筑波大学大学院博士課程

システム情報工学研究科修士論文

ユーザ定義手書きジェスチャの高速かつ軽量な認識アルゴリズム

山路 大樹

修士（工学）

（コンピュータサイエンス専攻 学位プログラム）

指導教員　志築 文太郎

2017年3月

概要

目 次

第 1 章　序論 **1**
1.1 背景 1
1.2 目的 2
1.3 貢献 4
1.4 本論文の構成 4

第 2 章　関連研究 **5**
2.1 手書きジェスチャ認識アルゴリズム 5
2.2 ユーザ定義に特化した手書きジェスチャ認識アルゴリズム 6
2.3 手書きジェスチャ認識を可能にするツールキット 6
2.4 $-Family Recognizer 6
2.5 手書きジェスチャの評価研究 8
2.6 本研究の位置づけ 8

第 3 章　ユーザ調査 **9**
3.1 アプリケーションユーザへの手書きジェスチャの調査 9
3.2 被験者 9
3.3 調査手順 9
3.4 調査結果 10

第 4 章　$1 アルゴリズム **13**
4.1 特徴 13
4.2 4 つのステップ 14
4.2.1 リサンプル 14
4.2.2 向きの正規化 14
4.2.3 大きさと位置の正規化 15
4.2.4 類似度を高くするための最適な角度の選定 15
4.3 類似度計算 16
4.4 リミテーション 16

第 5 章　$V アルゴリズム **18**
5.1 $V アルゴリズムが目指す特徴 18
5.2 $V アルゴリズムのアイディア 19

5.2.1 大きさ，向き，位置に関して識別可能にする方法 19
5.3 ジェスチャの類似度の定義 19
5.3.1 大きさ 19
5.3.2 向き 19
5.3.3 位置 20
5.4 学習データの保持の方法 20
5.5 ジェスチャグループごとの特徴量の選定 21
5.6 認識に用いる特徴量を選定した時の認識率と認識速度の実験 22
5.6.1 ジェスチャグループの作成方法 22
5.6.2 ジェスチャグループにおける認識に用いる特徴量の選定方法 23
5.6.3 ジェスチャの認識方法 23
5.6.4 被験者 24
5.6.5 実験機器 25
5.6.6 実験手順 25
ジェスチャの取得 25
認識率と認識速度の測定 26
5.6.7 実験結果 26
5.6.8 考察 26
5.7 最適な重み付けのための実験 27
5.7.1 実験結果 27
5.8 ジェスチャの認識 27

第 6 章 評価実験 **30**

第 7 章 アプリケーション例 **31**

第 8 章 議論 **32**

第 9 章 結論 **33**

謝辞 **34**

参考文献 **35**

図 目 次

1.1 ストロークの形状と書き順は同じであるが，大きさ (e), 向き (a) (b) (d), 位置 (c) が異なる，単一ストロークからなる手書きジェスチャの例 3

3.1 $1 において用いられる，一般的にスマートフォンやタブレット端末などのタッチパネルへの手書き入力やペン入力において良く用いられる，単一ストロークからなる手書きジェスチャの例 10
3.2 調査において 6 人の被験者 (P1〜P6) から得られた，用いたい手書きジェスチャの一覧 12

4.1 Each step in the $1 algorithm process 16

5.1 大きさ，向き，位置を特徴量として認識のために用いた場合に，入力データと学習データが一致しない例 21
5.2 An Example of how to decide a similarity (Sts, Sto, Stp) of tempaltes in a group gesture, in this case (0.8, 0.1, 0.4) is the similarities. 24
5.3 The screen shot on the smartphone. The green area is the input area 25
5.4 The procedure to decide the optimal weight values 28
5.5 The procedure to decide non optimal weight values 28
5.6 The result of the experiment to find the optimal weight values, and blue lines indicates power approximation curves. 29

第1章　序論

本研究においては，ユーザが独自に定義した手書きジェスチャを高速に認識し，かつどのような開発環境においても実装可能な軽量なアルゴリズムを示す．本章においては，まず初めに研究背景として既存の手書きジェスチャ認識手法とその課題について述べる．次にその課題を解決すべく本研究の目的について述べ，最後に本論文の構成を述べる．

1.1　背景

スマートフォンやタブレット端末のディスプレイへの入力手法として，ペンや指を用いた手書きジェスチャが多く採用されている．特にそれらを入力として用いるようなアプリケーションをプロトタイピングする環境において，手書きジェスチャをアプリケーションに組み込む際に求められることとして，[Ret94] において述べられているように，手書きジェスチャ認識を実現するためのパターン認識に関する専門的な知識 [HTH00, ABS04, SD05, CB05, Pit91, Cho06, Rub91a, AW10] がなくとも，素早く実装できること (すなわちアルゴリズムが簡潔であるということ)，素早く手書きジェスチャ入力のテストができること (すなわちあまり学習データを必要としないこと) ことなどが挙げられる．同時に，ジェスチャ認識であるため，認識率が高いこと，認識速度が速いことも求められる．また，手書きジェスチャであるため，入力されるジェスチャは毎回微妙に形状が異なるジェスチャが入力される可能性が高いため，たとえ形状が微妙に異なったとしても意図したジェスチャとして正しく認識されるようなロバスト性の高い認識アルゴリズムであることも求められる．また，学習データをあまり必要としないということは，アプリケーションユーザが独自に手書きジェスチャを定義することが可能なシステムを実現することにもつながり，これも手書きジェスチャを入力として用いるようなアプリケーションを開発する多くの開発者によって求められていることの１つである．

$1 [WWL07] は，まさにこれらの要件を満たす手書きジェスチャ認識アルゴリズムであり，「アルゴリズムが簡潔である，少ない学習データにおいて高い認識率を示す，認識速度が速い，ロバスト性が高い」といった特徴を持ち，単一ストロークからなる手書きジェスチャを認識可能なアルゴリズムである．$1 は，現に，ActionScript, Python, C#, C++, Objective-C, Java, Java ME, and JavaScript(全て URL 引用する) といった言語において実装されている．そのため，多くのソフトウェア開発者にとって，手書きジェスチャ認識を自身のシステムに組み込むことが容易となった．特に手書きジェスチャ認識のためのライブラリが存在しないようなプロトタイピング開発環境において，その存在価値は非常に高い．また，$1 を改良した多くのアルゴリズムは$-Family Recognizer [AW10, RSH11, Li10, AW12, HS12, VAW12, TL15] として知ら

れ，$1 の持つ「アルゴリズムが簡潔である」という特徴を維持しつつ，認識率や認識速度の向上，識別可能なジェスチャの種類の増加といった観点から改良が試みられた．ペンや指を入力として用いるようなインタフェースが普及し，それらを用いたソフトウェア開発者が増加している今日において，これらアルゴリズムは，手書きジェスチャ認識を自身のシステムに容易に組み込むための手段として必要とされており，アルゴリズムの開発/改良の機運が高まっている．これまで開発されてきた$-Family Recognizer は，ジェスチャを構成するストロークの，大きさ，向き，位置に関して，そのいずれかあるいはすべてについて不変になるようなアルゴリズムを採用することにより，不変にしたものについてロバストなジェスチャ認識を実現した．つまり，手書きジェスチャの書き順が同じでありかつ手書きジェスチャを構成するストロークの形状さえ類似していれば，ストロークの大きさ，向きやストロークが入力された位置などが多少異なっても，同じ形状と書き順の手書きジェスチャとして認識されることを示す．その結果，認識率の向上が実現された．それらを不変にしない，つまり特徴量として認識のために用いるということは，その特徴量について識別できるため，識別可能な手書きジェスチャが増加することにつながるが，不変にしない特徴量についてはロバストではなくなるため，認識率の低下を招く恐れがある．例えば，Rubine Classfier [Rub91b] は 13 もの手書きジェスチャの特徴量を用いることにより手書きジェスチャを認識し，手書きジェスチャを構成するストロークの形状と書き順は同じであるが，大きさ，向き，位置に関して異なるジェスチャ (図 1.1) を識別することが可能である．しかしながら，認識に用いる特徴量が多いため，結果的に認識率の低下を招いている．

認識速度の観点でいえば，DTW [Tap82, SC07] は，HCI 分野において，ジェスチャ認識をするためによく採用されているアルゴリズムであり，単純にストロークを構成する点の距離を比較するのみであるため，こちらも，手書きジェスチャを構成するストロークの形状と書き順は同じであるが，大きさ，向き，位置に関して異なるストロークを識別することは可能である．しかしながら，単純なアルゴリズムによって認識率を向上させるために，非常に計算量が大きいといった問題点がある．

しかしながら，第 3 章において示す事前調査により，これまでに述べたきたような，ストロークの形状と書き順は同じであるが，大きさや向きや位置が異なる単一ストロークからなる手書きジェスチャ (図 1.1) をアプリケーションユーザが入力として用いることを要望していることが導かれた．これらは既に述べたように既存の$-Family Recognizer において識別できないため，これらのような手書きジェスチャを入力として用いたいソフトウェア開発者は既存の$-Family Recognizer を認識アルゴリズムとして用いることができない．また，これらを識別可能な Rubine classfier や DTW などを採用した場合は，認識率や認識速度において十分なパフォーマンスを得られない可能性がある．

1.2 目的

本研究の目的は$1 を拡張し，単一ストロークからなる手書きジェスチャに対し，ストロークの特徴量である，大きさ，向き，位置に関して識別可能としながらも，$1 と比較し認識率

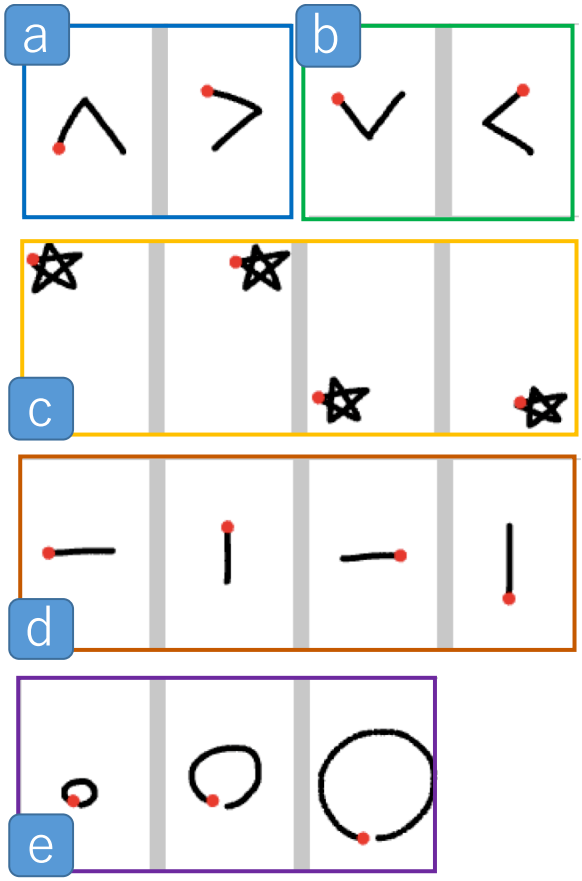

図 1.1: ストロークの形状と書き順は同じであるが，大きさ (e), 向き (a) (b) (d), 位置 (c) が異なる，単一ストロークからなる手書きジェスチャの例

の低下と認識速度の低下を最小限に抑えることを実現した$-Family Recognizer 開発することである．我々はこの，ストロークの大きさ，向き，位置に関して “V”ariant (不変) なストロークを認識する$-Family Recognizer を$V と名付けた．$V は$1 と同様，簡単なアルゴリズムによって構成され，どのような開発環境においても実装可能であることも目標としている．また，$1 と同様，少ない学習データにおいて高い認識率を示すことも目標としており，これは，アプリケーションユーザが手書きジェスチャを独自に定義することが可能であることを示している．また，大きさ，向き，位置に関して識別可能な既存アルゴリズムと比較することによって，$V の有用性を示すことも本研究における目的とする．

1.3 貢献

本研究における手書きジェスチャ認識アルゴリズム$V の貢献を以下に示す．

- 手書きジェスチャを構成するストロークの形状と書き順は同じであるが，大きさ，向き，位置に関して異なる手書きジェスチャ識別することが可能なアルゴリズムを開発した．
- 少ない学習データにおいて認識率を向上させるためのアルゴリズムを開発した．
- 認識速度を向上させるためのアルゴリズムを開発した．
- どのような開発環境においても実装可能なアルゴリズムを開発した．

1.4 本論文の構成

第 1 章では，研究背景と目的を述べた．第 2 章では，関連研究を述べる．第 3 章では，本研究の動機にもなった，アプリケーションユーザが入力として用いたい手書きジェスチャの調査について述べる．第 4 章では，$V の拡張元である$1 アルゴリズムについて述べ，第 5 章において，$V アルゴリズムの詳細について述べる．第 6 章では，$V のアルゴリズムとしての性能評価実験について述べる．第 7 章では，$V を用いたアプリケーション例を述べる．第 8 章では，$V アルゴリズムの今後の展望について議論する．第 9 章では，本研究の結論を述べる．なお，付録 A に$V アルゴリズムの擬似コードを，付録 B に第 3 章のユーザ調査に用いた調査同意書を，付録 B に調査について説明する際に用いた説明書を，付録 D に第 6 章の評価実験に用いた実験同意書を示す．

第2章　関連研究

本章においては，本研究に関する研究について述べる．本研究では，ユーザ定義手書きジェスチャを高速に認識し，かつどのような開発環境においても実装可能な軽量なアルゴリズムを提案している．また，ストロークの大きさ，向き，位置に関して識別可能な$-Family Recognizerである．よって，本研究の関連研究は，一般的な，手書きジェスチャ認識アルゴリズムの研究，ユーザ定義に特化した手書きジェスチャ認識アルゴリズムの研究，手書きジェスチャ認識を可能にするツールキット，$-Family Recognizer に関する研究，手書きジェスチャの評価に関する研究に分類される本章においては，それらの研究について述べた後，最後に本研究における手書きジェスチャ認識アルゴリズムである$V の位置づけについて述べる．

2.1　手書きジェスチャ認識アルゴリズム

文字，ストロークの形状，手書きジェスチャなどの認識は，長く広く研究されている分野であり，多くのアルゴリズムにより実現されてきた，finite state machines [HTH00] (有限オートマトン) は，有限個の状態と遷移と動作の組み合わせからなる論理モデルであり，ある「状態」において，何らかのイベントや条件によって別の状態へ「遷移」することを繰り返すことによって最終的な認識結果を導く．高い認識精度を示すためには，より詳細なモデルの定義が必要となる．Hidden Markov Models (HMMs) [ABS04, SD05, CB05] (隠れマルコフモデル) は，観測された出力の系列から，内部の状態系列を統計的に推測するためのアルゴリズムである．neural networks [Pit91] は脳機能の特性を計算機上に応用したアルゴリズムであり，大量の学習によってモデルを最適化し，多次元量のデータで線形分離不困難な問題に対して比較的小さい計算量で良好な解を得ることができる．feature-based statistical classifiers [Cho06, Rub91a] は，大量の学習データによる特徴量をもとに学習データをクラスタリングし，より低次元な認識モデルを生成するためのアルゴリズムである．ad hoc heuristic recognizers [AW10, WS03] は,「限定的な」認識アルゴリズムであるため，事前に定義されたジェスチャのみ認識することができる．アプリケーション実行時において，新たな学習データを追加した場合に，新たなヒューリスティック関数を定義しなければならないため，アプリケーションユーザが独自にジェスチャを定義することができない．template matching [KS05, KZ04] は．主に画像処理として用いられ，学習データと入力データの画像をそれぞれ走査し，画像上の各位置における類似度を算出するアルゴリズムであり，手書き文字にも応用されている．

これらアルゴリズムはオンライン文字認識及びオフライン文字認識に双方においてしばしば用いられるアリゴリズムである．オンライン文字認識とは，ディスプレイなどにペンや指

などによって入力された文字を認識する技術の総称であり，オフライン文字認識とは，紙に書かれた文書イメージを光学スキャンし，そのイメージを自動的にコンピュータで処理可能なテキストデータに変換する技術の総称である．しかしながら，これらアルゴリズムは，高い認識精度を示す認識モデルを生成するために膨大な数の学習データが必要であり，素早く手書きジェスチャ入力のテストしたい場合において不向きであるだけでなく，アプリケーションユーザが独自にジェスチャを定義する上で実用的であるとは言えない．また，これらアルゴリズムを実装することは，本分野に精通していない開発者にとって困難である．

2.2 ユーザ定義に特化した手書きジェスチャ認識アルゴリズム

Rubine classfier [Rub91b] や Dynamic programming (DTW) [Tap82] などは，少ない学習データにおいてジェスチャ認識可能なアルゴリズムであるが，Rubine classfier はアルゴリズムに用いられる数式が複雑である上，学習データが少ない場合は認識率が高いとは言えない．また，DTW はアルゴリズムが簡潔であるが，計算量が非常に大きいという問題点がある．計算量を改善した Fast DTW [SC07] が開発されたが，アルゴリズムは複雑になっており，プロトタイピング環境開発向けとは言い難い．また，Rubine は多くある特徴量を適切に選ばないと，認識率が低下するという欠点があり，特徴量の選定が難しい．このように認識に用いる特徴量を単に増やすことは，それについて識別できることにつながるが，ロバスト性に欠ける欠点もある．

2.3 手書きジェスチャ認識を可能にするツールキット

プロトタイピング環境向けに開発できるように，ジェスチャ認識を簡単に開発可能なツールキットも開発された．SATIN [HL00] はペンベースのユーザインタフェースであり，ジェスチャ認識のモデルを手書きによって定義することができ，ジェスチャ認識の開発を容易にしたツールキットである．Henry et al. [HHN90]，Landay and Myers [LM93] によるツールキットは〜，Amulet toolit [MMM+97] は〜がある．これらは，開発を手助けするのに非常に強力であるが，対応可能な開発環境が決まっており，自身の環境に適用できない場合がある．

2.4 $-Family Recognizer

$1 [WWL07] は，プロトタイピング環境において実装可能であり，少ない学習データにより，高い認識率を示すアルゴリズムであるため，ユーザが独自にジェスチャを定義するジェスチャ認識システムを開発することが可能である．プロトタイプ開発者は，自身のシステムに，簡単な数式のみを含む，およそ 100 行からなるアルゴリズムを追加するのみによって，図のような単一ストロークからなるジェスチャ認識を行うことが可能である．$1 は geometric template matchng approach を用い，canditate gesture と template gesture の対応する点のユークリッド距離が最小となるような最適な角度を探索することによってジェスチャ認識を行っている．しか

しながら，ストロークを，大きさ，向き，位置に不変にしているため，それらの特徴量に依存するようなストロークを認識することができないため，そのようなストローク認識を行いたい開発者は$1 を自身のシステムに採用することができない．$1 が識別/認識することができないストロークを識別/認識するために，これまで$1 を拡張したアルゴリズムが開発されてきた．$N [AW10] は，複数のストロークからなるジェスチャを認識することを可能にし，識別可能なストロークを大幅に増やすことに成功した．$N は，ストロークを複数の単一ストロークに分割し，それぞれの単一ストロークを$1 の手法によって認識した．また，自動的に all possible permutations of a multistroke を計算することによって，ストロークの向きや書く順番にロバストな認識も可能にした．Quick$ [RSH11] は，$1 の改良であり，Hierarchical Clustering によって，対応する点のユークリッド距離が最小となる canditate gesture と template gesture の組み合わせを効率的に探索し，認識速度を高速化することに成功した．Protractor [Li10] も，$1 に対し，認識速度の面において改良し，最適な角度を探索する際に，GSS [PTVF92](pp. 397-402) を用いた$1 とは異なり，a closed-form solution を用いることによって，より高速に探索することを可能とした．$N-Protractor [AW12] は，$N に対し，Protractor の手法を用いることによって，より高速にかつ，より正確に複数のストロークからなるジェスチャを認識することを可能にした 1 cent Recognizer [HS12] は，$1 よりも高速であり，アルゴリズムも非常に単純であるため実装が容易であるが，認識/識別可能なジェスチャの種類や，認識率の観点から見るとあまり実用的であるとは言えない．$P [VAW12] は，ストロークを構成する点を Point Cloud として扱うことによって，$N-Protractor よりも，memory や execution costs の点において効率的なアルゴリズムを示した．Penny Pincher [TL15] は，ストロークを構成する点間のベクトルを用いることによって，これまでの$-Family Recognizer と比べて，より高速にかつ，正確に認識することを可能にした．

これらの$-Family Recognizer は，2D のストローク認識をすることが可能であり，$1 に対し，アルゴリズムを簡略化したり，認識速度を高速化したり，認識率を高くしたり，認識できるストロークの種類を増やしたりするなどして，繰り返し改善されてきた．

しかしながらこれらのアルゴリズムは，ストロークの特徴量である，大きさ，向き，位置に関して，そのいずれかあるいはすべてについて不変になるようなアルゴリズムを採用することにより，それらの特徴量についてロバストなジェスチャ認識を実現し，その結果認識率や，認識速度の向上を実現してきた．それらを不変にしないことは，その特徴量について識別できるようになることにつながるが，不変にしない特徴量についてはロバストではなくなるため，認識率の低下や認識速度の低下を招く恐れがある．例えば，1 cent Recognizer はストロークを構成するすべての点の中心座標から，それぞれの点へのユークリッド距離のみを特徴量とし，canditate gesture と template gesture の特徴量の差が最小となるストロークを探索しているため，ストロークの形状は同じであるが，大きさ，向き，位置に関して異なるストロークを識別することは可能である．しかし，それぞれの特徴量についてロバストでないため，それぞれの特徴量について微妙に異なる場合，認識できないことが多々あり，結果的に認識率の低下を招く．認識速度の観点でいえば，DTW は，単純にストロークを構成する点の距離を比較するのみであるため，こちらも，ストロークの形状は同じであるが，大きさ，向

き，位置に関して異なるストロークを識別することは可能であるが，認識率を向上させるために，非常に計算量が大きい.

2.5 手書きジェスチャの評価研究

2.6 本研究の位置づけ

第3章　ユーザ調査

本章においては，本研究の動機にもなった，アプリケーションユーザが入力として用いたい手書きジェスチャの調査内容とその結果について述べる．

3.1　アプリケーションユーザへの手書きジェスチャの調査

単一ストロークからなる手書きジェスチャ認識を用いたシステムが，これまでにも多く開発されてきた．その中で，$1 は，[HL00, LM93, LNHL00] において用いられているような，一般的にスマートフォンやタブレット端末などのタッチパネルへの手書き入力やペン入力において良く用いられる，単一ストロークからなるジェスチャを抜粋し (図 3.1)，それらについて認識率を計測した．それぞれについて詳しく書く．しかしながら，ユーザ定義ジェスチャを用いた研究 [Vat12, BNLH11, WMW09, SMSJ+15] の中には，単一ストロークからなるジェスチャであり，かつ，ストロークの形状は同じであるが，ストロークの向きや位置の違いを利用したジェスチャを入力に用いる場合があった．

そこで，普段スマートフォンを利用する場面，およびスマートフォンを入力デバイスとし PC を操作する場面を想定した時，どのようなアプリケーションに対し，どのような手書きジェスチャを入力として用いたいかを事前調査した．

3.2　被験者

被験者は普段からスマートフォンを使用している大学院生の男性 6 名である．年齢は 21〜27 歳 (平均 23.8 歳) であり，全員右利きであった．6 名の被験者の中にはコンピュータサイエンスあるいはユーザインタフェースを専攻している人が 4 名存在し，残りの 2 名は，社会工学を専攻している男性と，エンジニアとして働いている男性であった．被験者は全員日頃からスマートフォンを使用していた．

3.3　調査手順

我々はまず，被験者に調査の目的を説明した．その後，普段スマートフォンを利用する場面，およびスマートフォンを入力デバイスとし PC を操作する場面を想定した時，どのようなアプリケーションに対し，どのような手書きジェスチャを入力として用いたいかを 20 以上考えるよう指示し，自身のスマートフォンに対し実際に入力しながら考えるよう指示した．そ

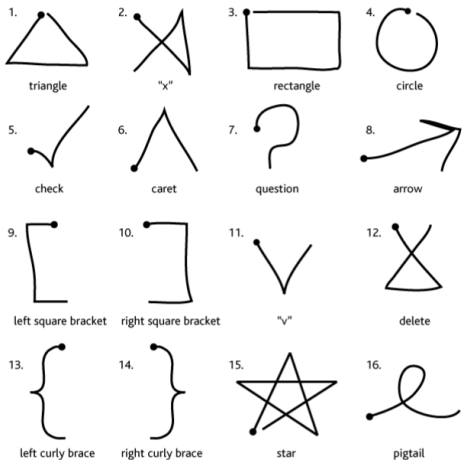

図 3.1: $1 において用いられる，一般的にスマートフォンやタブレット端末などのタッチパネルへの手書き入力やペン入力において良く用いられる，単一ストロークからなる手書きジェスチャの例

の際，ジェスチャを入力する姿勢は次の 3 つの姿勢 (姿勢 1，姿勢 2，姿勢 3) から，実際に自分がそのジェスチャを入力として用いるときに入力する姿勢として 1 つ選んでもらった．

1. 机にスマートフォンをおいて，利き手の人差し指によって入力する．
2. 利き手でスマートフォンを握りながら，同じ利き手の親指によって入力する．
3. 利き手とは反対の手でスマートフォンを握りながら，利き手の人差し指によって入力する．

被験者は，全ての姿勢を座って入力するよう指示された．また，入力ジェスチャとして，単一ストロークからなるよう指示した．

入力として用いたい手書きジェスチャを 1 つ考えるたびに，我々は付録何の付録かに示す紙にそのジェスチャをボールペンによって書くよう指示した．またそれに加え，そのジェスチャを入力した姿勢 (1〜3)，そのジェスチャをどのアプリケーションに対して用いたいのかも同時に書くよう指示した．

3.4 調査結果

図 3.2 は，6 人の被験者から得られた手書きジェスチャ一覧である．

図 3.2 において，例えば音楽再生アプリケーションにおいて，早送り，巻き戻し，音量の上げ下げ，といったような前後への移動や，値の上げ下げなど，対になるような操作に対して，向きや大きさの違いを用いて入力する要望があった．また，ブラウザアプリケーションにおいて，位置によって登録先のブックマークを変える，ページ内における表示位置をジェスチャ

の入力位置に対応させる，といったように，位置の違いを操作対象先の違いに割り当てたり，現在操作しているものの位置に対応付けるといった要望があることがわかった．

また，片手で操作する場面 (姿勢 2) においては，難しいストローク操作ができないため，極力簡単なストロークを用い，かつ，大きさ，向き，位置の特徴量を利用して入力する要望があることも分かった．ここら辺もうちょい詳しく書く？

$V は，図 3.2 が示すような，ジェスチャの形状や書き順は同じでも，大きさ，向き，位置に関して異なる手書きジェスチャを認識することが可能なアルゴリズムであり，今回調査を行った被験者の要望を満たすことのできる手書きジェスチャ認識である．

図 3.2: 調査において 6 人の被験者 (P1〜P6) から得られた，用いたい手書きジェスチャの一覧

第4章　$1アルゴリズム

$V は$1 の拡張であるため，本章において$1 アルゴリズムについて述べる．

4.1　特徴

ユーザが入力した手書きジェスチャは，図のように複数の点によって構成され，すでに登録された手書きジェスチャと，それぞれの点を比較することによって，どの手書きジェスチャと一致しているかが判別される．しかしながら，これらの手書きジェスチャを構成する複数の点は，入力に用いているハードウェアやソフトウェアに依存した速度によってサンプリングされる．それに加え，人間によって入力される手書きジェスチャはばらつきがあるため，入力される手書きジェスチャを構成する点の数は入力されるたびに異なる．そのため，ユーザが入力した手書きジェスチャと，すでに登録された手書きジェスチャを比較するにあたり，手書きジェスチャを構成する点どうしを単純に比較することは困難であると言える．

例えば，図のような “pigtail” と “x” の手書きジェスチャのそれぞれのペアにおいて，手書きジェスチャの大きさや，手書きジェスチャを構成する点の数が異なる．この特徴は，手書きジェスチャ認識において双方を比較する上で，1 つのハードルとなっている．また，pigtail は回転することによって x と似た手書きジェスチャとなる．これらの問題に対処しつつ，単純なアルゴリズムを実現するために，以下のような基準に従うようなアルゴリズムの実現を目指した．

- ハードウェアやソフトウェアのセンシング及び入力する速度などによって変わるサンプリングされる点の数の違いに対してロバストであること．
- 手書きジェスチャの大きさ，向き，位置の不変に関してオプショナルに設定可能であること．
- 数学的な高度な知識やテクニックをを必要としないこと (例えば，逆行列，微分，積分など)
- 少ないコードによって実装できること．
- 認識速度が速いこと．
- ソフトウェア開発者やアプリケーションユーザが，独自に手書きジェスチャを定義できること．

- N-best list に関して，高い識別能力を示すスコアを示すこと．
- 図 3.1 のような単一ストロークからなる手書きジェスチャを認識するにあたり，HCI 分野において多く用いられる既存の複雑な手書きジェスチャ認識アルゴリズムと比べても，高い認識率を示すこと．

ここで，N-best list とは，N 個の学習データそれぞれに対する入力データとの類似度を降順に並べたものであり，N-best list の最高値と 2 番目の値の差が大きいほど，高い識別能力を示していると言える．

次に，これらの基準に従うような$1 のアルゴリズムを述べる．アルゴリズムは大きく 4 つのステップから構成される．

4.2　4 つのステップ

前節において述べた基準を満たすために，入力データ及び学習データは 4 つのステップを経た後に比較される．4 つのステップは，リサンプル，向きの正規化，大きさと位置の正規化，〜からなる．

4.2.1　リサンプル

前節において述べたように，手書きジェスチャを構成する点の数は，ハードウェアやソフトウェアのセンシング及び入力する速度などによって変わる．特に，入力速度の違いによる点の数の違いは顕著である．(図)

点の数が違うことにより，入力データと学習データの手書きジェスチャを構成する点を互いに比較することが困難となっている．そこで，図に示すように N 個の等間隔に並ぶ点にリサンプルすることとする．N 個の点にリサンプルすることは，生のデータを扱うことと比べて正確なデータを扱っているとは言えず認識精度が落ちる可能性があるが，入力データと学習データ双方の手書きジェスチャの点の数が等しくなるため，容易に互いの対応する点を比較できる．

リサンプルすることは，既存の手書きジェスチャ認識アルゴリズムにおいても用いられている [PS00, TSW90, KZ04, ZK03, Tap82]．$1 はこれらのアルゴリズムを採用するとともに，これらのアルゴリズムとは違い向きに不変なアルゴリズムを提供する．また，手書きジェスチャを構成する点の数をリサンプルすることなく実行可能な DTW と比較している．

N 個の等間隔に並ぶ点にリサンプルする方法を図に示す．リサンプルの仕方を図で説明

また，32¡N¡256 の時，N=64 の時が最適であることがわかった．

4.2.2　向きの正規化

リサンプルされた手書きジェスチャの向きを “indicative angle” として定義する．indicative angle とは図のように，手書きジェスチャの書き始めの一番最初の点の座標，手書きジェスチャ

を構成する点全てによる中心座標，0 度方向によって形成される角度である．そして，indicative angle の沿って，全てのジェスチャを回転させる．これにより，ジェスチャは向きに正規化される．

4.2.3 大きさと位置の正規化

向きによって正規化されたジェスチャの大きさを “boundingbox” として定義する．boundingbox とは図のように，ジェスチャに隣接するような矩形を示す．全てのジェスチャについて，boundingbox をある一定の大きさの矩形に正規化することによって，ジェスチャは大きさに正規化される．

大きさによって正規化されたジェスチャの位置をジェスチャを構成する全ての点による中心座標として定義する．全てのジェスチャについて (0, 0) へと移動させることによって，向きに正規化される．

4.2.4 類似度を高くするための最適な角度の選定

リサンプルされた 2 つの手書きジェスチャ構成する点を 1 つずつ対応する点どうし比較するにあたり，1 度ずつ手書きジェスチャを回転させながら，最も類似度が高くなるような角度を見つけた上で，類似度を算出する方法 [KS05] は認識のために膨大な時間を要することになる．学習データの数が全て含めて 30 くらいであればそのような方法でも十分な速度によって認識することが可能であるが，$1 は黄金分割探索 [PTVF92] を用いる．黄金分割探索とは，単峰関数 (極値が 1 つしかない関数) において，極値を求めるための方法 (局所探索法) のうち効率的な方法の 1 つであり，極値が存在することが自明な範囲において極値を逐次的に求める方法である．

例えば図のように，f(x) の関数があり，極小値 f(x’) を求める時に，x1 と x3 の間に極値が存在することが自明な時に，その範囲内に存在する f(x2) を求める．この時 x2 は (x2 - x1) : (x3 - x2) が黄金比 (黄金比入れる) となるように設定する．これが黄金分割探索と言われる所以であり，常に 3 点 (この場合 x1, x2, x3) が存在する．その後広い区間 (この場合 x2 と x3 の間) において，同様に黄金比によって分割し新たな f(x4) を得る．この時，f(x4a) ならば，極小値は x1 と x4 の間に存在するため，新たな 3 点は x1, x2, x4 となる．f(x4b) ならば，極小値は x2 と x3 の間に存在するため，新たな 3 点は x2, x4, x3 となる．このように，極小値が存在する範囲を徐々に狭めていくことによって，効率よく極小値を求めることができる．

$1 の場合，480 個の手書きジェスチャにおいて，+-45 度の範囲において極小値が存在することが発見され，極小値が存在する範囲が 2 度になるまで黄金分割探索を行う．この時，入力データに類似する学習データが存在する場合あるいはしない場合においても，10 ステップ後には極小値が求められることが発見された．

局所探索法の 1 つである山登り法 (引用) を黄金分割探索の代わりに用いる場合，480 個の手書きジェスチャにおいて，類似するジェスチャの場合およそ 7.2 ステップ後に極小値を求め

ることができるが，類似するジェスチャが存在しない場合はおよそ 53.5 ステップ後に極小値が求められる．つまり学習データが 10 ずつ存在する 16 種類の手書きジェスチャ=160 において，黄金分割探索は 160= 1600 ステップ必要であるのに対し，山登り法は 7.2+ 53.5= 8097 ステップ必要であり，およそ 80.2%もの計算量の節約となっている．

このように，類似度を高くするための最適な角度を黄金分割探索によって効率的に求めることによって，認識速度の向上を実現している．

以上の 4 ステップをまとめ，入力データと学習データそれぞれを比較するまでの過程を図 4.1 に示す．

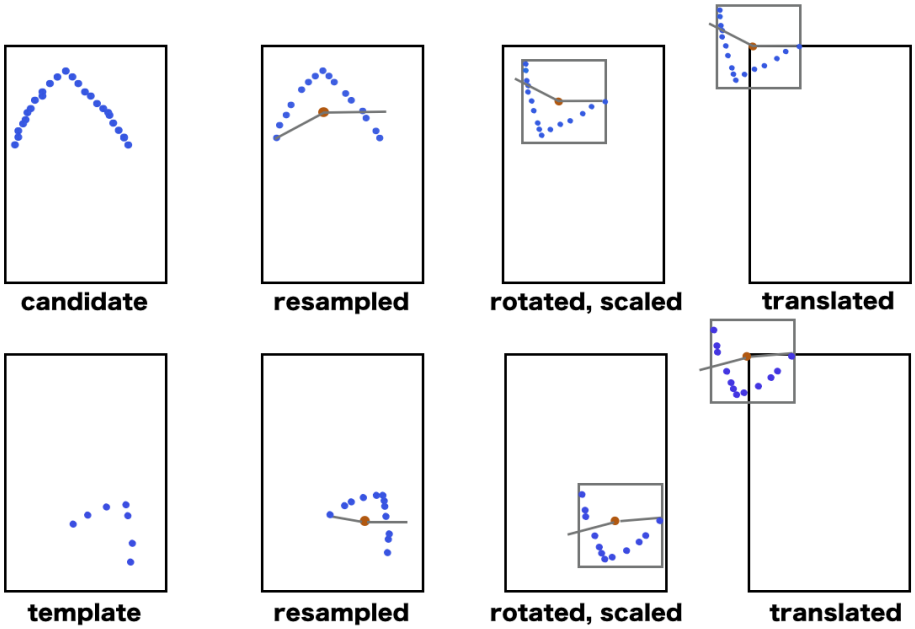

図 4.1: Each step in the $1 algorithm process

4.3 類似度計算

前節までの 4 ステップによって得られた入力データと学習データの最終的な類似度計算は式 (1) 及び式 (2) によって表される．式を載せるここで〜式の説明

4.4 リミテーション

$1 は，手書きジェスチャを，大きさ，向き，位置に正規化することによってアルゴリズム全体を簡潔化したり，それぞれについてロバストな認識を可能にし認識率の向上を実現した．しかしながら，このことが原因により，幾つかのリミテーションが存在する．

- 手書きジェスチャを大きさ，向き，位置によって識別しない.
- 直線のような 1 次元の手書きジェスチャを認識することができない.
- 手書きジェスチャを入力する速度による識別ができない.

「手書きジェスチャを大きさ，向き，位置によって識別しない」ことに関しては，それぞれについて正規化しないようにアルゴリズムを変えることによって識別することが可能となる．しかしながら，正規化しないことにより，正規化しないものに関してはロバスト性が低下するため，結果的に認識率の低下を招く恐れがある．

「直線のような 1 次元の手書きジェスチャを認識することができない」ことに関しては，向き，大きさに関して正規化した際に図のようになるためである．

「手書きジェスチャを入力する速度による識別ができない」ことに関しては，入力速度の要素を特徴量として用いることによって識別可能となる．例えば，Rubine [Rub91b] のような特徴量ベースの認識は，速度を特徴量として用いている．このように認識に用いる特徴量が複数存在する場合，用いる特徴量を適切に選択できれば，つまり，認識に本当に必要な特徴量のみを用いて，認識には必要のない特徴量を選択できれば，認識率の高い手書きジェスチャ認識アルゴリズムとなる．しかしながら，一般的に手書きジェスチャ認識アルゴリズムに関する深い知識がない限り，そのような適切な選択は困難である．$1 は，識別可能な特徴量が少ないが，そのようなわずらわしさを一切省いたアルゴリズムとなっている．

第5章　$V アルゴリズム

本章において，$1 を拡張した$V アルゴリズムについて述べる．

5.1　$V アルゴリズムが目指す特徴

$V アルゴリズムが目指す特徴を以下に示す．

- $1 の特徴を維持すること．つまり，
  - ハードウェアやソフトウェアのセンシング及び入力する速度などによって変わるサンプリングされる点の数の違いに対してロバストであること．
  - 数学的な高度な知識やテクニックをを必要としないこと (例えば，逆行列，微分，積分など)
  - 少ないコードによって実装できること．
  - 認識速度が速いこと．
  - ソフトウェア開発者やアプリケーションユーザが，独自に手書きジェスチャを定義できること．
  - N-best list に関して，高い識別能力を示すスコアを示すこと．
  - 図 3.1 のような単一ストロークからなる手書きジェスチャを認識するにあたり，HCI 分野において多く用いられる既存の複雑な手書きジェスチャ認識アルゴリズムと比べても，高い認識率を示すこと．
- その上で，形状や書き順が同じ手書きジェスチャを大きさ，向き，位置に関して識別可能にすること．

これまで述べてきたように，一般的に，特徴量を不変にすることによってその特徴量についてロバスト性が向上するが，不変せず，認識に用いる特徴量として扱う場合，ロバスト性が低下し，結果的に認識率の低下を招く恐れがある．つまり，$1 アルゴリズムを踏襲した上で，大きさ，向き，位置を特徴量として用い，それらに関して識別可能にするということは，$1 と比べて，認識率が低下すると言っていいだろう．以上を踏まえ，$1 に比べて，認識率や認識速度を著しく損なうことなく，図のような手書きジェスチャの形状と書き順は同じでも，大きさ，向き，位置が異なるジェスチャを識別するアルゴリズムを実現することが$V が目指すところである．

5.2 $V アルゴリズムのアイディア

ここでは，$V アルゴリズムのアイディアについて述べる．

5.2.1 大きさ，向き，位置に関して識別可能にする方法

ジェスチャを大きさ，向き，位置によって識別可能にするための方法として以下の 2 つが考えられる．

1. 単純にリサンプリングした点のみによって判別する (正規化しない).
2. 正規化した上で，それぞれを特徴量として用いる．

1. の場合，リサンプリングしただけの実質生データのまま比較するため，大きさ，向き，位置によって識別可能となる．しかしながら，手書きジェスチャの場合，アプリケーションユーザの入力は毎度微妙に異なることが予想される．そのため，類似したジェスチャにおいても，類似度が低くなり，ロバスト性が低下する恐れがあるとともに，認識されたか否かを判別するための類似度の閾値の設定が困難になることが予想される．2. の場合，ジェスチャを正規化するためロバスト性は維持され，その上で，大きさ，向き，位置を特徴量として用いるため，1. の場合と比べて，類似度が低くなりづらくなると予想される．そこで，$V は 2. の方法を用いることとする．

5.3 ジェスチャの類似度の定義

ここでまず，ジェスチャの大きさ，向き，位置を特徴量として用いた時の，それぞれの類似度の定義を示す．

5.3.1 大きさ

$$Ss = \begin{cases} \frac{S'}{S}(S > S') \\ \\ \frac{S}{S'}(S' > S') \end{cases} \tag{5.1}$$

それぞれの式の説明

5.3.2 向き

$$So = 1 - \frac{|\theta - \sigma|}{\pi} \tag{5.2}$$

それぞれの式の説明

5.3.3 位置

$$Sp = 1 - \frac{\sqrt{(X - x')^2 + (Y - y')^2}}{\sqrt{Width^2 \times Height^2}} \tag{5.3}$$

それぞれの式の説明

5.4 学習データの保持の方法

$V は学習データの保持の方法において特徴がある．

$V は学習データが追加されるたびに，ジェスチャの形状と書き順が同じ学習データを同じグループに分類する．ここでは形状と書き順が同じジェスチャを識別可能な$1 アルゴリズムを用いている．この形状と書き順に従って分類されたジェスチャを “ジェスチャグループ” と名付ける．ジェスチャグループの例を図に示す (あるいは図 1.1 もジェスチャグループの例である)．

このようにジェスチャグループを作成する理由は 2 つある．

- 認識速度の低下を防ぐため．
- 認識率の低下を防ぐため．

「認識速度の低下を防ぐため」について述べる．

大きさ，向き，位置を特徴量として認識に用いる場合，それぞれについての類似度計算を行うこととなる．これを全てのジェスチャについて類似度計算を行った場合，認識速度が低下する要因となる．$V の目的は，ジェスチャの形状と書き順が同じであるが，大きさ，向き，位置に関して識別可能にすることである．そこで，形状と書き順が同じジェスチャが集まったジェスチャグループを作成し，ジェスチャグループ内に存在する学習データのみに対し，大きさ，向き，位置の類似度計算をする (ここら辺がわかりやすくなるような，入力データと複数のジェスチャグループからなる図を作成する)．一般的に，認識に用いる特徴量を増やした場合，増やさない場合と比べて，認識速度は学習データの数に比例して大きくなるが，同一ジェスチャグループ内のみに対し認識に用いる特徴量を増やすことによって，全体的な認識速度の低下を防ぐことが可能となる．

次に「認識率の低下を防ぐため」について述べる．

大きさ，向き，位置の特徴量を認識のために全てのジェスチャに対し用いた場合を考える．

例えば，図 5.1A の場合について考える．学習データ (図 5.1A') が図 5.1A の右下のジェスチャと一致させようと入力されたとする．しかし，この 2 つのジェスチャは大きさが異なるため，大きさを認識のための特徴量として用いている限り類似度は低下する．

図 5.1B の場合について考える．学習データ (図 5.1B') が図 5.1B の右のジェスチャと一致させようと入力されたとする．しかし，この 2 つのジェスチャは向きが異なるため，向きを認識のための特徴量として用いている限り類似度は低下する．

図 5.1C の場合について考える．学習データ (図 5.1C') が図 5.1C のジェスチャと一致させようと入力されたとする．しかし，この 2 つのジェスチャは大きさ，向き，位置が異なるため，大きさ，向き，位置を認識のための特徴量として用いている限り類似度は低下する．

これまでに述べてきたが，大きさ，向き，位置を特徴量として認識のために新たに用いることによって，それぞれについてロバスト性が低下し，結果的に認識率の低下を招く恐れがある．図 5.1 はその例である．

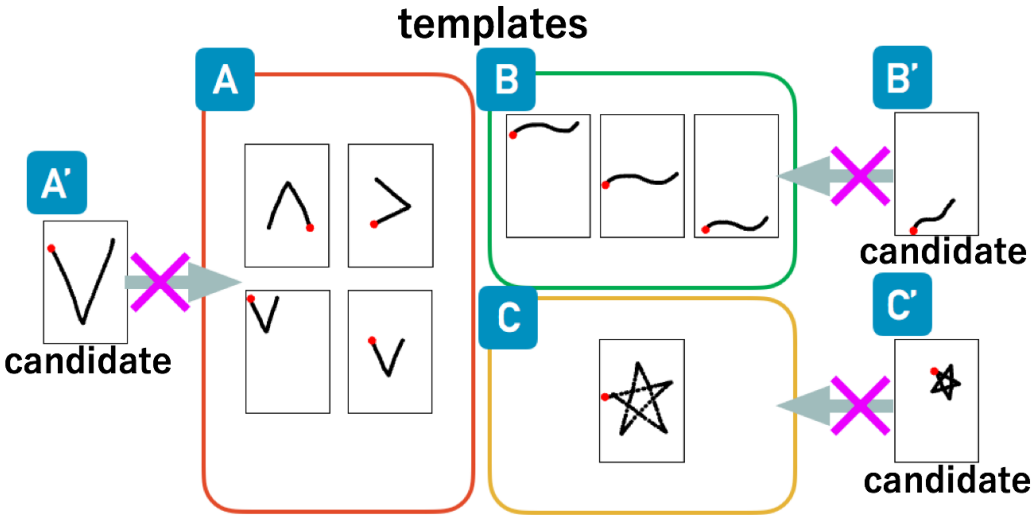

図 5.1: 大きさ，向き，位置を特徴量として認識のために用いた場合に，入力データと学習データが一致しない例

そのため，$V では，ジェスチャグループごとに，大きさ，向き，位置のうちどの特徴量を用いるか選ぶという処理を施す．

5.5 ジェスチャグループごとの特徴量の選定

ジェスチャグループごとに，大きさ，向き，位置のうちどの特徴量を用いるかを選ぶための方法について述べる．

$V の目的は，ジェスチャの形状と書き順が同じであるが，大きさ，向き，位置に関して識別可能にすることである．つまり，同一ジェスチャグループにおいて，大きさ，向き，位置のいずれかあるいはすべてが異なるジェスチャを識別できれば良い．

ここで，図 5.1A の場合について考える．図 5.1A のジェスチャグループには，ジェスチャの大きさは同じであるが，向きや位置が異なるジェスチャが存在している．つまり，向き，位置を特徴量として認識に用いることが必要となる．反対に，大きさは特徴量として認識に用いる必要がない．

図 5.1B のジェスチャグループには，ジェスチャの大きさや向きは同じであるが，位置が異

なるジェスチャが存在している．つまり，位置を特徴量として認識に用いることが必要となる．大きさや向きは特徴量として認識に用いる必要がない．

図 5.1C のジェスチャグループには，1 種類のジェスチャしか存在していない．つまり，いずれの特徴量も認識に用いる必要がない．

このようにして，ジェスチャグループに存在する学習データの種類によって，認識に用いる特徴量を選ぶ，つまり，ある特徴量については認識のために特徴量として用いないことによって，その特徴量について不変にし，ロバスト性を維持することによって，結果的に認識率の低下を防ぐことが可能となると考えた．

以上を踏まえ我々は,「同一ジェスチャグループ内において，他の学習データと類似している特徴量は，認識のための特徴量として用いなければ，認識率の低下を防ぐことができる」という仮説を立てた．

5.6　認識に用いる特徴量を選定した時の認識率と認識速度の実験

前節までにおいて述べてきたように，ジェスチャグループを作成し，ジェスチャグループ内に存在する学習データのみに対し，大きさ，向き，位置の類似度計算をすると認識速度の低下を防ぐことができるという仮説と，同一ジェスチャグループ内において，他の学習データと類似している特徴量は，認識のための特徴量として用いなければ，認識率の低下を防ぐことができるという仮説を検証するための実験を行った．ここではまず，実験を行うにあたって，ジェスチャグループの作成方法とジェスチャグループにおける認識に用いる特徴量の選定方法について述べてから，実験について具体的に述べる．

5.6.1　ジェスチャグループの作成方法

ジェスチャの形状と書き順が同じ学習データが集まったグループである，ジェスチャグループを作成するための手順を以下に示す．まず，学習データが新たに追加される場面を考える．

A. 同じ名前の学習データが存在する場合

新たに追加された学習データは同じ名前の学習データと一緒に保管される．この時，その名前を持つ学習データの，大きさ，向き，位置の特徴量は，それぞれの特徴量の算術平均によって表される．図を入れる

B. 同じ名前の学習データが存在しない場合全体的に図を入れる

$1 アルゴリズムを用いることによりジェスチャの形状と書き順を判断し，

(a) 同じ形状と書き順のジェスチャグループが存在しない場合
新たに追加された学習データは新しい，ジェスチャグループとして保管される．

(b) 同じ形状と書き順のジェスチャグループが存在する場合
そのジェスチャグループに保管される．

ここで，新たに追加された学習データが，既存のジェスチャグループに対し形状と書き順が同じかどうかを判別する方法は，既存のジェスチャグループ内の学習データ 1 つ 1 つと比較することによって行う．今回は$1 による類似度が 0.8 を超えたときに，形状と書き順が同じであると判断し，もし 1 つでも形状と書き順が同じジェスチャが存在すれば，そのジェスチャグループに保管されることになる．なお，保管候補のジェスチャグループが複数存在する場合は，最も類似度が高かったジェスチャが存在するジェスチャグループに保管される．

5.6.2 ジェスチャグループにおける認識に用いる特徴量の選定方法

我々は，同一ジェスチャグループ内において，他の学習データと類似している特徴量は，認識のための特徴量として用いなければ，認識率の低下を防ぐことができるという仮説を立てた．ここで，同一ジェスチャグループ内における他の学習データとの類似度を “ジェスチャグループの学習データ間の類似度” とし，その定義を以下のように定める．

まず，同一ジェスチャグループ内において，学習データを 2 つずつ抽出し，それぞれのペアについて学習データどうしの大きさ，向き，位置の類似度を式入れるを用いて求める．各ペアにおける類似度を比較した時に，それぞれの特徴量について，最小となった類似度を，その形状グループの学習データ間のそれぞれの特徴量についての類似度とする．

例えば，図 5.2 の場合，同一ジェスチャグループ内に 4 つの学習データが存在する．そのため，2 つずつ抽出した場合の学習データの組み合わせは ${}_4C_2=6$ 通り存在する．それぞれのペアについて，学習データどうしの大きさ，向き，位置の類似度は図 5.2 のようになる．この時それぞれの類似度の最小値は，大きさは 0.8，向きは 0.1，位置は 0.4 となるため，このジェスチャグループの学習データ間の類似度は (大きさ，向き，位置)= (0.8, 0.1, 0.4) となる．ここで，類似度の最小値を用いる理由は，類似度が小さいということは，その特徴量に関して類似していないことを示し，類似していない学習データの組み合わせが 1 つでも存在すれば，その特徴量について識別するために，認識のための特徴量として用いる必要があると考えたからである．

以上のようにして得られたジェスチャグループの学習データ間の類似度を用いて，そのジェスチャグループが認識に用いる特徴量を選定する．本実験においては，類似度が 0.8 を超えた場合において，認識に用いる特徴量として選定した．

5.6.3 ジェスチャの認識方法

入力データの認識方法の手順は 2 つに分かれている．

1. $1 アルゴリズムを用いることにより，どのジェスチャグループに属するか判別する．この時，学習データを追加する時と同様，ジェスチャグループ内のすべての学習データに対し類似度を求め，0.8 を超えた場合あるいは，0.8 を超えるジェスチャが複数存在する場合は，最も類似度が高いジェスチャが存在するジェスチャグループに属すると判別する．

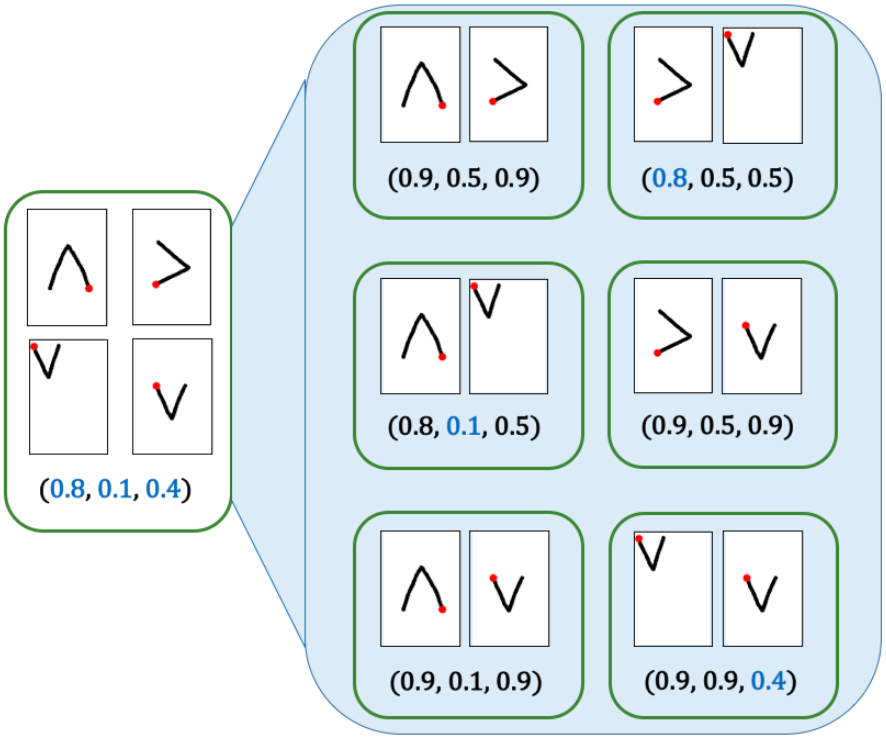

図 5.2: An Example of how to decide a similarity (Sts, Sto, Stp) of tempaltes in a group gesture, in this case (0.8, 0.1, 0.4) is the similarities.

2. ジェスチャグループ内において，どのジェスチャと最も類似しているかを判別する．

2. において，類似度は以下の式によって求められる．

$$S_{final} = S_{cs} \times \alpha_s + S_{co} \times \alpha_o + S_{cp} \times \alpha_p \tag{5.4}$$

平均値になるよう修正

$$\alpha = \begin{cases} 0(\alpha > 0.8) \\ 1(else) \end{cases} \tag{5.5}$$

ここで式の説明

5.6.4 被験者

被験者は，ユーザ調査において協力してもらった 6 名である (男性 6 名，21〜27 歳 (平均 23.8 歳)，全員右利き)．

5.6.5　実験機器

実験には，入力端末として iPhone5 を用い，実験における入力領域は 1.94” × 3.18” であり，解像度は 640 × 1036 である (図 5.3 における緑色の部分).

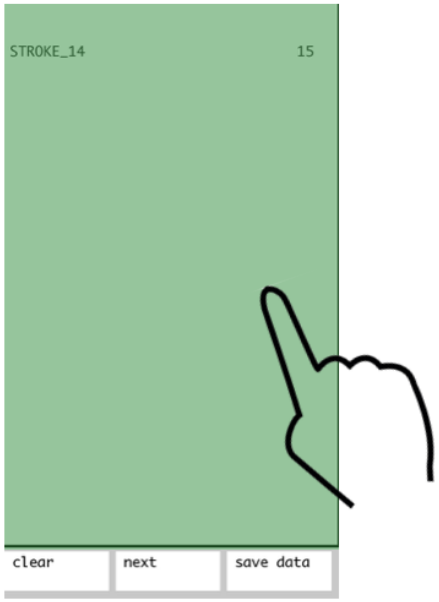

図 5.3: The screen shot on the smartphone. The green area is the input area

5.6.6　実験手順

ジェスチャの取得

我々はまず，被験者に実験の目的を説明した．その後，ユーザ調査において記入してもらった紙を見ながら，それぞれのジェスチャを入力するよう指示した．ジェスチャは図 5.3 における緑色の領域部分にジェスチャを入力するよう指示した．その際，紙に書かれた，そのジェスチャを入力するときの姿勢に従い入力するよう指示した．それぞれのジェスチャには，ジェスチャ番号が STROKE_1 のようにして割り振られており，図 5.3 に示すように，画面左上に入力すべきジェスチャが表示される．タスクの 1 試行は被験者が 1 つのジェスチャを入力するまでである．被験者はランダムに選択されたそれぞれのジェスチャを 1 回ずつ入力し，これを 1 セッションとした．これを 10 セッション行った．被験者によって入力すべきジェスチャの数は異なるが，いずれの被験者においても 20 以上のジェスチャを入力する (20〜24 個のジェスチャ，平均 22 個)．したがって，被験者は平均して計 220 試行 (22 ジェスチャ × 10 セッション) 行った．ジェスチャが思うように入力できなかった場合には，何度でも書き直し可能とした．

認識率と認識速度の測定

それぞれの被験者ごとのジェスチャを“ジェスチャセット”とする．本実験において被験者は 6 人であるため，ジェスチャセットは 6 つ存在する．実験はジェスチャセットごとに行うためユーザ依存となる．それぞれのジェスチャは 10 個ずつあり，学習データをランダムに E 個選ぶ．その際のジェスチャグループの決め方と，ジェスチャグループの学習データ間の類似度の計算方法は，〜節に示したとおりである．学習データを追加し終わった後，入力データを残りの 10 - E 個のジェスチャからランダムに 1 個選び，認識率と認識速度を測定した (10 分割交差検定)．これをそれぞれのジェスチャにつき 100 回行った．本実験は E = 1〜5 とした．また，認識率はジェスチャセットにおける認識率の 100 回平均を学習データごとに測定し，認識速度は，ジェスチャセット内のすべてのジェスチャを 1 回ずつ認識し終わるまでに要した時間の 100 回平均であり，テストに用いるジェスチャをランダムに選ぶ過程は含まれていない．

5.6.7 実験結果

あまり認識率が高くない結果を被験者ごとに載せる
認識速度はそこそこ速い結果を載せる

5.6.8 考察

認識速度についてまず触れる．これは速かった．

認識率が低かった原因を書く．

それぞれの特徴量は，認識に用いられるか用いられないかの二通りに分類され，閾値を設けることにより判別してきたが，同じ名前のジェスチャであったとしても，ランダムに選ばれる学習データによっては，認識に用いられる特徴量が異なる場合があった (閾値によって二通りのいずれかに分類されてしまうため，閾値の設定も難しいといった問題もある)．

また，図〜において，向き，位置が認識に用いる特徴量として選ばる可能性が高いが，向きは位置に比べ，学習データ間において，類似度が小さい組み合わせが存在するため (図 5.2 参照)，向きの方が位置よりも識別するための特徴量として考慮されるべきではないのかという疑問や，それぞれの特徴量による類似度を，同じ尺度において扱うことができるのかという疑問があった (例えば，大きさの類似度 0.9 と向きの類似度 0.9 は，同じくらい類似していると言えるのかなど)．

そこで，これらの問題点に対し，それぞれの特徴量を，認識に用いられるか用いられないかの二通りに分類するのではなく，重み付けを行うことによって解決しようと試みた．例えば，図〜の例を用いる場合，これまでは特徴量を用いるか用いないかであったため，それぞれの特徴量に対する重みの和が 1 となる場合，(大きさ，向き，位置) の重みが (0，0.5，0.5) であったところを，(0.1，0.6，0.3) といった具合にする．このように重みを導入することによって，ジェスチャグループ内の学習データ間の類似度から認識に用いる特徴量を，より高い尤度によって決定することを試みた．

5.7 最適な重み付けのための実験

認識に用いるべき特徴量の選定の方法は，ジェスチャグループの学習データ間の類似度と関係していることは，これまで述べてきた通りである．そこで我々は，認識率を向上させるための最適な重み付けの方法を，我々は実験的に求めた．

重みを決定するまでの手順は次の通りである．

学習データは，追加されるたびに，$1 アルゴリズムによりジェスチャの形状ごとに分類される (図**??**a)．この時，形状の類似度の閾値が 0.85 以上の時，同じ形状とみなした．これは $1 アルゴリズムにて，類似度の N-best List [0..1] の，１番目と２番目の差が 0.2 以上であることを利用している (図**??**)．ここで，類似度の N-best List [0..1](最大値を 1，最小値を 0 としてマッピング) とは，あるテストデータが，全ての種類の学習データに対してどれくらい類似しているかを示すものであり，１番目と２番目の差が大きいほど識別性能が高いことを示す．

その後テストデータを入力し，まず$1 アルゴリズムによりどの形状のジェスチャであるかを判定する．その後「大きさ，向き，位置」のそれぞれの特徴量に対する重みを，それぞれの和が 1 となるようにそれぞれ 0.1 ずつ変化させていく．それぞれの重みについて，その形状のジェスチャ内の学習データとのそれぞれの類似度を求め，重みを類似度に乗算することによって，最終的な類似度を求める．類似度の平均値が 0.9 以上，N-best List の１番目と２番目の差が 0.2 以上の時のそれぞれの特徴量に対する重みを記録する．(図**??**b)

また，学習データを追加する際，並行して，それぞれの形状ごとに，学習データ間の類似度を求める．類似度は，それぞれの学習データの「大きさ，向き，位置」の特徴量を元に求められ，それぞれ特徴量について最小となった類似度を，その形状のジェスチャの学習データ間の類似度とする．例えば図**??**a 右の場合,「大きさ」は，どの学習データのペアを取ってもさほど変わらないので 0.7,「向き」は正反対の学習データのペアが存在するので 0.1,「位置」は 1 つの学習データが他の学習データと比べ比較的異なる位置にあるため，0.5 といった具合である．最小となる類似度を採用する理由は，学習データ内において，1 つでも他の学習データと大きく異なる特徴量を持つ学習データが存在すれば，その特徴量を後に認識の際に考慮すべき特徴として扱うべきであると考えたからであり，今後実験によって検討すべき点でもある．

そして，それぞれの形状ごとの学習データ間の類似度と，先ほど記録した重みを比較することによって，どのような特徴を持つ学習データの場合に，どのような重み付けをすることが望ましいかを考察する．

$$Sf = Scs \times Ws + Sco \times Wo + Scp \times Wp \tag{5.6}$$

5.7.1 実験結果

5.8 ジェスチャの認識

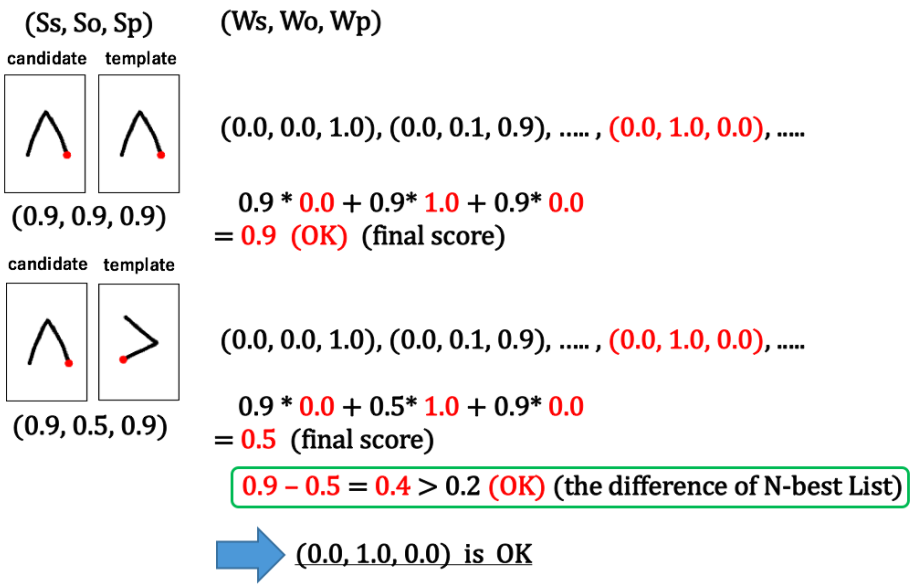

図 5.4: The procedure to decide the optimal weight values

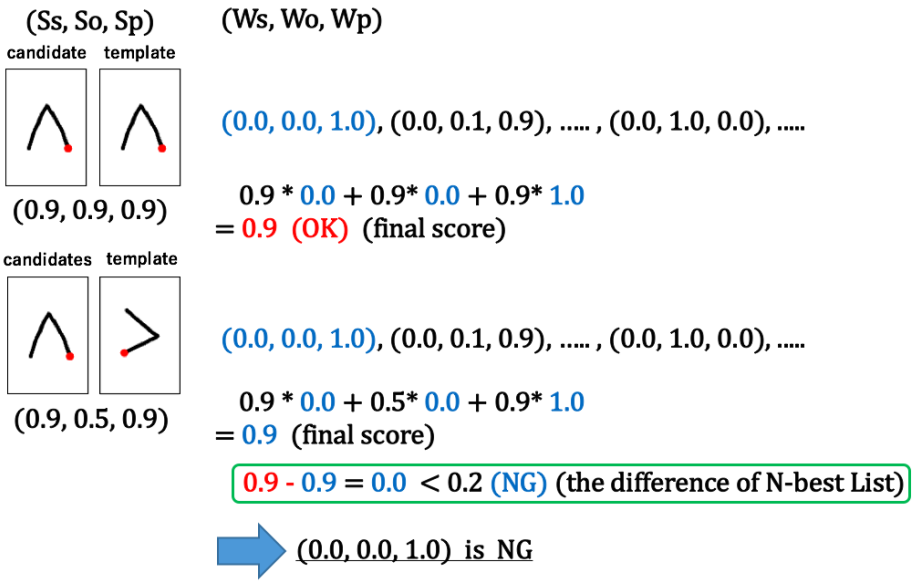

図 5.5: The procedure to decide non optimal weight values

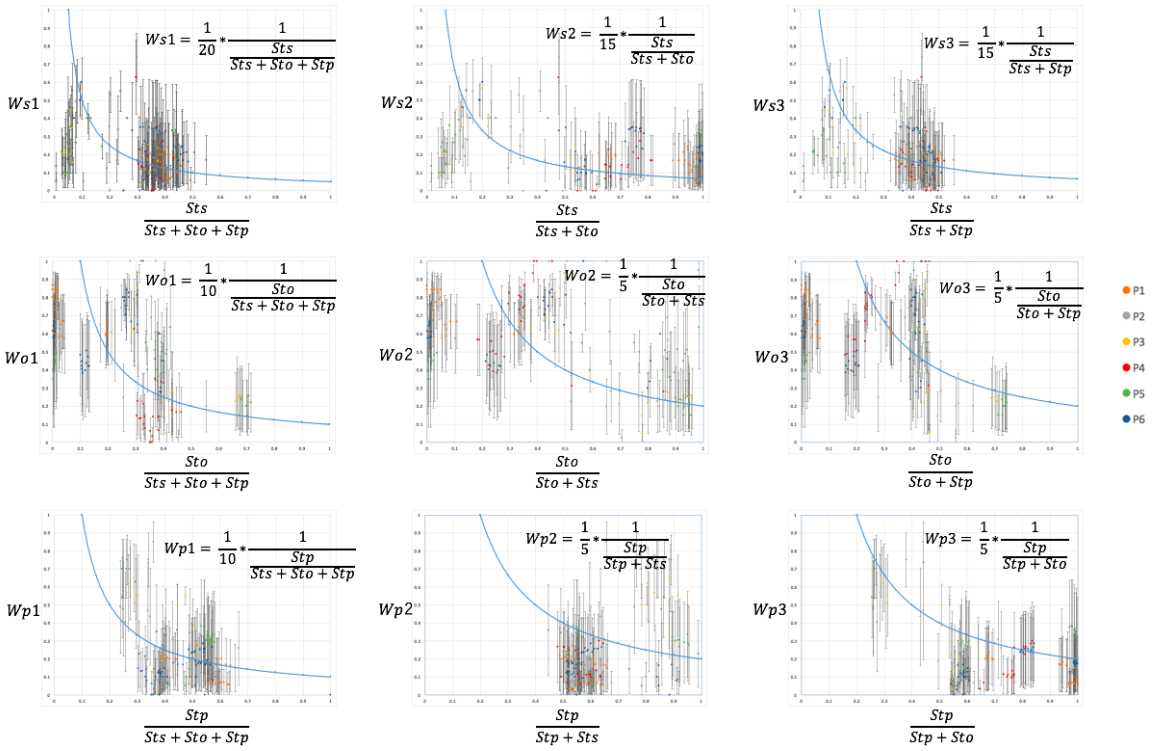

図 5.6: The result of the experiment to find the optimal weight values, and blue lines indicates power approximation curves.

第6章　評価実験

第7章　アプリケーション例

第8章　議論

第9章　結論

謝辞

参考文献

[ABS04] Derek Anderson, Craig Bailey, and Marjorie Skubic. Hidden Markov Model Symbol Recognition for Sketch-based Interfaces. 2004.

[AW10] Lisa Anthony and Jacob O. Wobbrock. A Lightweight Multistroke Recognizer for User Interface Prototypes. In *Proceedings of Graphics Interface 2010*, GI '10, pp. 245–252, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[AW12] Lisa Anthony and Jacob O. Wobbrock. $N-protractor: A Fast and Accurate Multistroke Recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, pp. 117–120, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[BNLH11] Andrew Bragdon, Eugene Nelson, Yang Li, and Ken Hinckley. Experimental Analysis of Touch-screen Gesture Designs in Mobile Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 403–412, New York, NY, USA, 2011. ACM.

[CB05] Xiang Cao and Ravin Balakrishnan. Evaluation of an on-line adaptive gesture interface with command prediction. In *Proceedings of Graphics Interface 2005*, GI '05, pp. 187–194, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[Cho06] Mi Gyung Cho. A New Gesture Recognition Algorithm and Segmentation Method of Korean Scripts for Gesture-allowed Ink Editor. *Information Sciences*, Vol. 176, No. 9, pp. 1290–1303, May 2006.

[HHN90] Tyson R. Henry, Scott E. Hudson, and Gary L. Newell. Integrating Gesture and Snapping into a User Interface Toolkit. In *Proceedings of the 3rd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*, UIST '90, pp. 112–122, New York, NY, USA, 1990. ACM.

[HL00] Jason I. Hong and James A. Landay. SATIN: A Toolkit for Informal Ink-based Applications. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00, pp. 63–72, New York, NY, USA, 2000. ACM.

[HS12] J. Herold and T. F. Stahovich. The 1&Cent; Recognizer: A Fast, Accurate, and Easy-to-implement Handwritten Gesture Recognition Technique. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pp. 39–46, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[HTH00] Pengyu Hong, Matthew Turk, and Thomas S. Huang. Constructing Finite State Machines for Fast Gesture Recognition. In *In Proc. 15th ICPR*, pp. 691–694, 2000.

[KS05] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Comput. Graph.*, Vol. 29, No. 4, pp. 501–517, August 2005.

[KZ04] Per-Ola Kristensson and Shumin Zhai. Shark2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pp. 43–52, New York, NY, USA, 2004. ACM.

[Li10] Yang Li. Protractor: A Fast and Accurate Gesture Recognizer. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 2169–2172, New York, NY, USA, 2010. ACM.

[LM93] James A. Landay and Brad A. Myers. Extending an Existing User Interface Toolkit to Support Gesture Recognition. In *INTERACT '93 and CHI '93 Conference Companion on Human Factors in Computing Systems*, CHI '93, pp. 91–92, New York, NY, USA, 1993. ACM.

[LNHL00] James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pp. 510–517, New York, NY, USA, 2000. ACM.

[MMM+97] Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan S. Ferrency, Andrew Faulring, Bruce D. Kyle, Andrew Mickish, Alex Klimovitski, and Patrick Doane. The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Transactions on Software Engineering*, Vol. 23, No. 6, pp. 347–365, June 1997.

[Pit91] James A. Pittman. Recognizing Handwritten Text. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pp. 271–275, New York, NY, USA, 1991. ACM.

[PS00] Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 22, No. 1, pp. 63–84, January 2000.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (2Nd Edition): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[Ret94] Marc Rettig. Prototyping for Tiny Fingers. *Communications of the ACM*, Vol. 37, No. 4, pp. 21–27, April 1994.

[RSH11] J. Reaver, T. F. Stahovich, and J. Herold. How to Make a Quick$: Using Hierarchical Clustering to Improve the Efficiency of the Dollar Recognizer. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pp. 103–108, New York, NY, USA, 2011. ACM.

[Rub91a] Dean Rubine. Specifying Gestures by Example. *SIGGRAPH Computer Graphics*, Vol. 25, No. 4, pp. 329–337, July 1991.

[Rub91b] Dean Rubine. Specifying Gestures by Example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pp. 329–337, New York, NY, USA, 1991. ACM.

[SC07] Stan Salvador and Philip Chan. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis*, Vol. 11, No. 5, pp. 561–580, October 2007.

[SD05] Tevfik Metin Sezgin and Randall Davis. Hmm-based Efficient Sketch Recognition. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, pp. 281–283, New York, NY, USA, 2005. ACM.

[SMSJ+15] Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. Exploring User-Defined Back-Of-Device Gestures for Mobile Devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pp. 227–232, New York, NY, USA, 2015. ACM.

[Tap82] C. C. Tappert. Cursive Script Recognition by Elastic Matching. *IBM Journal of Reseach and Development*, Vol. 26, No. 6, pp. 765–771, November 1982.

[TL15] Eugene M. Taranta, II and Joseph J. LaViola, Jr. Penny Pincher: A Blazing Fast, Highly Accurate $-family Recognizer. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, pp. 195–202, Toronto, Ont., Canada, Canada, 2015. Canadian Information Processing Society.

[TSW90] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 12, No. 8, pp. 787–808, August 1990.

[Vat12] Radu-Daniel Vatavu. User-defined Gestures for Free-hand TV Control. In *Proceedings of the 10th European Conference on Interactive tv and video*, EuroiTV '12, pp. 45–48, New York, NY, USA, 2012. ACM.

[VAW12] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pp. 273–280, New York, NY, USA, 2012. ACM.

[WMW09] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 1083–1092, New York, NY, USA, 2009. ACM.

[WS03] Andrew Wilson and Steven Shafer. Xwand: Ui for intelligent spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pp. 545–552, New York, NY, USA, 2003. ACM.

[WWL07] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pp. 159–168, New York, NY, USA, 2007. ACM.

[ZK03] Shumin Zhai and Per-Ola Kristensson. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pp. 97–104, New York, NY, USA, 2003. ACM.

Algorithm 1 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$

Ensure: $y = x^n$

```
y ⇐ 1
if n < 0 then
    X ⇐ 1/x
    N ⇐ −n
else
    X ⇐ x
    N ⇐ n
end if
while N ≠ 0 do
    if N is even then
        X ⇐ X × X
        N ⇐ N/2
    else {N is odd}
        y ⇐ y × X
        N ⇐ N − 1
    end if
end while
```

Algorithm 2 Calculate $y = x^n$

Require: $n \geq 0 \vee x \neq 0$

Ensure: $y = x^n$

```
y ⇐ 1
if n < 0 then
    X ⇐ 1/x
    N ⇐ −n
else
    X ⇐ x
    N ⇐ n
end if
while N ≠ 0 do
    if N is even then
        X ⇐ X × X
        N ⇐ N/2
    else {N is odd}
        y ⇐ y × X
        N ⇐ N − 1
    end if
end while
```