曲線で近似することを考える。ダイコンのシルエットの輪郭は1つの閉曲線となるが、ベジエ曲線は開曲線の記述であるので、輪郭をどこかで切断して開曲線化する必要がある。ここでは根の先端部を切断点として採用することにする。主題から外れるので説明は省略するが、ほとんどの個体で、下部先端部は曲率に基づき自動抽出可能である。

ここでは、まず、近似次数ごとのあてはめ結果の違いと誤差の程度を調べるための基礎実験を行った後、各シルエットの全周を近似するのに必要な次数、標本点数の違いによる近似結果の違い、オーバーフィッティング対策の効果について調べる。

① 実験 A 基礎実験

4つの実装法それぞれで次数をさまざまに変え、図10に示した9つのシルエットの全周輪郭のベジエ近似を試みた。4つの実装はいずれも正しく機能し、近似曲線が得られることが確認できた。標本点数65で次数を変化させていった場合のあてはめ結果の違いの例を図11に示す。図11に示した結果は実装法 fitT0 による結果であるが、fit1, fitT1 を用いても目に見える大きな違いはない。

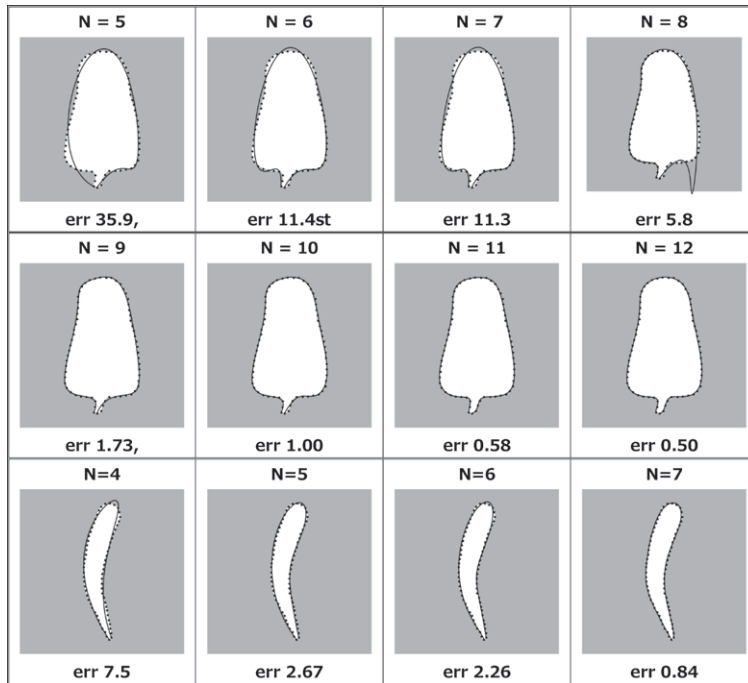


図11 近似次数による近似結果の違い (N : 近似次数)

② 実験 B 全周近似に必要な次数

ダイコンのシルエットをベジエ曲線で近似するのに必要な次数と実装法の優劣を調査するため、標本点数を65、許容誤差（あてはめ誤差の許容値）を1.0, 0.65, 0.5として、実装法 fit0, fit1, fitT0, fitT1 のそれぞれで、許容誤差内かつオーバフィッティングとならない最低の近似次数を調べてみた。

なお、オーバフィッティングの判定基準は上述のとおりであり、オーバフィッティング判定のしきい値は、「標本点間の平均経路長×あてはめ誤差のしきい値」とした。

i) 必要次数の比較

実験結果を表2に示す。網掛けは同じ対象に対してもっとも次数の低いものである。近

表2 全周近似に必要な次数

err_th = 1.0				
	fit0	fit1	fitT0	fitT1
rd0	17	7	7	7
rd1	18	10	10	9
rd2	20	8	9	7
rd3	16	8	5	5
rd4	21	10	12<11>	10
rd5	15	8	9	8
rd6	17	10	12<11>	10
rd7	19	6	6	6
rd8	18	8	9	9<8>

err_th = 0.65				
	fit0	fit1	fitT0	fitT1
rd0	23	7	7	7
rd1	24	10	12	10
rd2	24	10	9	8
rd3	18	9	6	10<9>
rd4	27	11	12	10
rd5	19	9	9	9
rd6	21	11	12	11(10)
rd7	26	8	7	6
rd8	22	9	10	9<8>

err_th = 0.5				
	fit0	fit1	fitT0	fitT1
rd0	28	9	8	8
rd1	>30[0.51]	15	15	10
rd2	26	11	9	11
rd3	19	10	7	10<9>
rd4	>30[0.60]	13	12	10
rd5	24	11	9	9
rd6	27	14	12	12
rd7	>30[0.54]	11	8	7
rd8	25	12	10	9

似次数の上限は30までとし、30次でもあてはめ誤差が許容誤差以下にできなかった場合は“>30 [数値]”と記してある。数値は30次で収束した時点のあてはめ誤差である。<次数>と(次数)はオーバフィッティング判定を省いた場合の最低次数である。<次数>と(次数)の違いは、目視によりオーバフィッティングが顕著であると感じるかそうでない

かの違いである。これは個人の主観に左右されるのであくまでも参考情報と考えられたい。

fit0は他の実装と比べて近似に必要な次数が2倍程度以上となる。この表の数値には現れないことであるが、標本点数65に対して次数が30にもなると、標本点のぶれに忠実にフィットしすぎて曲線全体の滑らかさが失われてくる。

許容誤差内でより少ない次数でフィッティングできることを実装法の優劣の尺度とするならば、fit1, fitT0, fitT1のうちのいずれかの実装法が常に他よりすぐれているということとはなかった。処理対象と許容誤差により優劣は入れ替わる。fit1の場合、対象rd0～rd8の全周近似に必要な次数の範囲は、許容誤差1.0, 0.65, 0.5に対して、6～10次、7～11次、9～15次であった。同じくfitT0の場合は、5～12次、6～12次、7～15次、fitT1の場合は、5～10次、6～11次、7～12次であった。許容誤差1.0では3つの実装に大きな違いはないが、許容誤差が0.5になるとfit1は他の2つの実装より1ないし2次多い次数が必要となっている。

なお、同じ次数同じ実装法であれば、許容誤差1.0, 0.65, 0.5の結果の違いはわずかであり、目視ではほとんど違いがわからない。近似次数や実装法が違ってこの許容誤差の範囲では違いはわずかであり、変曲点や端点付近をよく見比べると違いがわかる、という程度である。

最後に、オーバフィッティングに関してコメントしておく。この実験の範囲では、fit0はオーバフィッティングが生じにくく、もっともオーバフィッティングが多く生じたのはfitT0であった。fitT0はベジエパラメータと制御点の両方に勾配法を適用するため、ローカルミニマにトラップされる可能性が組み合わせて増えるためであろう。また、今回の実験の範囲内でひどいオーバフィッティングが生じたのは、許容誤差以下で近似できる最低次数ちょうどの場合であった。もちろん標本数もオーバフィッティングの発生に大きく影響する。今回の実験では標本数を65と多めに設定したが、標本数をもっと減らすとオーバフィッティングが発生しやすくなる。

ii) 処理時間の比較

次に処理時間に焦点を当てるため、許容誤差を1.0として、近似に必要なベジエ曲線の次数とその結果を得るまでの繰り返しのステップ数および処理時間を調べた。標本点数は先と同じ65とした。結果を表3に示す。なお、処理時間は処理環境や使用しているライブラリ関数に大きく左右される。実験対象の4つの実装法はそれぞれ依存しているライブラリ関数が異なるので、実装法間での処理時間の絶対値の比較は意味がない。また、学習率などの設定を変えると結果は大きく変わる。

数値以外の記号の意味はつぎのとおりである。

UF あてはめ誤差が1.0未満とならずに処理打ち切りとなった。カッコ内は誤差の最小値。

OF あてはめ誤差1.0未満となって停止したがオーバフィッティングが生じている。

表3 あてはめ誤差が1.0未満となる次数と処理時間およびステップ数

	fit0		次数	fit1	fitT0	fitT1
	次数	処理時間		処理時間 / ステップ数	処理時間 / ステップ数	処理時間 / ステップ数
rd0	17	2.9s	7	210.0s / 236	59.9s / 455	82.1s / 161
rd1	18	3.3s	9	UF(5.60)	UF(3.03)	615.7s / 739
			10	521.3s / 334	327.8s / 1621	167.1s / 162
rd2	20	3.9s	7	UF(1.11)	UF(1.33)	343.1s / 675
			8	1867.7s / 1722	UF(1.16)	832.2s / 1279
			9	226.6s / 166	226.8s / 1260	45.0s / 50
rd3	16	2.5s	5	UF(1.46)	50.9s / 555	137.1s / 525
			6	UF(1.18)	92.5s / 818	97.6s / 261
			7	UF(1.18)	111.3s / 826	175.3s / 347
			8	371.0s / 337	OF(138.0s/895)	OF(49.0s/44)
rd4	21	4.4s	10	2202.1s / 1395	UF(2.05)	674.1s / 658
			11	1504.4s / 843	OF(653.6s/2373)	618.4s / 498
			12	122.4s / 57	152.8s / 573	30.7s / 18
rd5	15	2.3s	8	629.5s / 581	UF(2.88)	160.2s / 243
			9	281.0s / 213	340.0s / 1875	87.2s / 101
rd6	17	2.9s	10	3254.5s / 2080	UF(1.14)	2808.3 / 2781
			11	333.2s / 183	UF(1.02)	72.9s / 56
			12	181.3s / 85	72.0s / 268	61.0s / 39
rd7	19	3.6s	6	111.0s / 160	30.0s / 259	31.1s / 79
rd8	18	3.2s	8	670.8s / 613	OF(137.0s/894)	OF(1511.0s/2313)
			9	613.0s / 463	102s / 559	184.9s / 220

UF：許容誤差内で収束せず。カッコ内は誤差の最小値，OF：オーバフィッティング

fit0は近似に高い次数を必要とするが直接解法であるため非常に高速である。fit0以外の3実装法は近似曲線の初期値を決めるのにfit0を用いているためfit0より処理時間がかかるのは当然なのであるが、勾配法の反復計算の計算量は非常に大きく、また、時間幅が広い（本実験の環境で30秒～1時間弱）。反復計算1回分の計算量は次数が高いほど大きくなるはずであるが、同じ誤差に到達するまでに必要な反復回数は次数が高いほど少なくて済む傾向がある。結果として次数が高い方が処理時間は短傾される傾向が見て取れる。

fit1とfitT1の結果を比較すると、両方でステップ数当たりの処理時間に大きな違いはないが、同対象、同次数で比較すると必要なステップ数がfit1よりfitT1の方が少なくて済んでいる。fit1とfitT1は制御点の最適化に最小二乗法を用いるという点で共通しており、違いはベジエパラメータの最適化法である。fit1のベジエパラメータの最適化フェーズでは、その時点での近似曲線に対して個々の標本点の最寄り点を求め、標本点に対応付けるパラメータ値を最寄り点のそれに書き換える。この方法はいきなり最適解に到達できる可能性

はあるが、時間軸にそって見たときに値の連続性がないため振動を伴いやすい。一方 fitT1 の勾配法は、パラメータの値をもとの値を基準として最適と思われる方向に徐々に変化させていく。そうしても多かれ少なかれ振動は発生するが、fitT1 で採用している Adam オプティマイザには振動を抑える仕組みが組み込まれており、とても優秀である。結果として fitT1 の方が fit1 より少ないステップ数で収束できているものと推測される。

fitT0 と他の 2 つの実装法を比べると、fitT0 のステップ数は総じて大きな値となっており、処理時間は fitT1 と同程度、fit1 と比べると短めである。fitT0 と fitT1 はベジエパラメータの最適化法は同じであり、違いは制御点の求め方である。fitT1 が最小二乗法で制御点を求めるのに対し、fitT0 は勾配法で徐々に制御点を変化させていく。この構図は先のベジエパラメータの最適化における fit1 と fitT1 の関係と同じであるが結果は逆で、勾配法を使う fitT0 の方がステップ数は多くなっている。勾配法における収束までのステップ数は学習率に左右されるので学習率の設定を変えれば fitT0 のステップ数を抑え、処理時間を短縮できる可能性はある。

以上の結果と、fitT1 に特化した値を使えば、fitT1 の結果はさらに改善される（先に述べたが、この実験では比較のため fitT0 に合わせた同一の学習率を用いた）ことを考えると、今回比較した実装の中では fitT1 が総合的に優れていると言えよう。

許容誤差 $err_th=1.0$ 、近似次数 11、実装法 fitT1 の場合の近似結果を図 12 に示す。他の実装法による結果との違いはわずかである。

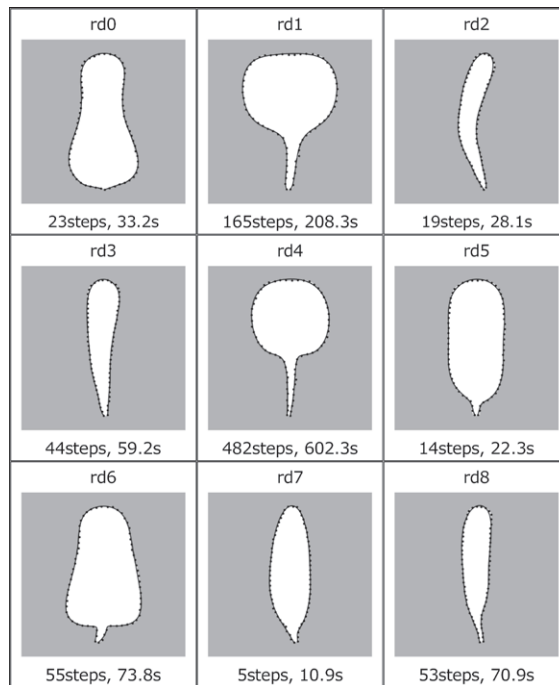


図 12 全周近似結果 (fitT1, 標本数 65, 11 次近似, $err_th=1.0$)

③ 実験 C 標本点数の違いによる近似結果の違い

ここまでの実験では標本数を65と多めに設定したが、標本数を変化させた場合のあてはめ結果の違いを調べる。近似次数を11, 許容誤差 err_th を1.0に統一し、標本数を21, 25, 33, 41と変化させて、fitT1で曲線あてはめを試みた。結果を図13~15に示す。

輪郭の曲率変化が緩やかな個体であれば標本数が21でもフィッティング結果は良好である。比較的複雑な形状の個体 rd1, rd4, rd6 は標本数21では変曲点付近でずれが目立つ。rd1では輪郭としてありえない曲線になってしまっており、標本数は21では少なすぎといえる。標本数25では rd4, rd6 の末端部で少し不自然な箇所が見られる。標本数33以上では、輪郭と近似曲線のずれはわずかであり、ダイコンの形状記述としては必要十分であると考えられる。

標本数の増加は当然計算量の増加につながる。標本数を横軸、処理時間を縦軸に取ったグラフを図16に示す。図13~図15には示していないが、図16には標本数29, 51, 65の結果も含まれる。

繰り返し最適化というアルゴリズムの性質上、計算量は標本数の単純な関数ではない。処理時間はおおむね繰り返し回数と標本数の積に比例するはずである。グラフを見る限り、標本数21の結果を除けば、グラフはおおむね緩やかな右肩上がりとなっている。上述のように標本数をむやみに大きくしても近似結果にはあまり差はないので全周近似の標本数は30~40程度が適当だといえる。

④ 実験 D オーバフィッティング対策の効果

ここまでの実験結果はすべて、両端の標本間に1点ずつ、計2点標本点を追加するという簡易的なオーバフィッティング対策を施した上での結果である。これは対策なしでは端部付近の曲線が不自然になるという現象が頻発することが経験上わかっているからであるが、実際対策がどの程度効果があるのかを確認するために、対策を施さなかった場合と施した場合を比較しておく。

同じ個体に対してオーバフィッティングが生じるかどうかは標本数と近似次数に左右される。組み合わせをすべて試すのは実験に時間がかかりすぎるので、標本数は65とし、10, 11, 12次近似。許容誤差1.0という条件でオーバフィッティング対策なしの場合とありの場合を比べてみた。

結果を表4に示す。数字は3.5の②で述べた、簡易的なオーバフィッティングの判定基準により、オーバフィッティングと判定された区間の番号である。標本数が65であるので区間は64区間、区間番号は0~63である。x印は当該次数では許容誤差内のフィッティングができなかったことを意味する。その場合は、許容誤差の設定がその結果の最小誤差と等しかったと仮定して判定している。

この実験により次のことがわかった。

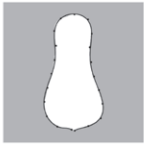


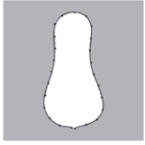


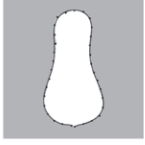


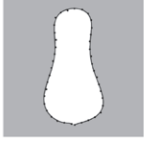


標本数	rd0	rd1	rd2
21	 25steps, 14.8s	 >3000steps, 1509.2s	 9steps, 6.7s
25	 24steps, 16.0s	 233steps, 134.7s	 16steps, 11.5s
33	 26steps, 21.3s	 236steps, 167.8s	 20steps, 16.8s
41	 30steps, 28.6s	 156steps, 133.1s	 22steps, 21.8s

図13 標本数によるフィッティング結果の違い (rd0, rd1, rd2)



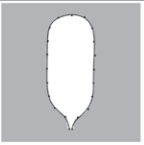
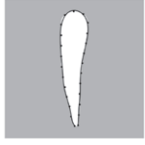
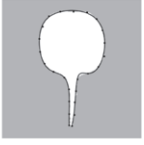
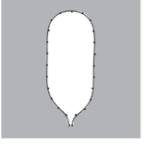

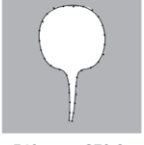


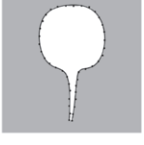

標本数	rd3	rd4	rd5
21	 26steps, 15.4s	 1381steps, 709.5s	 45steps, 24.8s
25	 30steps, 19.5s	 672steps, 383.0s	 29steps, 18.9s
33	 41steps, 31.9s	 540steps, 379.8s	 15steps, 13.9s
41	 46steps, 41.8s	 575steps, 480.0s	 25steps, 24.5s

図14 標本数によるフィッティング結果の違い (rd3, rd4, rd5)

ベジェ曲線によるダイコンの形状記述とその応用



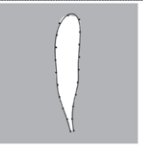


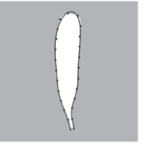
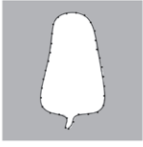
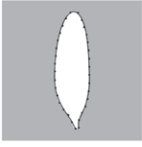
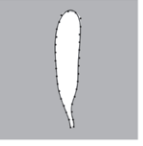
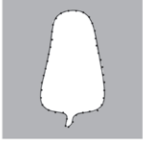
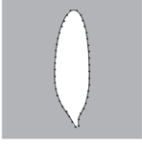
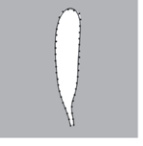
標本数	rd6	rd7	rd8
21	 92steps, 48.3s	 11steps, 7.7s	 14steps, 9.5s
25	 94steps, 55.8s	 3steps, 4.2s	 29steps, 19.0s
33	 57steps, 42.6s	 5steps, 6.5s	 33steps, 26.4s
41	 86steps, 74.8s	 4steps, 7.1s	 75steps, 66.3s

図15 標本数によるフィッティング結果の違い (rd6, rd7, rd8)

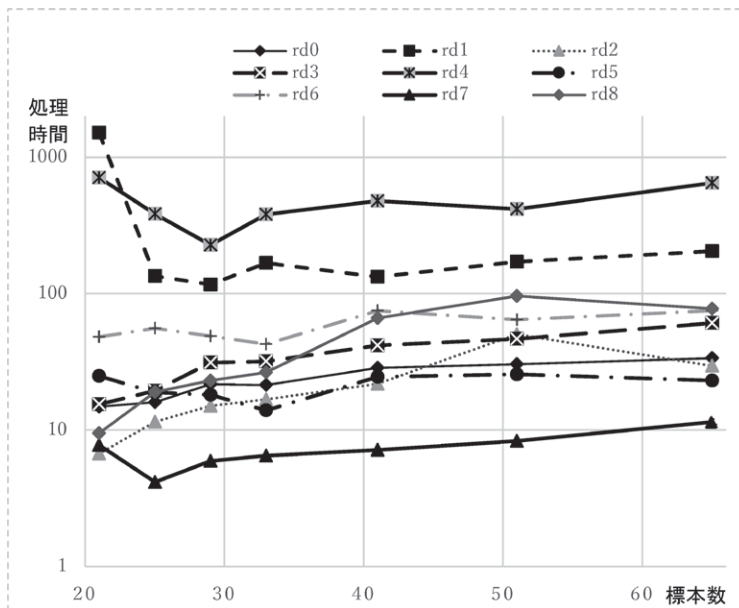


図16 標本数と処理時間の関係

- rd2, rd5, rd7はオーバーフィッティングを起こしにくい。対策を施さなくても、10~12次の範囲でいずれの実装でも自動で検出されるような大きなオーバーフィッティングは発生していない。
- fit1, fitT1は12次近似ならばオーバーフィッティング対策を施さなくても、自動検出されるような大きなオーバーフィッティングは発生していない。
- オーバーフィッティングは端点部と変曲点付近に発生することが多い。端点部のオーバーフィッティングはfd0, rd1, rd3, rd4, rd8で、変曲部のオーバーフィッティングはrd4, rd6で発生することがある。
- 端点部のオーバーフィッティング対策は効果的である。fit1, fitT1では端点部のオーバーフィッティングが見られなくなり、fitT0では程度を抑えることができている。
- 端点部のオーバーフィッティング対策が間接的に変曲点付近のオーバーフィッティングを抑制することもある。

表4 端部オーバーフィッティング対策の効果 (許容誤差1.0, 標本数65)

オーバーフィッティング対策なし (数字はずれの大きい区間の番号, xは誤差1.0以上)									
	fit1			fitT0			fitT1		
対象	10	11	12	10	11	12	10	11	12
rd0	0, 63						0, 63		
rd1	0, 63	0, 63					0, 63	0, 63	
rd2									
rd3	63			x 0, 63	0	0	63		
rd4	x 0, 63	0		x 52	52		x 0, 63	0, 63	
rd5									
rd6				x	x 2				
rd7									
rd8	63						63		

オーバーフィッティング対策あり									
	fit1			fitT0			fitT1		
対象	10	11	12	10	11	12	10	11	12
rd0									
rd1									
rd2									
rd3				0, 63	0	0			
rd4				x	x 52				
rd5									
rd6				x	x 2				
rd7									
rd8									

図17にオーバーフィッティングが抑制された結果の例を示す。左二つは端点部にオーバーフィッティングが発生する例、右二つは変曲点付近にオーバーフィッティングが発生する例である。

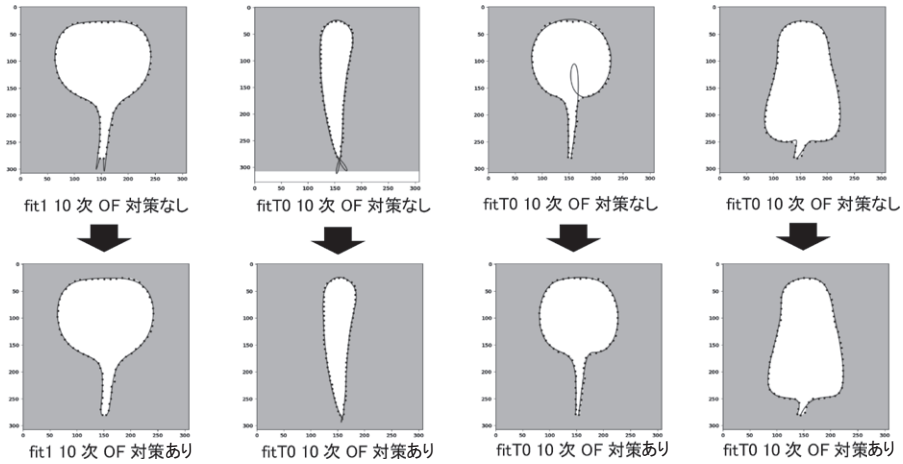


図17 オーバーフィッティング対策の効果

4.3 輪郭の分割近似

図3では一つのシルエットを4分割し、4本の3次ベジエ曲線を接続することで1つの輪郭を表現するという例を示した。単に形状を記述すればよいだけであればどう分割しても記述は可能であるが、記述同士の類似性や差異を解析したり、その結果を分類に利用したりするには、自動であれ手動であれ、一意性のある分割法が必要になる。

ダイコンの形状を記述すると同時に計測を可能にするという観点からは、図18に示すような中心軸を基準とするような分割が理想的である。

論点が発散するし個体によって成否が左右されるので詳細説明は割愛するが、下端の分割点としては、先の全周近似で述べたように曲率最大点を候補点とし、上端の分割点としては局所的な対称性が極大化される点を候補として算出するプログラムを作成した。本稿で実験対象としている9つの個体では、妥当な分割位置が自動判定できている。また候補点が人の判断する分割位置とずれている場合に手動で分割位置を修正指定するようなツールも作成してある。以下の実験では、上部下部ともに自動判定された分割候補点から半径3画素以内の輪郭点を取り除き、左右2本の輪郭線点列とした。

3.1でダイコンの輪郭は2分割したとしても曲線上の座標値の成分同士が一意対応となるとは限らないと述べたが、図18の2つの形状はその例である。これらの輪郭は中心軸の上下端が水平になるように回転させたとしても、元の図形の上部位置で曲線がオーバハンクし、同じ x 座標を持つ点が複数できてしまう。そのような曲線は多項式近似できない。

また、下部で同様のことが起こる個体もある。ベジエ曲線による近似はそのようなケースにも適用可能である。

全周輪郭を対象とした実験により、必要な標本点数、オーバフィッティング対策、実装法の優劣については一定の知見が得られたので、左右分割した輪郭の近似については近似に必要な次数のみを調べる。

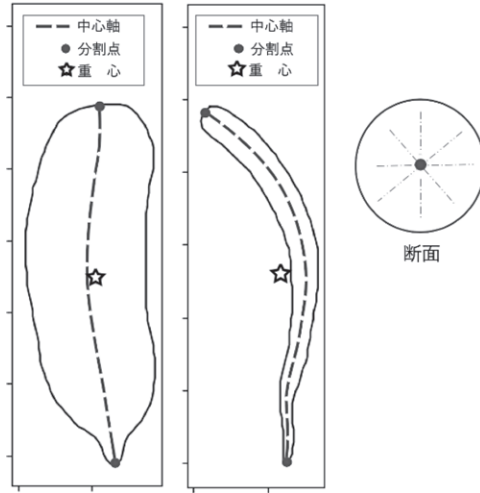


図18 中心軸を基準とした輪郭の分割

実験 E 左右分割輪郭の近似に必要な次数

標本点の数を22とし、実装法は fitT1 を用いて、あてはめ誤差1.0未満を達成できる近似次数を調べてみた。結果を表5に示す。表からわかるようにシルエットを左右に分割して別々に扱い、あてはめ誤差の許容基準1.0で近似する場合、4～6次のベジエ曲線で近似できる。

ダイコンの形状の中でもっとも複雑なのは rd0 や rd6 のように中くびれのある形状であろう。直感的にはそれらは4次の曲線で近似できそうであるが、結果は rd0 が予想通り4次であるのに対し、rd6 は6次となった。違いは rd0 には尻尾状の部分がなく、rd6 にはあるという違いである。他の個体についても、尻尾状の部分を近似対象から省いてよいのであれば次数を1ないし2減らすことができると思われる。しかしながら、形状記述前で中心軸も不明な時点で、不要部分を自動判定するのは困難である。また、先行研究[5,6]では、肥大部下部の幅の減衰曲線を品種の特徴の一つとして挙げており、むやみに先端部をカットするわけにもいかない。

近似次数を6次に統一し、err_th = 1.0とした場合の近似の結果を図19に示す。

rd5 と rd6 の下部の先端に輪郭とのずれが少し見られるが、下部はひげ根の処理の仕方によって形状がさまざまに変化するため、それらの部分を忠実にトレースするのは無意味であ

る。また上部も葉部との切断によって形状が不連続となりがちである。本稿の実験では上下の除外対象を小さめに設定したが、形状解析を目的とする場合は切断面や下部尖端部で形状が不規則な部位は対象から除外すべきである。

表5 左右分割近似に必要な最低次数

	左側	右側
rd0	4	4
rd1	4	5
rd2	4	4
rd3	4	4
rd4	5	6
rd5	5	5
rd6	6	6
rd7	4	4
rd8	4	5

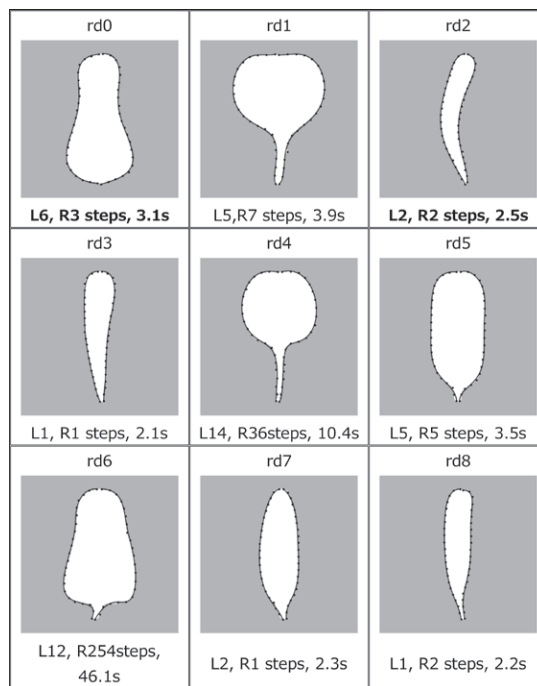


図19 左右分割輪郭の近似結果 (fitT1, 標本数22, 6次近似, err_th=1.0)

5. 自動計測への応用

本稿で提案したベジエ曲線による形状記述を形状分析や自動計測に応用することを考える。

5.1 中心軸の推定

砲弾型の典型的なダイコンをイメージすると、「長さ」や「幅（径）」という属性は自明で疑問の余地がないように思われるが、個々の個体は多かれ少なかれ曲がりがあり、角型のように曲がっていることが特徴となっている品種さえ存在する。何をもち「長さ」と定義するのは自明ではない。「幅（径）」についても同様である。「長さ」や「幅」という概念は中心軸の存在を暗黙の裡に前提としており、「長さ」とは中心軸の長さ、「幅」とは中心軸に対する垂直断面の径と考えるのが自然である。人手で計測する際には、頭の中でイメージする中心軸に沿った長さや幅を計測することになるが、それでは個人差が避けられない。

ここでは4.3で説明した左右分割のベジエ近似曲線を用いて中心軸を推定する方法を示す。

自動計測が目的であるので、得られる軸がダイコンの真の中心軸に一致するかどうかはあまり重要ではない。人がイメージするのに近い軸が機械的に得られるということ、よって計測結果に個人差がでない、ということに意味がある。

中心軸の推定は、①まず、左右の輪郭の平均曲線を求め、②その曲線上に等間隔に取った標本点に低い次数のベジエ曲線を当てはめて中心軸と見なす、という手順で行う。

① 左右輪郭の平均曲線

左右の近似ベジエ関数をそれぞれ $B_L(t)$, $B_R(t)$ とし, $B_c(t) = (B_L(t) + B_R(t))/2$ と定義する。rd0～rd8について $B_L(t)$, $B_R(t)$, $B_c(t)$ を描いたものを図20に示す。 $B_L(t)$, $B_R(t)$ のパラメータ t は互いに独立であり, $B_c(t)$ は単に $B_L(t)$, $B_R(t)$ の同一パラメータ値の点同

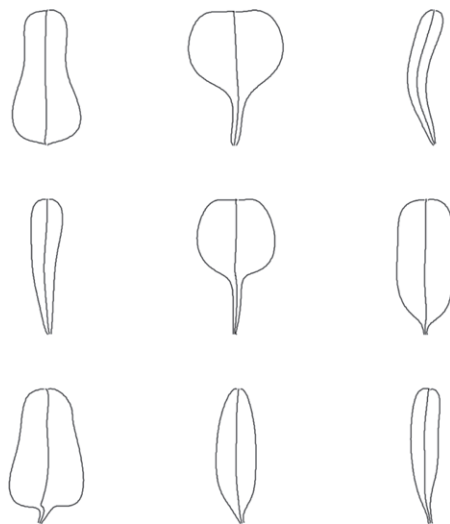


図20 左右の輪郭の平均曲線

土の中心の軌跡を表すに過ぎないが、 $B_c(t)$ はおおむね中心軸として違和感のない位置を通ることがわかる。ただし、例えば rd6（左下）の下部のように、左右の輪郭形状の曲率が大きく変化する場所では不自然になりがちである。

② 再近次による中心軸の推定

$B_c(t)$ の次数は $B_L(t)$, $B_R(t)$ の次数と同一になるが、中心軸が複雑な凹凸をもつのは不自然である。S字状や弓状に湾曲した個体まで考慮するとしても高々4次程度の曲線ととらえる方が自然である。そこで、曲線 $B_c(t)$ 上に等間隔になるように標本点を設定し、4次のベジエ曲線をあてはめる。あてはめのアルゴリズムは輪郭線と同一である。結果を図21に示す。

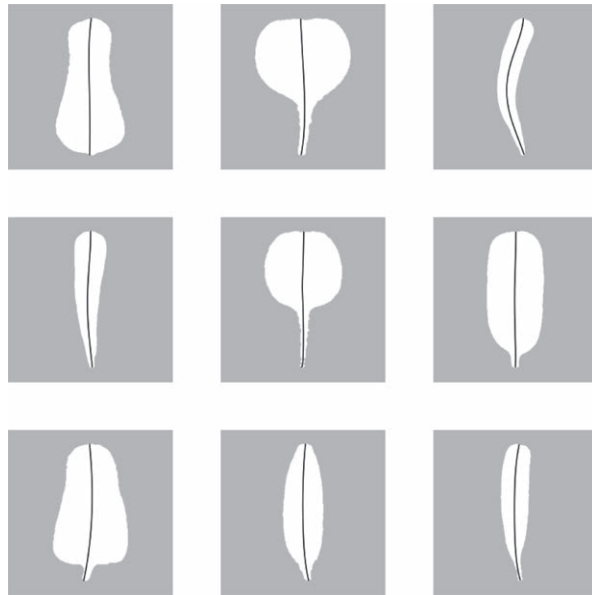


図21 中心軸の推定

5.2 曲がり補正と自動計測

中心軸上の各点における垂線断面がシルエットから切り取る線分の長さを、その点における「幅（径）」、中心軸に沿って測った中心軸の長さを「個体長」と定義する。中心軸の一方の端点を始点として中心軸に沿って測った距離を縦軸、幅を横軸にとってプロットすれば、曲がりになかった場合の形状を合成することができる。

① 長さの自動計測

原画像と同じサイズの画像キャンバスに中心軸を点列として描き、OpenCVの

arcLength()関数でその点列の長さを求めることで、中心軸の長さを近似計算することにする。中心軸がパラメトリック関数として表現されているので、数値積分で長さを求めることも可能である。

② 幅の計測

ベジエ曲線で表現した中心軸が数式として表現されているので、中心軸上の任意の点での垂線の方程式も数式处理的に得られる。理論的には、この直線の方程式と両側のベジエ曲線の方程式を連立させて解けばよい。両側が6次で表現されているならば、6次方程式となり、6つの複素解が得られるが、パラメータ範囲が[0,1]の実数と決まっているので、その範囲の解を取り出せばよい。実際それが可能であることは確かめたが、次数が高いため大変計算時間がかかる。別の解法として、実際に輪郭線と垂線を画像として描くことで交点を求めるという方法が考えられる。これは厳密解とはならないが、原理的に誤差はサブピクセル程度であり、数式処理より圧倒的に高速に計算できる。

③ 曲がり補正形状の合成

中心軸に沿って幅をサンプリングし、幅の半値を座標、中心軸に沿った長さを座標とみなしてベジエ曲線をあてはめることで、曲がりのない場合の輪郭線形状を求めることができる。この際、中心軸上で等間隔になるように幅の標本を採るのは容易であるが、そうすると、図22の(a)のように、輪郭の方向が垂直に近い箇所では標本点は密に、水平に近い箇所ではまばらになってしまう。逆に(b)のように左右の輪郭上に等間隔に標本点を設定して対応する点を結んで幅とするという考え方だと対応点同士を結ぶ直線が中心軸と垂直になるとは限らず、幅の値としてはふさわしくなくなる。ここでは次善の策として、次のようにして幅の標本データを作成することにした。

- i) 左右の輪郭の近似曲線上で等間隔になるように標本点を設定する。

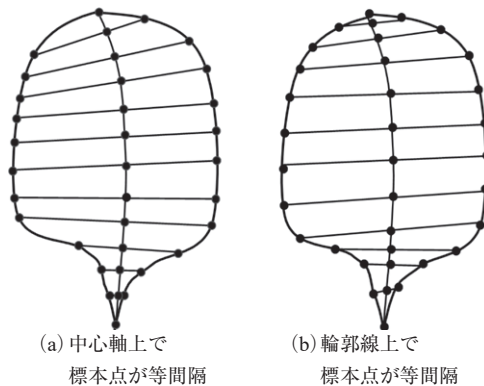


図22 幅の標本化

ii) 左右の対応する標本点を結んでできる直線と中心軸の交点を中心軸上の標本点とする。

iii) 中心軸上の各標本点において中心軸に対する垂線を引き、左右の近似曲線との交点をそれぞれ求め、それらを結ぶ距離を幅の標本値とする。

この手順で定まる中心軸上の各標本点について、中心軸上端から軸に沿って測った距離を y 座標、幅の半値を x 座標として座標列をつくり、これまでと同じようにベジエ曲線をあてはめてやれば、曲がりがない場合の形状が得られる。

標本点数を22、近似の次数を6とした場合の結果を図23に示す。

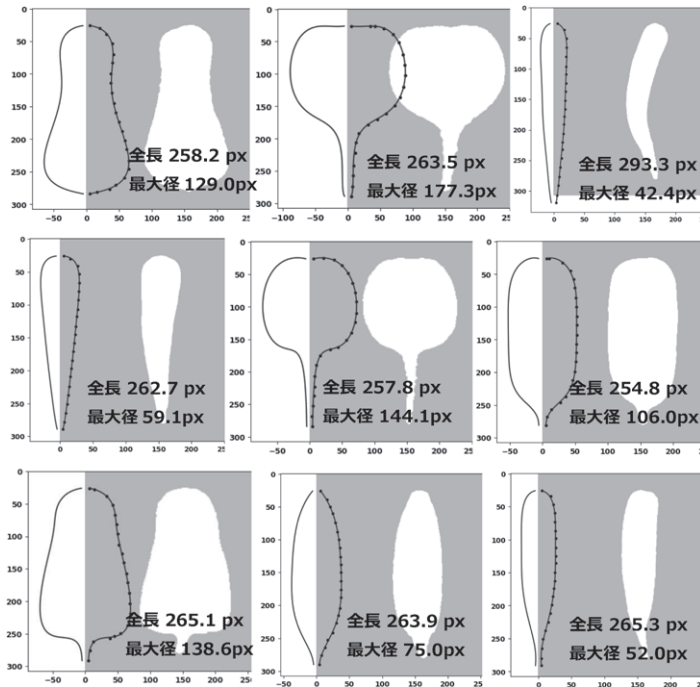


図23 補正形状の生成と自動計測結果

6. おわりに

われわれがイメージする典型的なダイコンのシルエットは左右対称であるので、形状を記述したいなら輪郭の左右どちらかの形状を記述すれば十分なように思われる。また、典型的なダイコンだけをイメージすると、その形状は2次関数が、多めに考えても高々4次関数で近似すれば十分なように思える。しかし、実際の個体の中心軸は直線とは限らず、中心軸の曲がりが特徴となっている品種もある。また輪郭の形状は品種によって多種多様で、ダルマ型のような複雑な輪郭を持つ品種もあるし、個体それぞれは完全な左右対称ではない。本研究では、そのような個性を含めて、個体の形状を少ないパラメータでなるべ

く忠実に記述することを目的としており、今回はベジエ曲線をあてはめるという方法について検討した。

まず、シルエットにベジエ曲線をあてはめるアルゴリズムを示し、輪郭全体の近似に必要な次数と標本数、輪郭を2分割した場合の近似に必要な次数を実験的に示した。もちろん、これらの値はどこまで忠実に形状の凹凸を再現したいかによって変わるが、形状特徴を大まかにとらえて品種の識別を行ったり、個体同士の形状の差異を比較したりすることが目的であるならば、本稿で示した程度（全周で10次、片側で6次）の近似で十分目的を果たせるであろう。また、得られた形状記述により機械的に計測を行う方法も示した。個体の曲がりにも対応でき、計測者ごとの計測値のばらつきをなくすることができるようになる。

本研究を通して、輪郭をベジエ曲線で記述することで様々な応用の可能性が拓けることが分かった。本稿では輪郭線の近似をスタートとして、中心軸、そして補正形状の記述にベジエ近似を利用した。説明を省略したが実はシルエットの分割点を求めるための曲率最大点を見つける際にも内部的にベジエ近似を利用している。その他、接線や垂線を求めたり、交点を計算したりといった、数値的に行くと大変面倒な処理が数式処理として簡単に記述できる点もメリットである。

本稿では曲がりに沿った全長と最大径の自動計測の例を示したが、本手法では中心軸と径の関数表現が得られるので、任意の部位の径や径の変化のパターン、中心軸の曲率やその変化のパターンなどをとらえることが可能である。条件に合う部分の取り出しや削除も容易であり、全体だけでなく部分形状に基づいた解析や品種の分類、識別などに有用である。最初に述べたように、文献[5,6]における評価方式では本報告の形状記述法を利用している。ベジエ近似した記述に基づいて、長幅比、最大径の位置と程度、曲がり特徴量を自動計測し、また、先端部のみの形状を再記述、根形評価に利用している。

参考文献・参考 URL

- [1] Iwata, H., S. Niikura, S. Matsuura, Y. Takano and Y. Ukai : “Evaluation of variation of root shape of Japanese radish (*Raphanus sativus* L.) based on image analysis using elliptic Fourier descriptors”, *Euphytica* 102, pp.143-149 (1998).
- [2] 淡誠一郎 : “根菜の画像解析のための形状記述についての研究ノート—フーリエ記述子による形状記述—”, *大阪学院大学 人文自然論叢*, 79-80, pp.55-67 (2020).
- [3] 淡誠一郎 : “デジタル画像におけるフーリエ記述子の回転不変性の検証”, *大阪学院大学 人文自然論叢*, 79-80, pp.1-20 (2020).
- [4] Iwata, H. and Y. Ukai : “SHAPE: a computer program package for quantitative evaluation of biological shapes based on elliptic Fourier descriptors”, *J. Hered.* 93, pp.384-385 (2002).

- [5] 淡裕美子, 淡誠一郎, 吉田康子: “ダイコンの根形を特徴づける形質の選定とその評価方法の確立”, 育種学研究19 (日本育種学会 第132回講演会要旨集), 別2, p.77 (2017).
- [6] 淡裕美子, 淡誠一郎, 吉田康子: “画像解析を用いたダイコンの根形を特徴づける形質による多様性評価”, 育種学研究21 (日本育種学会 第135回講演会要旨集), 別1, p.101 (2019).
- [7] “農林水産省 農林水産植物種類別審査基準－ダイコン”, <http://www.hinshu2.maff.go.jp/info/sinsakijun/kijun/1528.pdf> (2017).
- [8] “農業生物資源ジーンバンク 特性評価マニュアル”, https://www.gene.affrc.go.jp/manuals-plant_characterization.php (2023).
- [9] IBPGR: “Descriptors for Brassica and Raphanus”, Bioversity International (1990).
- [10] “UPOV Guidelines for the conduct of tests for distinctness, uniformity and stability, Radish”, https://www.upov.int/en/publications/tg-rom/tg064/tg_64_6.pdf (1999).
- [11] “UPOV Guidelines for the conduct of tests for distinctness, uniformity and stability. Black radish”, https://www.upov.int/en/publications/tg-rom/tg063/tg_63_6.pdf (1999).

プログラムの公開

本研究で説明したシルエットのベジエ曲線あてはめを含む形状解析用プログラムをGitHubに公開している. → RadishLab <https://github.com/iciromaco/RadiShLab>

(2023年2月9日受理)