

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）

指導教員 志築 文太郎

2017年3月

概要

タッチパネルの普及により、ペンや指を用いた手書きジェスチャを入力として用いるアプリケーションが多く開発されている。手書きジェスチャを認識するための既存アルゴリズムを使うことは専門的な知識が必要であったり、認識に必要な学習データの数が膨大であったりする。 $\$1$ はこれらの問題を解決した手書きジェスチャ認識のためのアルゴリズムの 1 つであり、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う特徴を持ち、様々な開発環境において実装してきた。しかしながら、ユーザ調査により、アプリケーションユーザは、手書きジェスチャの形状や書き順が同じでも、大きさ、向き、位置の違いを利用したジェスチャを入力したいという要望があることがわかった。 $\$1$ を始め $\$1$ を改良した多くのアルゴリズムは、認識率や認識速度の低下を危惧し、ユーザが定義するこれらの手書きジェスチャを識別できるように実装されていない。そこで本研究にて、認識率や認識速度の低下を最小限に抑えつつ、手書きジェスチャの形状や書き順が同じでも、大きさ、向き、位置に関して識別可能なアルゴリズム $\$V$ を開発した。 $\$V$ は $\$1$ を拡張し、保管されている学習データをもとに、類似度を計算するジェスチャを選ぶことによって認識速度の低下を抑え、かつ認識に用いる特徴量に重み付けすることによって、認識率の低下を抑えた。また、 $\$1$ に簡単な数式からなるアルゴリズムを加えるだけであるため、アプリケーション開発者は、本論文において示されるアルゴリズムを自身のシステムに組み込むことにより、大きさ、向き、位置の違いを利用した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容易に開発することができる。また、 $\$V$ の性能評価のために既存アルゴリズムとの比較実験を行い、認識率、認識速度及びアルゴリズムとしての識別性能において高いパフォーマンスを示し、 $\$V$ の有用性を示した。

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 Programming by Example	6
2.4 \$-Family Recognizer	7
2.5 手書きジェスチャ認識を可能にするツールキット	8
2.6 手書きジェスチャの評価研究	8
2.7 本研究の位置づけ	9
第3章 ユーザ調査	10
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	10
3.2 被験者	10
3.3 調査手順	10
3.4 調査結果	11
第4章 \$1 アルゴリズム	14
4.1 特徴	14
4.2 \$1 アルゴリズムの4つのステップ	15
4.2.1 リサンプル	15
4.2.2 向きと大きさの正規化	16
4.2.3 位置の正規化	17
4.2.4 類似度を高くするための最適な角度の選定	17
4.3 類似度計算	19
4.4 制約	19
第5章 \$V アルゴリズム	21
5.1 \$V アルゴリズムが目指す特徴	21

5.2	\$V アルゴリズムのアイディア	22
5.2.1	大きさ，向き，位置に関して識別可能にする方法	22
5.2.2	1 次元の手書きジェスチャを認識する方法	22
5.2.3	学習データの保持の方法	22
	ジェスチャグループの作成が認識速度の低下を抑える理由	23
	ジェスチャグループの作成が認識率の低下を抑える理由	24
	ジェスチャグループごとの特徴量の選定	25
5.3	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	26
5.3.1	ジェスチャの特徴量と類似度の定義	26
	大きさ	26
	向き	27
	位置	27
5.3.2	ジェスチャグループの作成方法	28
5.3.3	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	29
5.3.4	ジェスチャの認識方法	31
5.3.5	被験者	31
5.3.6	実験機器	32
5.3.7	実験手順	32
	ジェスチャの取得	32
	認識率と認識速度の測定	33
	N-best List の 1 番目と 2 番目のスコアの差と類似度の測定	33
5.3.8	実験結果と考察	33
	認識率	34
	認識速度	35
	N-best List の 1 番目と 2 番目のスコアの差	35
	ジェスチャが正しく認識された時の類似度	36
	ジェスチャが正しく認識された時の類似度の最小値	37
5.3.9	議論	38
5.4	重み付けのための実験	39
5.4.1	重み付けの手順	39
5.4.2	実験結果	42
5.5	重みを用いたジェスチャの認識	46
第 6 章	評価実験	47
6.1	実験設計	47
6.2	実験結果と考察	47
6.2.1	認識率	47
6.2.2	認識速度	49
6.2.3	識別性能の結果	51

第7章 アプリケーション例	54
7.1 手書きジェスチャを利用したアプリケーション開発ツールキット	54
第8章 議論	57
8.1 最適な重み付けの再定義	57
8.2 ユーザに依存しない手書きジェスチャの精度評価	57
8.3 書き順に依存しないアルゴリズムの導入	58
8.4 スマートフォン以外の端末への応用	58
8.5 空中手書きジェスチャへの応用	59
第9章 結論	60
謝辞	61
参考文献	62
付録A \$Vアルゴリズムの擬似コード	69
付録B ユーザ調査により得られた手書きジェスチャ	75

図 目 次

1.1 ジェスチャの形状と書き順は同じであるが，向き (a) (b) (d)，位置 (c)，大きさ (e) 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，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一般的に良く用いられる，単一ストロークからなる手書きジェスチャの例，黒い点は手書きジェスチャの書き始めを示す .	11
3.2 ユーザ調査により得られた手書きジェスチャに関して，ユーザが考案した利用例	12
3.3 調査において6人の被験者 (P1 ~ P6) から得られた手書きジェスチャのうち，抜粋された手書きジェスチャ	13
4.1 複数の点から構成されるストローク．同じストロークでも，入力される速度によって点の数が異なる	15
4.2 リサンプリングの例．点の数が異なる同じジェスチャも，同じ点の数へリサンプリングされる	16
4.3 向きに正規化する例．向きが異なったとしても，同じ向きになるように回転する	16
4.4 大きさに正規化する例．大きさが異なったとしても，同じ大きさになるように変形する	17
4.5 位置に正規化する例．位置が異なったとしても，同じ位置になるように移動する	17
4.6 黄金分割探索によって，極値を求める例	18
4.7 \$1アルゴリズムにおいて4つのステップを経て，入力データと学習データが比較されるまでの流れ	19
4.8 \$1が1次元の手書きジェスチャを認識できない理由．向き及び大きさに正規化することによって，ストロークを構成する各点はまばらになってしまふ .	20
5.1 形状と書き順が同じ手書きジェスチャの学習データが集まった，ジェスチャグループの例	23
5.2 同一ジェスチャグループ (A) のみに対し，大きさ，向き，位置の類似度計算を行う	24

5.3	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	25
5.4	大きさの定義	26
5.5	向きの定義	27
5.6	位置の定義	28
5.7	同じ名前の学習データが追加された時に，それぞれの大きさ，向き，位置の特徴量を $(S, \theta, P), (S', \theta', P')$ とした時の，そのジェスチャの新たな特徴量の求め方	28
5.8	学習データを新たに追加する場面の例	29
5.9	ジェスチャグループの学習データ間の類似度を求める方法，この場合，(大きさ，向き，位置) = (0.8, 0.1, 0.4) となる	30
5.10	実験に用いたスマートフォンのスクリーンショット. 緑色のエリアはジェスチャが入力されるエリア	32
5.11	各手法における，被験者ごとの認識率の平均	34
5.12	各手法における，被験者ごとの認識速度の平均	35
5.13	各手法における，被験者ごとの N-best List の 1 番目と 2 番目のスコア差の平均	36
5.14	各手法における，被験者ごとのジェスチャが正しく認識された時の類似度の平均	37
5.15	各手法における，被験者ごとのジェスチャが正しく認識された時の類似度の最小値の平均	38
5.16	条件を満たす重みが決定されるまでの手順	40
5.17	条件を満たさない重みが決定されるまでの手順	41
5.18	ジェスチャグループ内の学習データ間の類似度と大きさの重みの関係の被験者ごとの結果	43
5.19	ジェスチャグループ内の学習データ間の類似度と向きの重みの関係の被験者ごとの結果	44
5.20	ジェスチャグループ内の学習データ間の類似度と位置の重みの関係の被験者ごとの結果	45
6.1	各手法における，被験者ごとの認識率の平均	48
6.2	\$V, \\$1, Rubine における，被験者ごとの認識速度の平均	49
6.3	DTW における，被験者ごとの認識速度の平均	50
6.4	\$V における，N-best List の 1 番目と 2 番目のスコアの差の平均	51
6.5	\$V における，ジェスチャが正しく認識された時の類似度の平均	52
6.6	\$V における，ジェスチャが正しく認識された時の類似度の最小値の平均	52

7.1 手書きジェスチャを利用したアプリケーション開発ツールキット . (a) まず , ユーザはスマートフォンなどの手書きジェスチャを入力する端末を用いて学習 データを追加することによって , (b)V が実装されたツールキットが追加され た学習データを自動的にジェスチャグループに分類し , ジェスチャグループご とに重みを計算する . (c) 最後にユーザは , アプリケーションにおける処理と 手書きジェスチャを対応付ける	55
7.2 ツールキットを使って開発されたメディアプレイヤの例 . (a) アプリケーショ ンのスクリーンショット , (b) 実際にスマートフォンを用いて入力している場面.	56
8.1 形状が同じであるが書き順の異なる手書きジェスチャの例	58

第1章 序論

本研究にて，ユーザが定義した手書きジェスチャを高速に認識し，かつ容易に実装可能な軽量なアルゴリズムを示す．本章にて，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を述べる．次にその課題を解決すべく本研究の目的を述べ，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タッチパネル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(すなわち複雑な数式やアルゴリズムを用いていないということ)，素早く手書きジェスチャ入力のテストができるここと(すなわちあまり学習データを必要としないこと)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，手書きジェスチャは，入力されるジェスチャの形状が毎回微妙に異な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，单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 といった言語において実装されている．そのため，多くの開発者にとって，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, Li10b, AW12, HS12, VAW12, TL15, PTIL16, Vat12a] 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ーフェースが普及し、それらを用いたアプリケーションの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变になるようなアルゴリズムを採用することにより、不变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。大きさ、向き、位置に関して不变にするということは、手書きジェスチャの書き順が同じであります。大きさ、向き、位置に関して不变にするということは、手書きジェスチャの形狀さえ類似していれば、ジェスチャの大きさ、向きあるいはジェスチャが入力された位置などが多少異なっても、同じ形狀と書き順の手書きジェスチャとして認識されるということである。その結果、認識率の向上が実現された。それらを不变にしない、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、不变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 Classifier [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狀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、手書きジェスチャ認識を実現するため広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狀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、ユーザ調査により、これまで述べたきたような、ジェスチャの形狀と書き順は同じであるが、大きさや向きや位置が異なる单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、ユーザが定義するこれらの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したい開発者は既存の\$-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 classifier 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、单一ストロークからなる手書きジェスチャに対し、ジェスチャの大きさ、向き、位置に関して識別可能としながらも、\$1 と比較し認識率の低下と認識

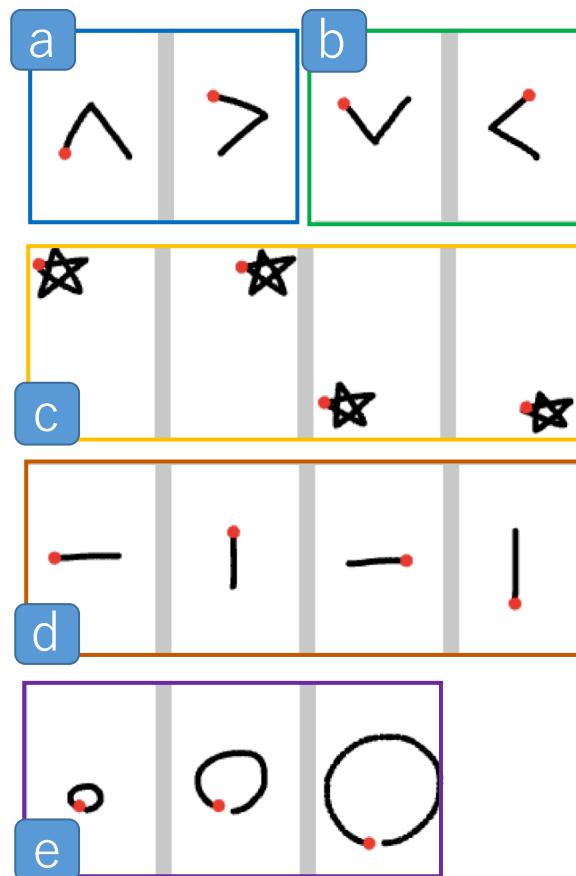


図 1.1: ジェスチャの形状と書き順は同じであるが，向き (a) (b) (d) , 位置 (c) , 大きさ (e) が異なる，単一ストロークからなる手書きジェスチャの例 .

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、ジェスチャの大きさ、向き、位置に関して “V”ariant (不变) な手書きジェスチャを認識する\$-Family Recognizer を\$V と名付けた。\$V は\$1 に簡単な数式からなるアルゴリズムを追加することによって、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す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を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。
- 実装が容易な軽量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を述べる。第4章では、\$V の拡張元である\$1 アルゴリズムを述べ、第5章において、\$V のアルゴリズムの詳細を述べる。第6章では、\$V のアルゴリズムとしての性能評価実験を述べる。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 アルゴリズムの改善点と今後の展望を議論する。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示す。

第2章 関連研究

本章にて、本研究に関連する研究を述べる。本研究では、ユーザが定義する手書きジェスチャを高速に認識し、かつ容易に実装可能な軽量なアルゴリズムを開発している。また、本研究において開発したアルゴリズム\$Vは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。以上を踏まえ、本研究の関連研究を、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、Programming by Exampleに関する研究、\$-Family Recognizerに関する研究、手書きジェスチャ認識を可能にするツールキットに関する研究、手書きジェスチャの評価に関する研究に分類する。本章にて、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してきた。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可能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。すなわち、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、主に画像処理に利用され、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双方においてしばしば

用いられるアルゴリズムである。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、それぞれの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

ユーザ定義の手書きジェスチャを認識できるアルゴリズムのうち、少ない学習データによって高い認識率を示すアルゴリズムを示す。Rubine classifier [Rub91b] 及び 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 classifier はアルゴリズムに用いられる式が複雑である上、学習データが2つの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 classifier は多くの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、それぞれの特徴量が認識結果に及ぼす影響について深い知識がない場合、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、ロバスト性の低下を招くという欠点もある。

2.3 Programming by Example

Programming by Example とは、学習データをもとに、認識モデルを自動生成する手法である。システムに明示的に学習データを例として与え、ユーザの意図や好みを推論することにより、ユーザの意図に沿った処理を行ったり、処理自体を削減したりすることが可能となる [増井 97]。この手法は手書きジェスチャ認識においても活用されている。Taranta ら [TMPL16] は、Gesture Path Stochastic Resampling (GPSR) という、学習データをもとに入力が書くような手書きジェスチャを生成するためのアルゴリズムを考案した。このように、あるデータをもとにそのデータにより近く、より自然な新たなデータを自動生成するアルゴリズムとして、Synthetic Data Generation (SDG) [NFC07, SFC⁺11, GFMDog09, LKJ02, GKN⁺05, RSP12, FVKS13] があり、これらのアルゴリズムは手書きジェスチャを始めとするパターン認識において用いられている。また、Perlin noise [Per85] あるいは、SigmaLognormal [Sig06] も、手書きジェスチャを自動生成するためのアルゴリズムとして広く利用されている。GPSR はユーザから得られたそれぞれの手書きジェスチャの特徴をもとに、そのジェスチャの認識率が高くなるような自動生成方法を計算することによって、これらのアルゴリズムよりも高い認識率を示した。これ

らは，学習データを大量に自動生成することができるため，少ない学習データにおいて高い認識率を示す，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でもある．

2.4 \$-Family Recognizer

\$1 [WWL07] は，\$-Family Recognizer の先駆的なアルゴリズムであり，\$1 を改良したアルゴリズムは，\$-Family Recognizer と呼ばれている．\$1 は，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．また，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，单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ため，手書きジェスチャ認識をプロトタイピングするような環境において実装可能である．\$1 は入力データと学習データ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，位置に不变にしているため，それらの特徴量に依存するようなジェスチャを認識することができない．したがって，そのようなジェスチャを認識するようなアプリケーションを開発したい開発者は\$1 を自身のシステムに採用することができない．\$1 が認識することができないジェスチャを認識するために，これまで\$1 を拡張した\$-Family Recognizer が開発してきた．\$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单一ストロークに分割し，それぞれの单一ストロークを\$1 の手法によって認識した．また，自動的に考えられる複数の单一ストロークの組み合わせ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，最短距離法によるクラスタリングによって，対応する点のユークリッド距離が最小となる入力データと学習データの組み合わせを効率的に探し，認識速度を高速化するアルゴリズムである．Protractor [Li10b] も\$1 に対し，認識速度の面において改良した．最適な角度を探索する際に，Golden Section Search (GSS) [PTVF92] を用いた\$1 とは異なり，閉形式解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，メモリ消費量や認識速度の点において効率的なアルゴリズムである．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，2 次元のストロークからなる手書きジェスチャを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变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．それらを不变にしないことは，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，不变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，入力データと学習データ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，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いている．

2.5 手書きジェスチャ認識を可能にするツールキット

手書きジェスチャ認識を簡単に開発することが可能なツールキットも開発された [HHN90, LM93, MMM⁺97]．SATIN [HL00] はジェスチャ認識の開発を容易にするだけでなく，ペンベースのユーザインタフェースを採用し，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るツールキットである．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．

2.6 手書きジェスチャの評価研究

手書きジェスチャは，これまで様々な研究において入力手法として用いられており，曲線などを用いた複雑な構造からなる手書きジェスチャ [LL11, Li10a, MCvM97, HZS⁺07, AZ09, LGHH08, ZBLF08] や，1本あるいは2本の直線のみからなる単純な構造からなる手書きジェスチャ [KB93] などが用いられている．Bragdon ら [BNLH11] は，これらの研究において用いられる手書きジェスチャのうち，利用頻度の高い手書きジェスチャを抜粋し，それらをスマートフォンやスマートウォッチに対して入力する際に，どのような場面においてどのような手書きジェスチャが求められているのかを評価した．Bragdon らは，アイズフリーによる操作及び歩行時における片手操作のみならず，立ち止まって端末を見ながら操作する場面でさえも，曲線などを用いた複雑な構造からなる手書きジェスチャよりも，1本あるいは2本の直線からなる単純な構造からなる手書きジェスチャの方が，認識率が高く，入力するまでの速度が速いため，ユーザによる利用頻度が高いということを示した．

2.7 本研究の位置づけ

本研究における手書きジェスチャ認識アルゴリズム\$Vは，アルゴリズムが簡潔である\$1に，簡単な数式からなるアルゴリズムを追加するのみによって実装できるため，パターン認識に関する深い知識がなくとも実装可能である上，どのような開発環境においても実装可能である．また，少ない学習データにより高い認識率を示すため，ユーザ定義手書きジェスチャを認識することが可能であり，既存の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と比較し，認識率及び認識速度において高いパフォーマンスを示すアルゴリズムである．

\$Vは，既存の\$-Family Recognizerとは違い，ジェスチャの形状と書き順が同じであるが，大きさ，向き，位置に関して異なる手書きジェスチャを識別することが可能である上，認識率及び認識速度においてパフォーマンスの低下を抑えたアルゴリズムである．また，\$Vは，得られた学習データから，ユーザがそれぞれの手書きジェスチャをどのように区別して登録しているのかを推測する Programming by Example の考え方に基づき，大きさ，向き，位置に関して異なる手書きジェスチャを識別するために必要な特徴量に高い重み付けをするという処理を施している．これにより，ロバスト性が維持され，結果的に認識率の低下を抑えている．また，識別するために必要なジェスチャのみに対し，類似度計算を行うことによって，認識速度の低下を抑えている．

これらに加え，\$Vは大きさ，向き，位置に関して異なる手書きジェスチャを識別することができるため，ユーザは，曲線などを用いた複雑な構造からなる手書きジェスチャを多く考える必要がなく，1本あるいは2本の直線のみからなる単純な構造からなる手書きジェスチャを用いて，大きさ，向き，位置に関して様々なバリエーションを持たせた手書きジェスチャを考えるだけで良いといった利点がある．

第3章 ユーザ調査

本章にて、本研究に取り組む動機にもなった、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を述べる。

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、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一般的に用いられる、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。しかしながら、ユーザ定義ジェスチャを用いた研究[Vat12b, 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、ストロークの形状と書き順は同じであるが、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大学院にてコンピュータサイエンスを専攻している人が4名存在し、残りの2名は、大学にて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

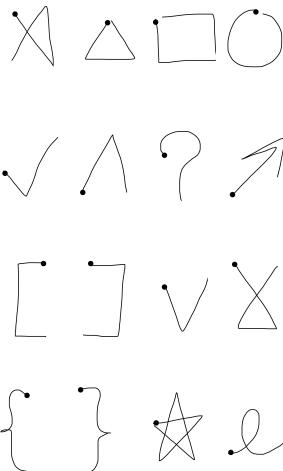


図 3.1: \$1において用いられる，スマートフォンやタブレット端末などのタッチパネルへの手書き入力やペン入力において一般的に良く用いられる，単一ストロークからなる手書きジェスチャの例，黒い点は手書きジェスチャの書き始めを示す．

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) のうちのいずれかになるように指示した．

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．

被験者には，全ての姿勢を座って入力するよう指示した．また，入力ジェスチャは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 B 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

6 人の被験者から得られた手書きジェスチャの一覧は付録 B に記載されている．

これらの結果からわることは，例えば図??において示されるように，音楽プレイヤーにおいて，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

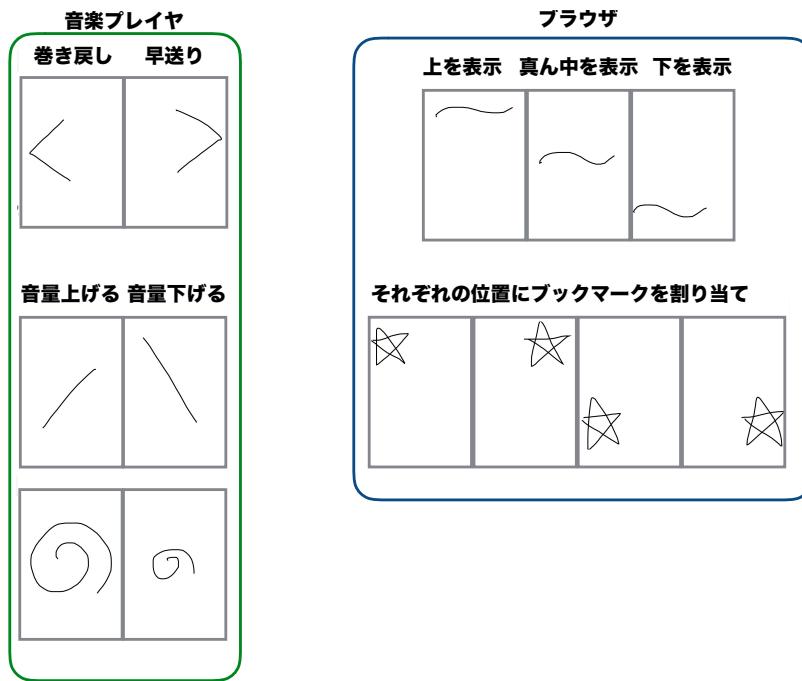


図 3.2: ユーザ調査により得られた手書きジェスチャに関して，ユーザが考案した利用例 .

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，片手で操作する姿勢(姿勢 2)においては，複雑な手書きジェスチャを入力することが困難であるため，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(図??における音楽アプリケーションの音量の上げ下げなど)．

ユーザ調査によって得られた手書きジェスチャは，重複しているジェスチャや\$1によって認識可能なジェスチャが多く含まれていた．本研究は，ジェスチャの形状や書き順は同じであるが，大きさ，向き，位置が異なる手書きジェスチャを識別することが目的であるため，本研究における性能評価実験に用いる手書きジェスチャを，それぞれの被験者において，大きさ，向き，位置が異なる手書きジェスチャができるだけ多く含まれるように，かつジェスチャの数 20 以上となるように抜粋した(図 3.3)．

また，抜粋する際には，それぞれの手書きジェスチャに“STROKE_1”のように番号を順に振り，それぞれの手書きジェスチャを入力する姿勢を，姿勢 1～姿勢 3 の中から 1 つ選んでもらい，手書きジェスチャとともにそれぞれ紙に記入した．

\$V は，図 3.3 に示すような，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アルゴリズムである．

P 1	
P 2	
P 3	
P 4	
P 5	
P 6	

図 3.3: 調査において 6 人の被験者 (P1 ~ P6) から得られた手書きジェスチャのうち，抜粋された手書きジェスチャ .

第4章 \$1アルゴリズム

\$V は \$1 の拡張である。そこで本章にて、\$1 アルゴリズムを述べる。

4.1 特徴

ユーザが入力した手書きジェスチャは、図 4.1 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、入力に用いられ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。それらに加え、人間によって入力される手書きジェスチャに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 4.1 の手書きジェスチャは、入力速度によって手書きジェスチャを構成する点の数が異なる例である。それだけでなく、ジェスチャ自体の大きさや向きも異なる。これらは、手書きジェスチャ認識においてジェスチャを比較する上で、1 つのハドルとなっている。これらのような手書きジェスチャによって生じる問題点に対処しつつ、高い認識率を示し、かつどのような開発環境においても実装可能な簡易的なアルゴリズムを実現するために、\$1 は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、位置に不变な認識をすること。
- 数学的な高度な知識やテクニック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 のような单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 1 番目と 2 番目のスコアの差が大きいほど，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に開発された\$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 \$1 アルゴリズムの 4 つのステップ

4.1 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，向きと大きさの正規化，位置の正規化，類似度を高くするための最適な角度の選定からなる .

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 4.1) .

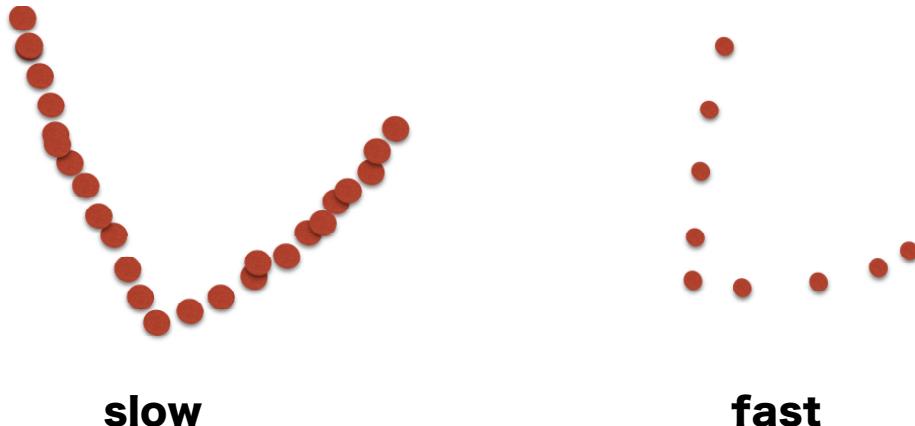


図 4.1: 複数の点から構成されるストローク . 同じストロークでも，入力される速度によって点の数が異なる .

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 4.2 に示すように N 個の等間隔に並ぶ点に

リサンプルする。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。

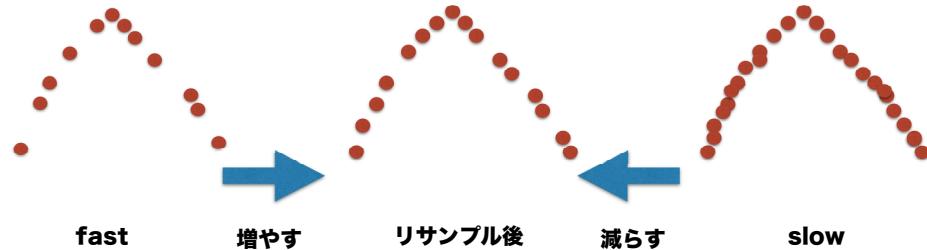


図 4.2: リサンプリングの例。点の数が異なる同じジェスチャも、同じ点の数へとリサンプリングされる。

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]。

また、リサンプルされる点の数 N は、 $32 \leq N \leq 256$ の場合、 $N=64$ の場合に認識率、認識速度双方において高いパフォーマンスが得られることも \$1\$ において報告されている。

4.2.2 向きと大きさの正規化

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 4.3 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 に沿って、全てのジェスチャを回転させる。これにより、ジェスチャは向きが正規化される。

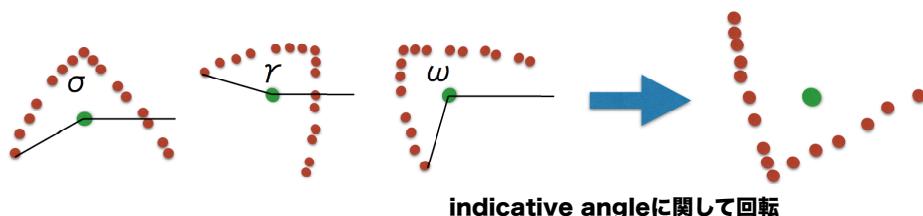
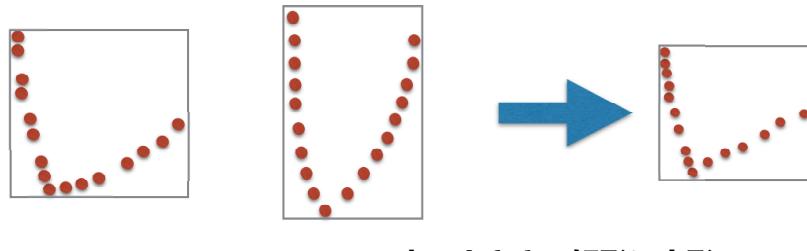


図 4.3: 向きに正規化する例。向きが異なったとしても、同じ向きになるように回転する。

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 4.4 のように、ジェスチャに隣接するような矩形である。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が正規化される。

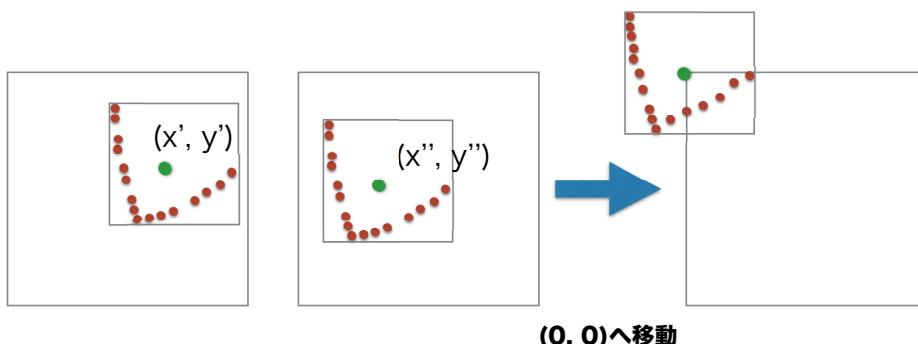


一定の大きさの矩形に変形

図 4.4: 大きさに正規化する例 . 大きさが異なったとしても , 同じ大きさになるように変形する .

4.2.3 位置の正規化

向きと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 (図 4.5 における緑色の点) . 全てのジェスチャについてこの中心座標を $(0, 0)$ へと移動させることによって , 位置が正規化される .



$(0, 0)$ へ移動

図 4.5: 位置に正規化する例 . 位置が異なったとしても , 同じ位置になるように移動する .

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について , 1 つずつ対応する点どうし比較するにあたり , 1° 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] がある . しかしながらこの方法は認識のために膨大な時間を要することになる . 全ての学習データの数が 30 くらいの場合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] を用いることによって最も類似度が高くなるような角度を求める . 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 4.6において、 $f(x)$ の関数があり、極小値 $f(x')$ を求める時に、 x_1 と x_3 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 $f(x_2)$ を求める。この時 x_2 は $(x_2 - x_1) : (x_3 - x_2)$ が黄金比 ($1 : \frac{1+\sqrt{5}}{2}$) 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 x_1, x_2, x_3 ）が存在する。その後広い区間（この場合 x_2 と x_3 の間）において、同様に黄金比によって分割し新たな $f(x_4)$ を得る。この時、 $f(x_{4a})$ ならば、極小値は x_1 と x_4 の間に存在するため、新たな 3 点は x_1, x_2, x_4 となる。 $f(x_{4b})$ ならば、極小値は x_2 と x_3 の間に存在するため、新たな 3 点は x_2, x_4, x_3 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、効率よく極値を求めることができる。

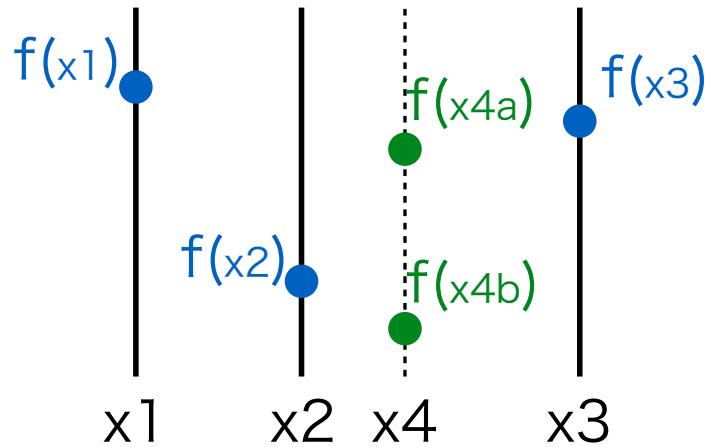


図 4.6: 黄金分割探索によって、極値を求める例。

\$1 の場合、480 個の手書きジェスチャにおいて、 $\pm 45^\circ$ 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° 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存在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 [PJ09, DS94] 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、類似しないジェスチャの場合はおよそ 53.5 ステップ後に極小値が求められることが発見された。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 個のジェスチャにおいて、黄金分割探索は $160 \times 10 = 1600$ ステップ必要であるのに対し、山登り法は $7.2 \times 10 + 53.5 \times 150 = 8097$ 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 4.7 に示す。

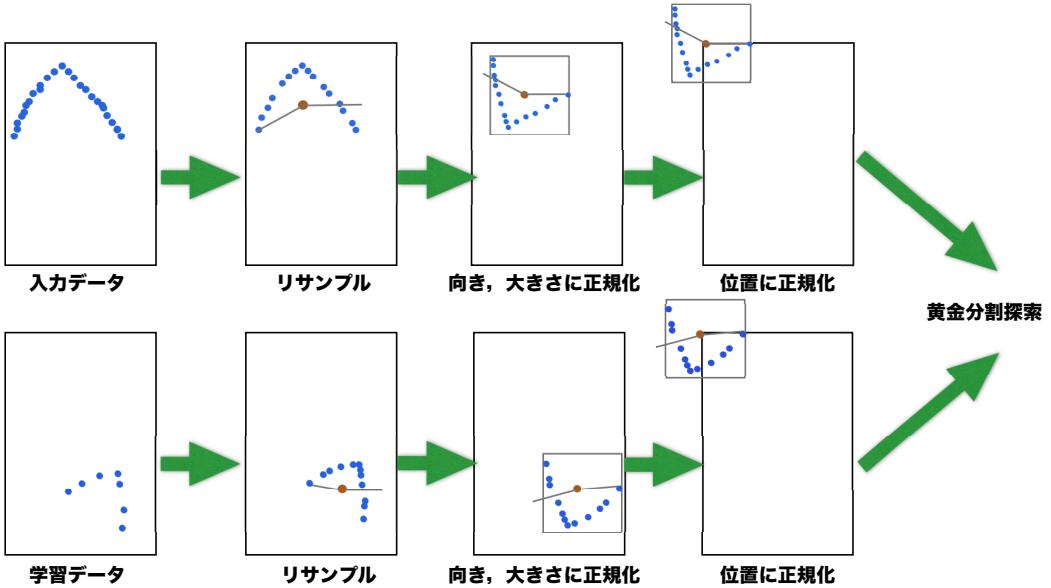


図 4.7: \$1 アルゴリズムにおいて 4 つのステップを経て、入力データと学習データが比較されるまでの流れ。

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は式 (4.1) によって表される。

$$score = 1 - \frac{d}{\frac{1}{2}\sqrt{size^2 + size^2}} \quad (4.1)$$

ここで、 d は対応する点どうしのユークリッド距離の全ての点に関する平均値であり、 $size$ は大きさに正規化するに用いられる矩形(正方形)の 1 辺の長さである。

4.4 制約

\$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、幾つかの制約が存在する。

- 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4.8のように、ストロークを構成する各点が分散してしまうためである。

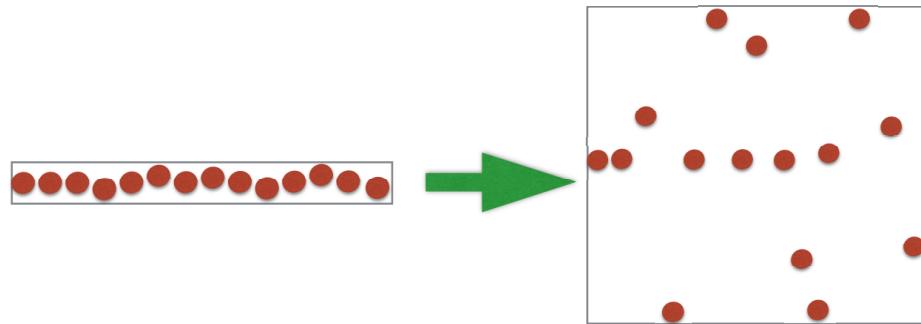


図4.8: \$1が1次元の手書きジェスチャを認識できない理由。向き及び大きさに正規化することによって、ストロークを構成する各点はまばらになってしまう。

\$Vは、これらの問題点を解決したアルゴリズムである。

第5章 \$V アルゴリズム

本章にて，\$1 を拡張した\$V アルゴリズムを述べる．

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．
- 前項目を満たした上で，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，特徴量を不变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．しかしながら，不变にしない場合，つまり，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える．以上を踏まえ，\$1 と比べて，認識率や認識速度が著しく低下することなく，図1.1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 の目指すところである．

5.2 \$V アルゴリズムのアイディア

本節にて，\$V アルゴリズムのアイディアを述べる。

まず，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方法を述べる。次に，\$1において認識できなかった1次元の手書きジェスチャを認識する方法を述べる。そして，\$V アルゴリズムの最大の特徴である，学習データの保持の方法を述べる。

5.2.1 大きさ，向き，位置に関して識別可能にする方法

\$1 アルゴリズムを活用し，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。

5.2.2 1次元の手書きジェスチャを認識する方法

\$V は\$1 とは異なり，1次元の手書きジェスチャを認識できるようにした。これは\$N[AW10]において実装されているため，その手法を用いる。具体的には，\$1において大きさを正規化する際に用いられる，手書きジェスチャに隣接する矩形の縦の長さが横の長さに対し(あるいはその逆が)0.3未満の長さであれば，その手書きジェスチャは1次元の手書きジェスチャであるとみなす，大きさに正規化しない。これにより，1次元の手書きジェスチャを認識することが可能となる。

5.2.3 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 5.1 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)。

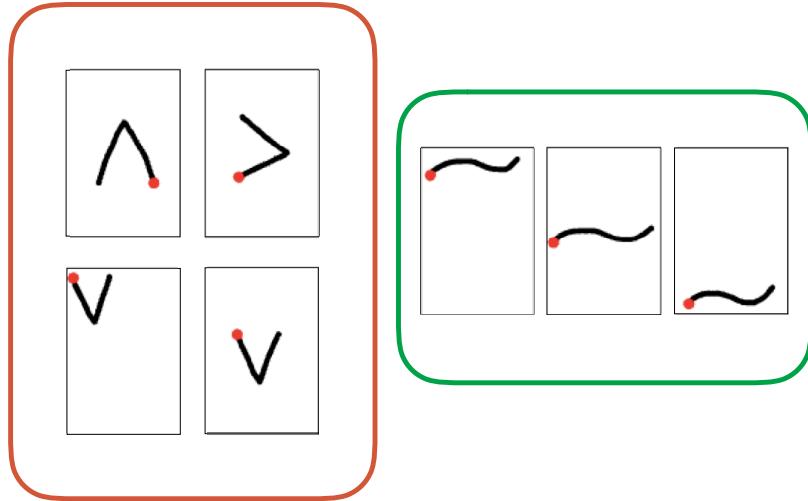


図 5.1: 形状と書き順が同じ手書きジェスチャの学習データが集まった、ジェスチャグループの例。

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。

それぞれについて理由を述べる。

ジェスチャグループの作成が認識速度の低下を抑える理由

大きさ、向き、位置を特徴量として認識に用いる場合、そ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つたジェスチャグループを作成し、入力データと学習データを比較した時に、形状と書き順が同じであった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(図 5.2)。一般的に、認識速度は、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、学習データの数に比例してその差が大きくなるが、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と考えた。以上を踏まえ我々は「ジェスチャグループを作成し、形状と書き順

が同じである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抑えることができる」という仮説を立てた。

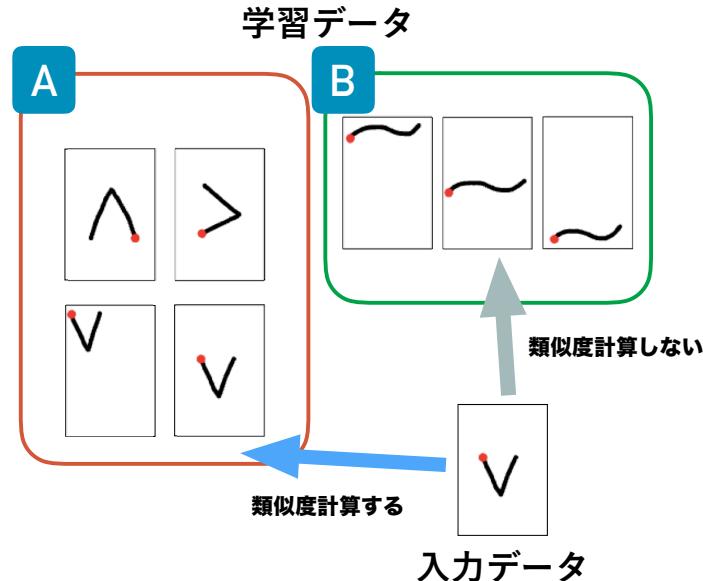


図 5.2: 同一ジェスチャグループ (A) のみに対し、大きさ、向き、位置の類似度計算を行う。

ジェスチャグループの作成が認識率の低下を抑える理由

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 5.3A の場合について考える。入力データ (図 5.3A') が図 5.3A 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が低下し認識率が下がることが考えられる。また、これはロバスト性の低下につながる。

図 5.3B の場合について考える。入力データ (図 5.3B') が図 5.3B 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し認識率が下がることが考えられる。また、これはロバスト性の低下につながる。

図 5.3C の場合について考える。入力データ (図 5.3C') が図 5.3C 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し認識率が下がることが考えられる。また、これはロバスト性の低下につながる。

これまでに述べてきたが、大きさ、向き、位置を認識のための特徴量として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、つまり、認識率の低下を招く恐れがある。図 5.3 はその例である。

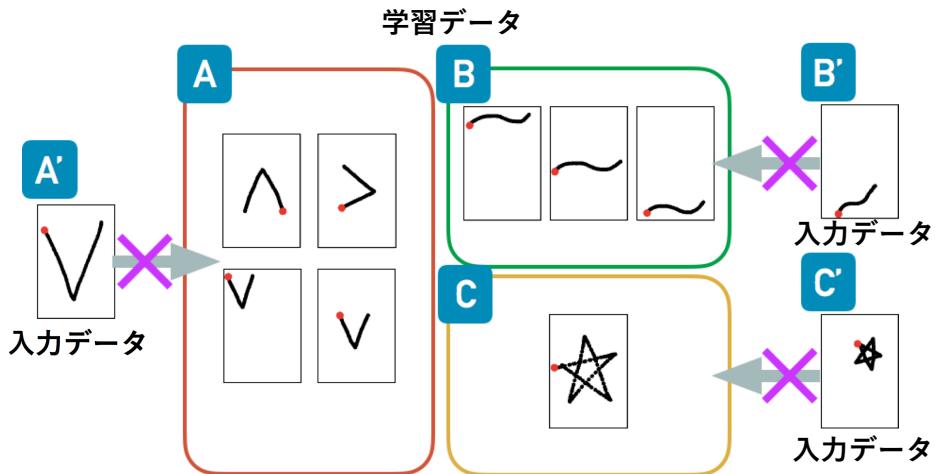


図 5.3: 大きさ , 向き , 位置を特徴量として認識のために用いた場合に , 入力データと学習データが一致しない例 .

そのため , \$V では , 大きさ , 向き , 位置を認識のための特徴量として単に用いるのではなく , 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認識に用いるか選ぶ , つまり , 識別するために必要な特徴量を選ぶという処理を施すことによってロバスト性を維持できる部分を維持する . これにより , 認識率の低下を防ぐことが可能となると考えた .

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を述べる .

\$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 5.3A の場合について考える . 図 5.3A のジェスチャグループには , ジェスチャの大きさはどの学習データも同じであるが , 向きや位置が異なるジェスチャが存在している . つまり , これらを識別するためには , 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 5.3B 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どの学習データも同じであるが , 位置が異なるジェスチャが存在している . つまり , これらを識別するためには , 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 5.3C 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 , 識別のために大きさ , 向き , 位置全て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特徴によって , 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变にし、ロバスト性を維持する。これにより認識率の低下を防ぐことが可能となる。

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、認識率の低下を抑えることができる」という仮説を立てた。

5.3 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

5.2節までにおいて述べてきた、ジェスチャグループを作成し、形状と書き順が同じである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抑える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、認識率の低下を抑える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。本節にてまず、実験を行うにあたって、大きさ、向き、位置、それぞれの特徴量と類似度の定義、ジェスチャグループの作成方法、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をそれぞれ具体的に述べてから、実験の内容を述べる。

5.3.1 ジェスチャの特徴量と類似度の定義

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、それぞれの特徴量と類似度の定義を示す。

大きさ

ジェスチャを構成するストロークの点を内包するように隣接した矩形の面積をジェスチャの大きさとして定義する(図5.4)。そして、2つのジェスチャの類似度 S_S を式5.1によって定義する。

$$S = w * h$$

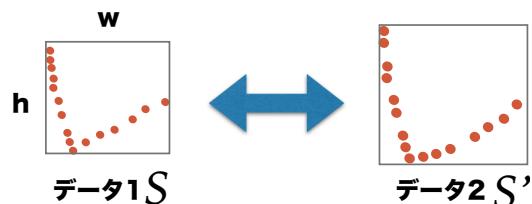


図5.4: 大きさの定義。

$$S_s = \begin{cases} \frac{S'}{S}(S > S') \\ \frac{S}{S'}(S' > S) \end{cases} \quad (5.1)$$

ここで， S はデータ 1 の矩形の面積， S' はデータ 2 の矩形の面積である．

向き

ジェスチャを構成するストロークの始めの点の座標，全サンプル点の中心座標，その中心座標から右横に延長した線(0度)によって形成させる角度すなわち indicative angle をジェスチャの向きとして定義する(図 5.5)．そして，2つのジェスチャの類似度 S_o を式 5.2 によって定義する．

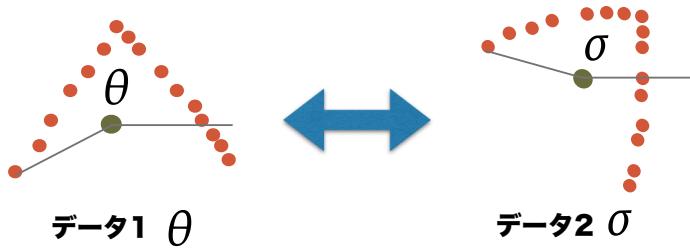


図 5.5: 向きの定義．

$$S_o = 1 - \frac{|\theta - \sigma|}{\pi} \quad (5.2)$$

ここで， θ はデータ 1 の向き， σ はデータ 2 の向きである．

位置

ジェスチャを構成するストロークのすべての点の中心座標をジェスチャの位置として定義する(図 5.6)．そして，2つのジェスチャの類似度 S_p を式 5.3 によって定義する．

$$S_p = 1 - \frac{\sqrt{(x - x')^2 + (y - y')^2}}{\sqrt{Width^2 * Height^2}} \quad (5.3)$$

ここで， (x, y) はデータ 1 の位置， (x', y') はデータ 2 の位置であり， $Width$ は入力領域全体の横， $Height$ は縦の長さである．

また，大きさ，向き，位置の特徴量はそれぞれ，\$1において計算されているものであるため，再利用することによって計算量の増加を抑える．

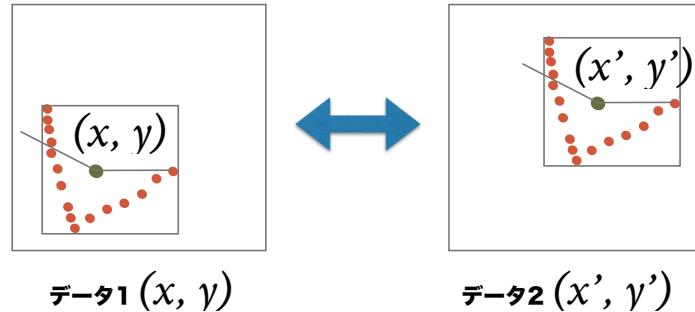


図 5.6: 位置の定義 .

5.3.2 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つたグループである，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 (図 5.7)

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 5.7)．

			$S_{new} = \frac{S + S'}{2}$
(S, θ, P)		(S', θ', P')	$\theta_{new} = \frac{\theta + \theta'}{2}$
$P = (x, y)$		$P' = (x', y')$	$P_{new} = \left(\frac{x + x'}{2}, \frac{y + y'}{2} \right)$

図 5.7: 同じ名前の学習データが追加された時に，それぞれの大きさ，向き，位置の特徴量を (S, θ, P) ， (S', θ', P') とした時の，そのジェスチャの新たな特徴量の求め方．

ここで，図 5.7において， S_{new} ， θ_{new} ， P_{new} は，同じ名前の学習データにおける，大きさ，向き，位置の特徴量を示している．

B. 同じ名前の学習データが存在しない場合

\$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 (図 5.8C)

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(図 5.8A)

そのジェスチャグループに保管される。

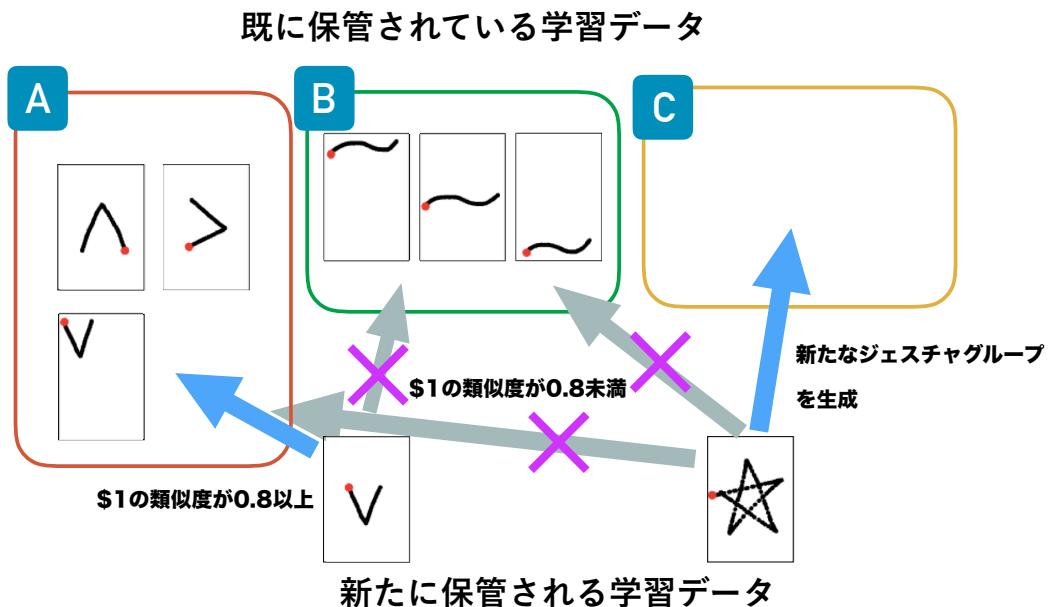


図 5.8: 学習データを新たに追加する場面の例。

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た。これは\$1アルゴリズムにて、類似度の N-best List の 1番目と 2番目のスコアの差が 0.2 以上であることを利用している。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、そのジェスチャが存在するジェスチャグループに保管される。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。

5.3.3 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 5.1 ~ 式 5.3 を用いて求める。

そして、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 5.9 左に示すジェスチャグループの場合、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 $4C2 = 6$ 通り存在する(図 5.9 右)。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、位置の類似度はそれぞれのペアにおいて図 5.9 右に示されている。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(図 5.9 左)。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。

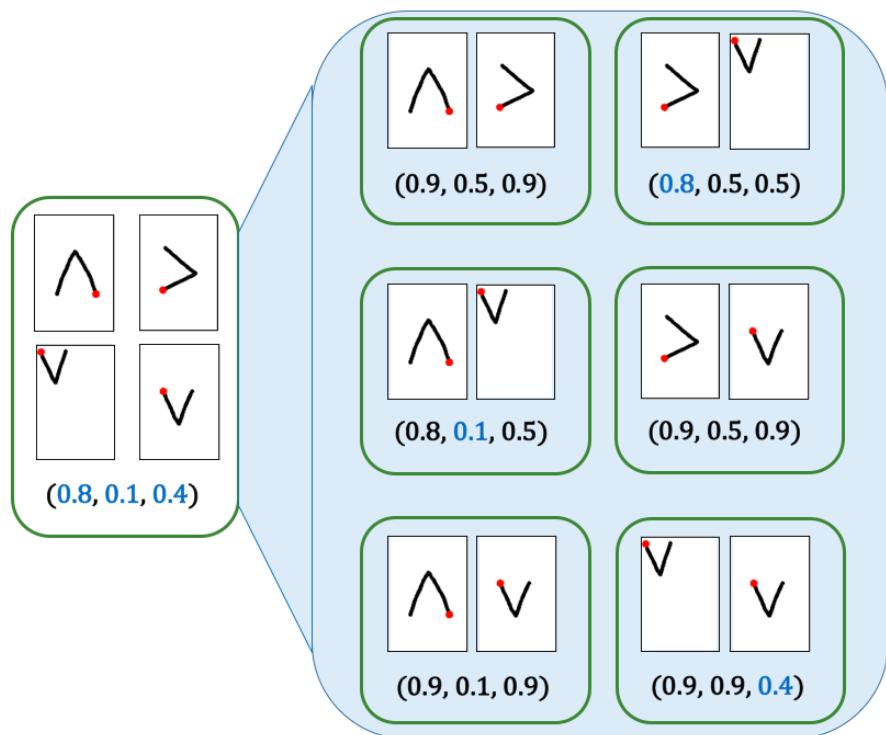


図 5.9: ジェスチャグループの学習データ間の類似度を求める方法、この場合、(大きさ、向き、位置) = (0.8, 0.1, 0.4) となる。

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 0.2 を下回った特徴量を、認識に用いる特徴量として選定した。

5.3.4 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 つである .

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 .
2. 1. によって判別された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 S_{final} は以下の式によって求められる .

$$S_{final} = \frac{S_{cs} \times s + S_{co} \times o + S_{cp} \times p}{n} \quad (5.4)$$

$$s = \begin{cases} 1(S_{ts} < 0.2) \\ 0(else) \end{cases} \quad (5.5)$$

$$o = \begin{cases} 1(S_{to} < 0.2) \\ 0(else) \end{cases} \quad (5.6)$$

$$p = \begin{cases} 1(S_{tp} < 0.2) \\ 0(else) \end{cases} \quad (5.7)$$

ここで , S_{cs} は入力データと学習データの大きさの類似度 , S_{co} はジェスチャグループの入力データと学習データの向きの類似度 , S_{cp} はジェスチャグループの入力データと学習データの位置の類似度を示しており , S_{ts} はジェスチャグループの学習データ間の大きさの類似度 , S_{to} は学習データ間の向きの類似度 , S_{tp} は学習データ間の位置の類似度を示している . また , n は 1 となる s , o , p の数であり , $0 \leq n \leq 3$ を満たす正の整数である . なお , $n = 0$ の時 , S_{final} は計算せず , 手順 1. において , 最も類似度が高いジェスチャを一致するジェスチャとして判別する .

5.3.5 被験者

被験者は , 第 3 章の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) .

5.3.6 実験機器

実験には、入力端末として iPhone5 を用いた。実験における入力領域は $1.94'' \times 3.18''$ であり、解像度は 640×1036 である(図 5.10 における緑色の部分)。

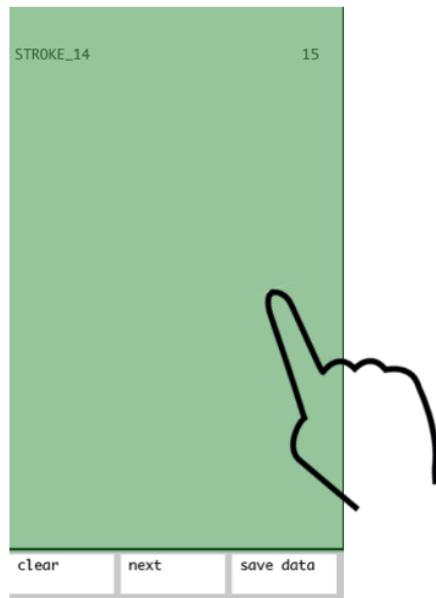


図 5.10: 実験に用いたスマートフォンのスクリーンショット。緑色のエリアはジェスチャが入力されるエリア。

5.3.7 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た手書きジェスチャのうち抜粋された手書きジェスチャについて、手書きジェスチャが記入された紙を見ながら、自身が入力したそれぞれのジェスチャを入力するよう指示した。この、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。ジェスチャは図 5.10 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、そのジェスチャを入力するときの姿勢が紙に書かれているため、それに従い入力するよう指示した。それぞれのジェスチャセットにおいて、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 5.10 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、自身のジェスチャセットに含まれるジェスチャ全てを入力し、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、何度も書き直し可能とした。

認識率と認識速度の測定

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。実験は各被験者が自身のジェスチャセットについてのみ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 個選んだ。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、5.3.2 節及び 5.3.3 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 1 つを認識し終わるまでに要した時間について、ジェスチャセットに含まれる全ジェスチャの平均値の 100 回平均を測定し、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。

N-best List の 1 番目と 2 番目のスコアの差と類似度の測定

認識率と認識速度に加え、類似度の N-best List の 1 番目と 2 番目のスコアの差及びジェスチャが正しく認識された時の、類似度の測定も行う。ここで、類似度の N-best List とは、N 個の学習データそれぞれに対するテストデータとの類似度を降順に並べたものであり、1 番目と 2 番目のスコアの差が大きいほど識別性能が高いことを示す。また、本実験においては、正しく認識されたジェスチャの類似度の平均値の 100 回平均、正しく認識されたジェスチャのうち、類似度が最も低かったジェスチャの 100 回平均も測定した。

5.3.8 実験結果と考察

それぞれの被験者ごとの認識率、認識速度、N-best List の 1 番目と 2 番目のスコアの差、ジェスチャが正しく認識された時の類似度、ジェスチャが正しく認識された時の類似度の最小値を示す。

本実験は、特徴量を選定する手法を用いた場合において、上記についてを測定することが目的であるが、正規化せずにリサンプリングした点のみによってジェスチャを比較する手法及び、ジェスチャグループごとに特徴量を認識に用いるが選定しない手法、つまり、どのジェスチャグループに対しても大きさ、向き、位置の特徴量を類似度に用いる手法についても比較のために測定を行った。

認識率

図 5.11 に各手法ごとの認識率を示す。特徴量を選定する手法及び特徴量を選定しない手法は、どの被験者のジェスチャセットにおいても 80%以上であり、認識率は高かった。リサンプリングのみの場合は、ジェスチャセットによっては認識率は低かった。また、リサンプリングのみの手法は、学習データの数に比例して認識率が高くなつたが、特徴量を選定する手法及び特徴量を選定しない手法において、認識率は必ずしも学習データの数と認識率は比例しなかつた。

これは、リサンプリングのみの手法は、学習データが増えるほど、入力データに類似する学習データが存在する可能性が高くなるが、特徴量を選定する手法及び特徴量を選定しない手法は、同じジェスチャの学習データを追加するたびに、大きさ、向き、位置それぞれの特徴量を算術平均するため、必ずしも入力データに類似する学習データが存在する可能性が高くなるとは言えないからである。

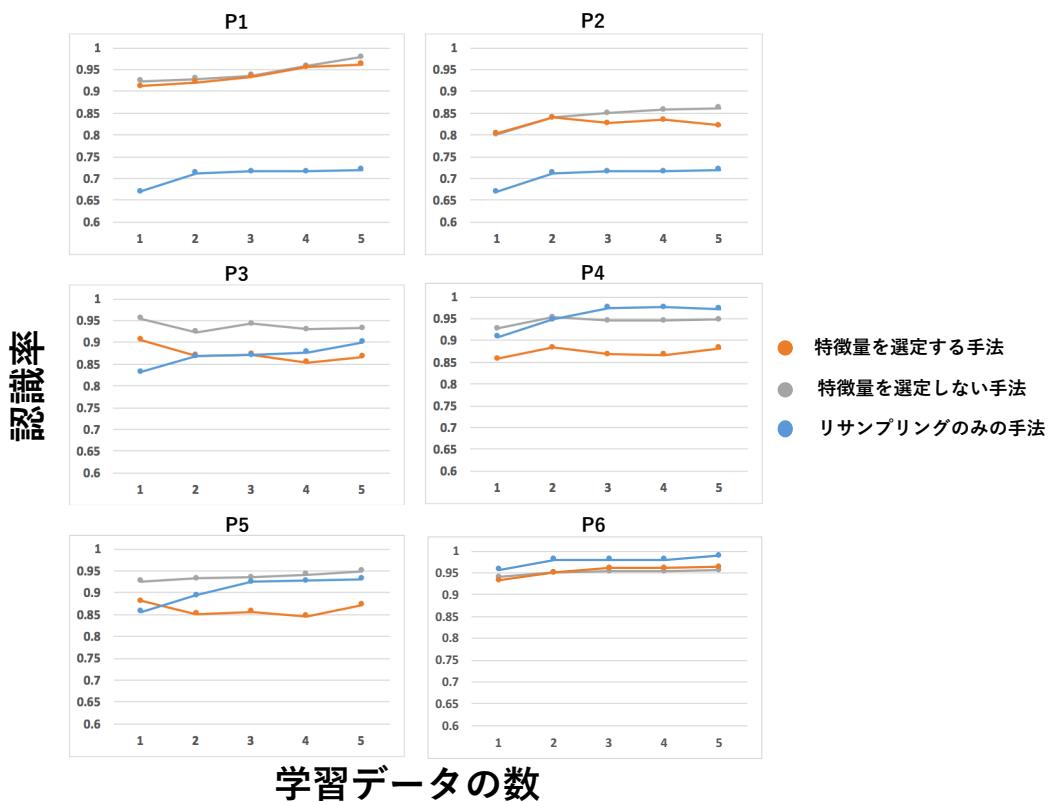


図 5.11: 各手法における、被験者ごとの認識率の平均。

認識速度

図 5.12 に各手法ごとの認識速度を示す。いずれの手法においても、認識速度はほとんど変わらなかった。リサンプリングのみの手法は、正規化処理を行わないため、最も認識速度は速くなった。特徴量を選定しない手法は、正規化したのち、特徴量を選定するための処理を行わないため次に認識速度が速くなった。特徴量を選定する手法は正規化に加え、特徴量を選定する処理を行う必要があるため最も認識速度が遅くなかった。しかしながら、特徴量を選定する手法及び特徴量を選定しない手法の認識速度は、リサンプリングのみの手法と同程度であり、エ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抑えることができるという仮説は正しいと言える。また、どの場合においても認識速度は学習データの数に比例して遅くなった。

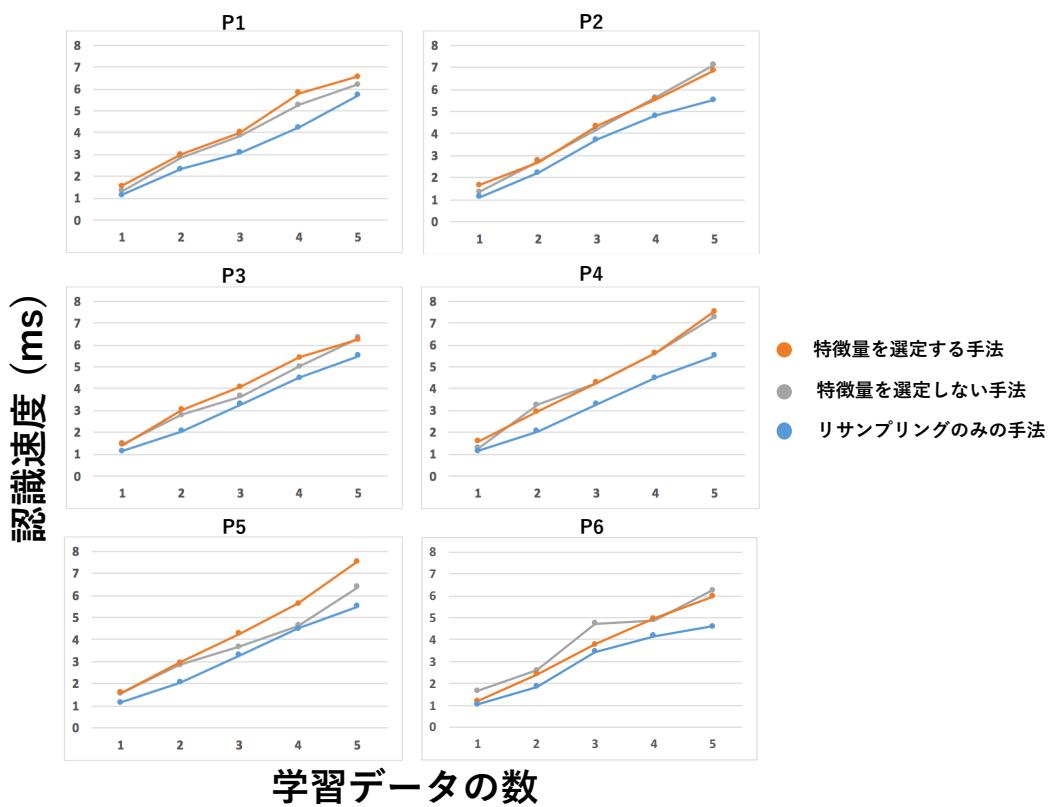


図 5.12: 各手法における、被験者ごとの認識速度の平均。

N-best List の 1 番目と 2 番目のスコアの差

図 5.13 に各手法ごとの N-best List の 1 番目と 2 番目のスコアの差を示す。N-best List の 1 番目と 2 番目のスコアの差は、リサンプリングのみの手法、特徴量を選定する手法、特徴量を選定しない手法の順に大きくなかった。リサンプリングのみの場合においては、正規化しな

いため、類似するジェスチャとそれ以外のジェスチャにおいて、類似度の差が大きくなりやすくなつたと言える。特徴量を選定する手法及び特徴量を選定しない手法は、正規化するため、類似度の差が大きくなりづらいが、特徴量を選定する手法は、ジェスチャグループの学習データ間の類似度が小さい特徴量のみ認識に用いているため、類似度の差が比較的大きくなつたと言える。それに対し、特徴量を選定しない手法は識別に必要のない特徴量も認識に用いるため、例えば、向きが異なるが位置がほぼ同じであるジェスチャを識別しようとする場合に、位置の類似度を加味してしまい、類似度の差が小さくなつたと言える。

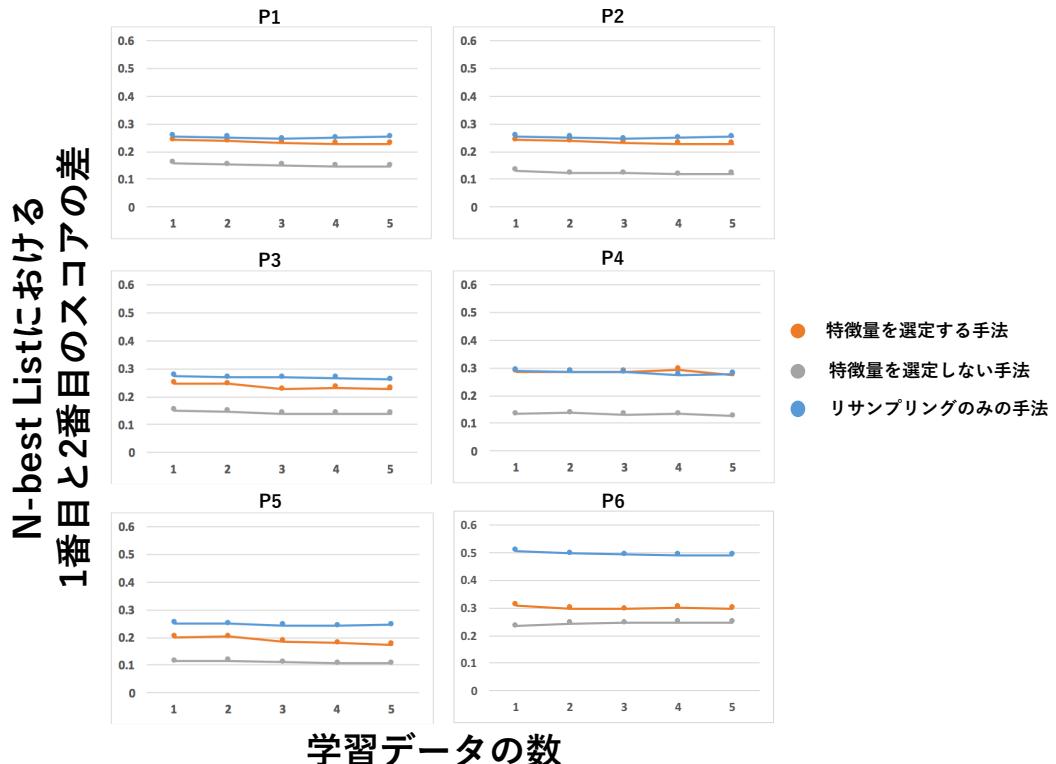


図 5.13: 各手法における、被験者ごとの N-best List の 1 番目と 2 番目のスコア差の平均 .

ジェスチャが正しく認識された時の類似度

図 5.14 に各手法ごとのジェスチャが正しく認識された時の類似度を示す。特徴量を選定する手法及び特徴量を選定しない手法は、ジェスチャが一致した時の類似度の平均値は高くなり、ほぼ同じような結果となった。特徴量を選定しない手法は、特徴量を選定する場合と比べて類似度が小さくなると予想されたが、類似度は高くなつた。これは、本実験がユーザ依存であったため、被験者の書くジェスチャの再現率が高かったことが原因であると言える。リサンプリングのみの手法は被験者の書くジェスチャの再現率が高くとも、正規化しないため類似度が低くなつたと言える。

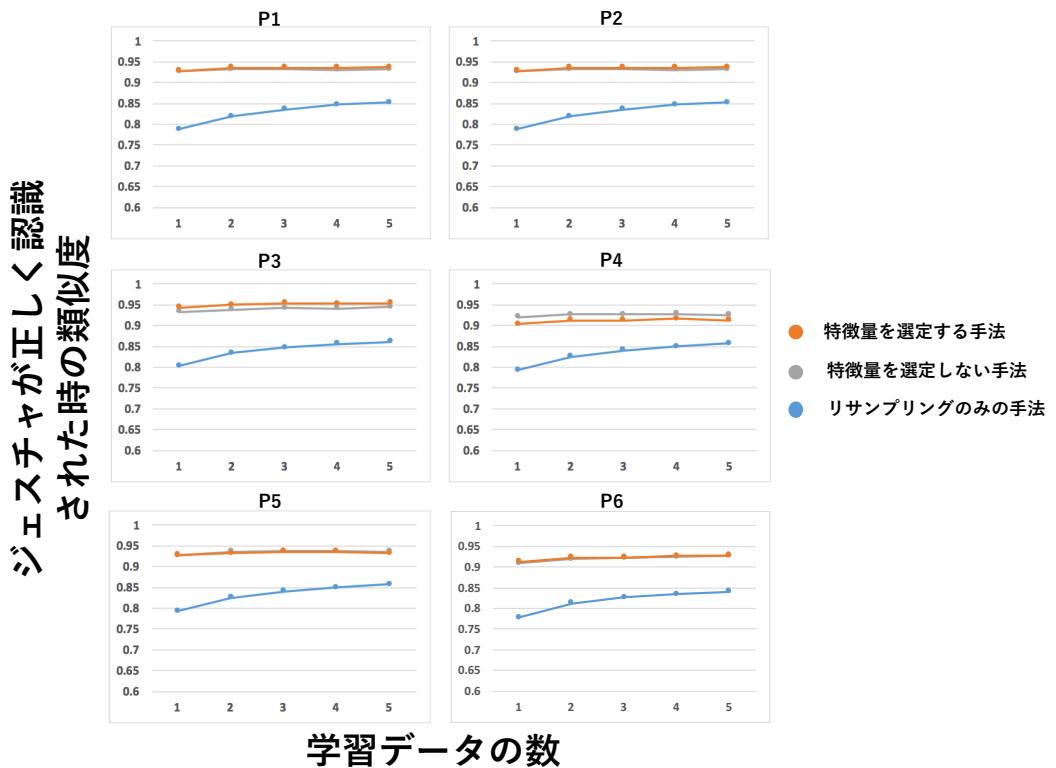


図 5.14: 各手法における、被験者ごとのジェスチャが正しく認識された時の類似度の平均 .

ジェスチャが正しく認識された時の類似度の最小値

図 5.15 に各手法ごとのジェスチャが正しく認識された時の類似度の最小値を示す . 特徴量を選定する手法及び特徴量を選定しない手法は、ジェスチャが一致した時の類似度の最小値は高くなった . また、リサンプリングのみの手法は低くなった . これらの要因は、ジェスチャが一致した時の類似度と同様であると考えられる .

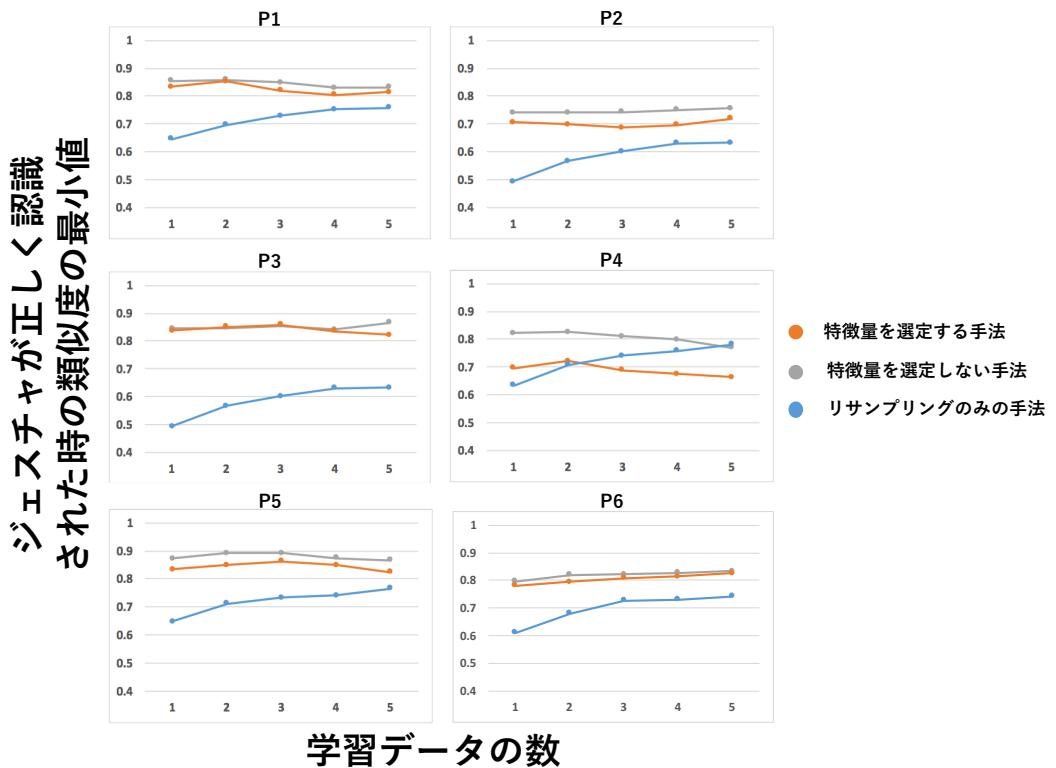


図 5.15: 各手法における、被験者ごとのジェスチャが正しく認識された時の類似度の最小値の平均 .

5.3.9 議論

特徴量を選定する手法及び特徴量を選定しない手法はいずれも、認識率、認識速度のスコアは高く、特徴量を選定する手法は N-best List の 1 番目と 2 番目のスコアの差も大きくなつた。以上を踏まえ、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抑えることができるという仮説及び、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、認識率の低下を抑えることができるという仮説はある程度正しいと言える。それらに加え、特徴量を選定することによって、識別性能が向上するということも言える。

しかしながら被験者によっては、認識率及び N-best List の 1 番目と 2 番目のスコアの差は小さくなつた。この原因について考察する。

特徴量を選定する手法において、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、ランダムに選ばれる学習データによっては、同じジェスチャグループであったとしても、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，例えば図 5.9 に示すジェスチャグループにおいて，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，類似度が小さい組み合わせが存在するため，向きの方が位置よりも識別するための特徴量としてより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，特徴量に重み付けをすることによって解決しようと試みた。例えば，図 5.9 の場合，それぞれの特徴量に対する重みの和が 1 となる場合，これまでには特徴量を用いるか用いないかであったため，（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，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。

そこで，我々は，認識率及び N-best List の 1 番目と 2 番目のスコアの差を向上させる重みを求めることにした。

5.4 重み付けのための実験

あるジェスチャグループが認識に用いるべき特徴量が，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，ジェスチャグループの学習データ間の類似度をもとに，認識率及び N-best List の 1 番目と 2 番目のスコアの差を向上させるための，それぞれの特徴量に対する重み付けの方法を実験的に求めることとした。

5.4.1 重み付けの手順

まず，重み付けの手順を述べる。

学習データは，追加されるたびに，\$1 アルゴリズムを用いてジェスチャグループとして保管される。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，5.3.2 節及び 5.3.3 節に示したとおりである。

全ての学習データを追加し終わった後，入力データを入れていく。まず \$1 アルゴリズムによりどの形態のジェスチャであるかを判定する。これは 5.3.4 節において述べた方法と同じである。その後，ジェスチャグループにおける大きさ，向き，位置のそれぞれの特徴量に対する重みを求める手順へと移行する。その方法を図 5.16，図 5.17 に図示する。まず，入力データと学習データの類似度を求める。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。そして，重みを先ほど求めた類似度に式 5.8 に則って乗算することによって，最終的な類似度を求める。ここで，式 5.8 において， S_{cs} は入力データと学習データの大きさの類似度， S_{co} は入力データと学習データの向きの類似度， S_{cp} は入力データと学習データの位置の類似度を示しており， W_s は

大きさの重み， W_o は向きの重み， W_p は位置の重みを示している．これを各ジェスチャセットに含まれる，全てのジェスチャについて行う．この時，ジェスチャが一致した時の類似度の平均値が 0.9 以上，N-best List の 1 番目と 2 番目のスコアの差が 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．これを各被験者から得られた各ジェスチャセットごとに行う．

このようにして，各ジェスチャセットから得られる，それぞれのジェスチャグループ内の学習データ間の類似度と，重みをセットにして記録することによって，どのような特徴を持つジェスチャグループの場合に，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．

$$S_{final} = S_{cs} \times W_s + S_{co} \times W_o + S_{cp} \times W_p \quad (5.8)$$

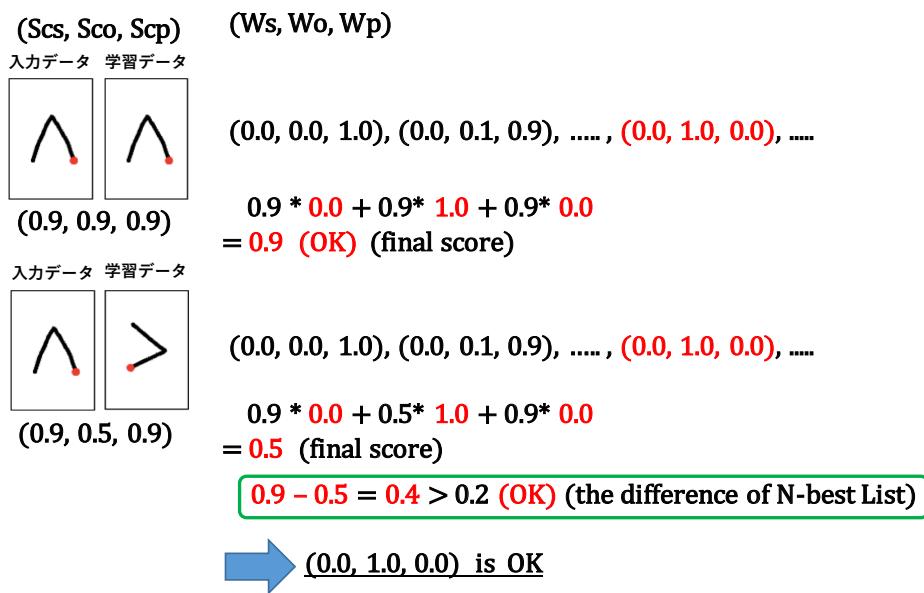


図 5.16: 条件を満たす重みが決定されるまでの手順．

(Scs, Sco, Scp)	(Ws, Wo, Wp)
入力データ	学習データ
$(0.9, 0.9, 0.9)$	$(0.0, 0.0, 1.0), (0.0, 0.1, 0.9), \dots, (0.0, 1.0, 0.0), \dots$
	$0.9 * 0.0 + 0.9 * 0.0 + 0.9 * 1.0$
	$= 0.9 \text{ (OK) (final score)}$
入力データ	学習データ
$(0.9, 0.5, 0.9)$	$(0.0, 0.0, 1.0), (0.0, 0.1, 0.9), \dots, (0.0, 1.0, 0.0), \dots$
	$0.9 * 0.0 + 0.5 * 0.0 + 0.9 * 1.0$
	$= 0.9 \text{ (final score)}$
	$0.9 - 0.9 = 0.0 < 0.2 \text{ (NG) (the difference of N-best List)}$
	$\underline{(0.0, 0.0, 1.0) \text{ is NG}}$

図 5.17: 条件を満たさない重みが決定されるまでの手順 .

5.4.2 実験結果

図 5.18 は、ジェスチャグループ内の学習データ間の類似度と大きさの重みの関係、図 5.19、ジェスチャグループ内の学習データ間の類似度と向きの重みの関係、図 5.20 は、ジェスチャグループ内の学習データ間の類似度と位置の重みの関係の結果を被験者ごとに示している。ここで、 S_{ts} はジェスチャグループの学習データ間の大きさの類似度、 S_{to} はジェスチャグループの学習データ間の向きの類似度、 S_{tp} はジェスチャグループの学習データ間の位置の類似度を示している。

ある類似度の時の条件を満たす重みは複数存在するため、プロットされているデータは、その平均値と標準偏差である。

それについて、我々はジェスチャグループ内の学習データ間の類似度と重みを関係式によって表すこととした。関係式は、それぞれのグラフにおいて青色の線によって示されており、グラフの右上に式が示されている。

この関係式の求め方について述べる。

まず、実験から得られたジェスチャグループ内の学習データ間の類似度と重みの関係の近似式を求める。この時、両対数グラフにおいて、およそ直線で表せられることがわかった。つまり、これらの関係は累乗近似曲線に近似できることがわかった。しかしながら、近似された累乗近似曲線に対し値が離れている元データが多く存在するため、近似式がどれほど元データを表すものになっているかを示す決定係数 R^2 は、どのグラフにおいても低くなつた。

そこで我々は、累乗近似曲線に近似できることを手掛かりに、近似式を以下のように定めた。

$$W = \frac{1}{S} \times \frac{1}{S} \quad (5.9)$$

ここで、W は重み、S は類似度を示す。

そして、 S を 5~30 まで 5 ずつ増やし、その時に求められる重みを元に認識率を測定したところ、図 5.18~図 5.20 において示されるような近似式において、認識率が高くなることがわかった。

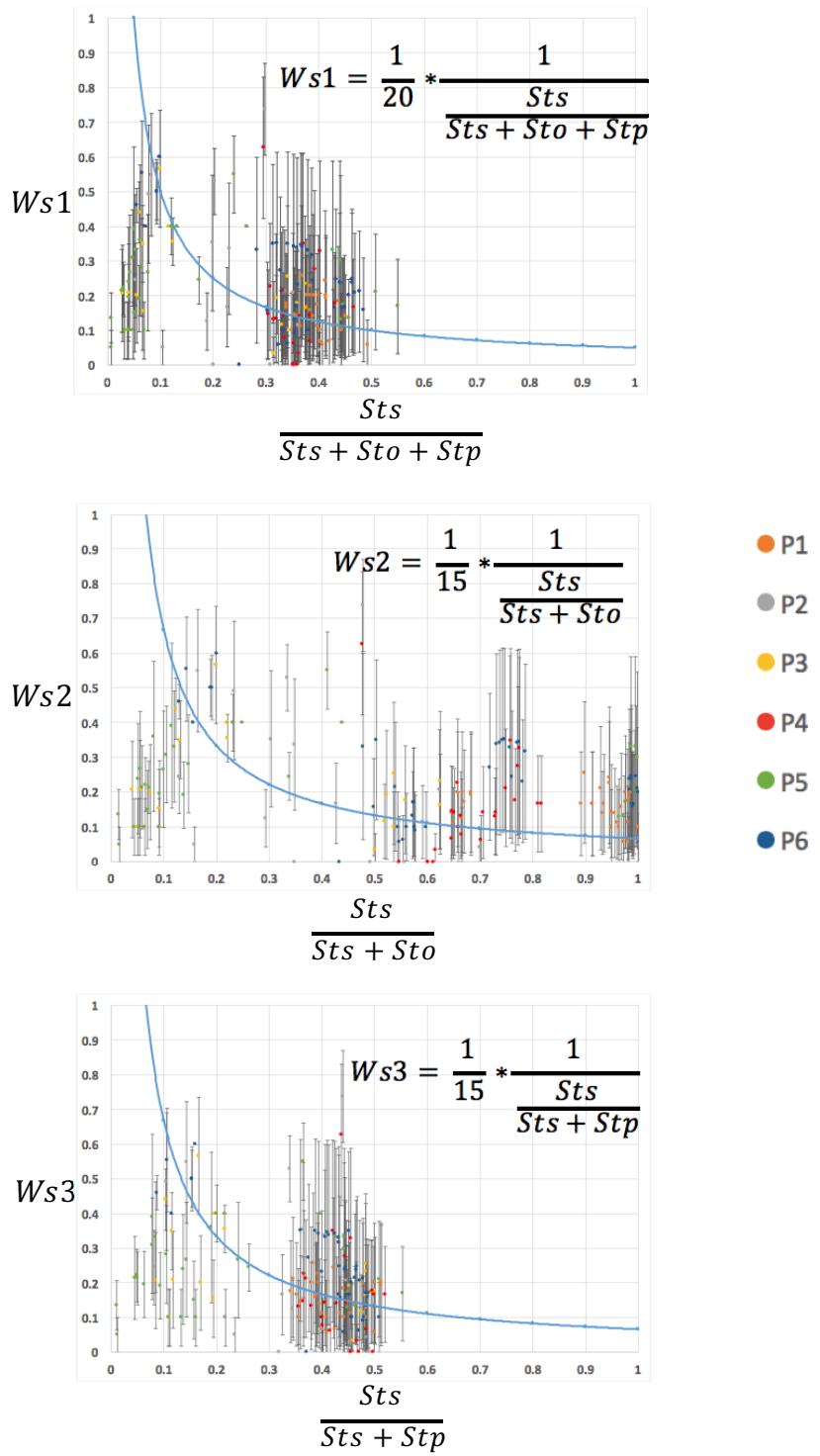


図 5.18: ジエスチャグループ内の学習データ間の類似度と大きさの重みの関係の被験者ごとの結果

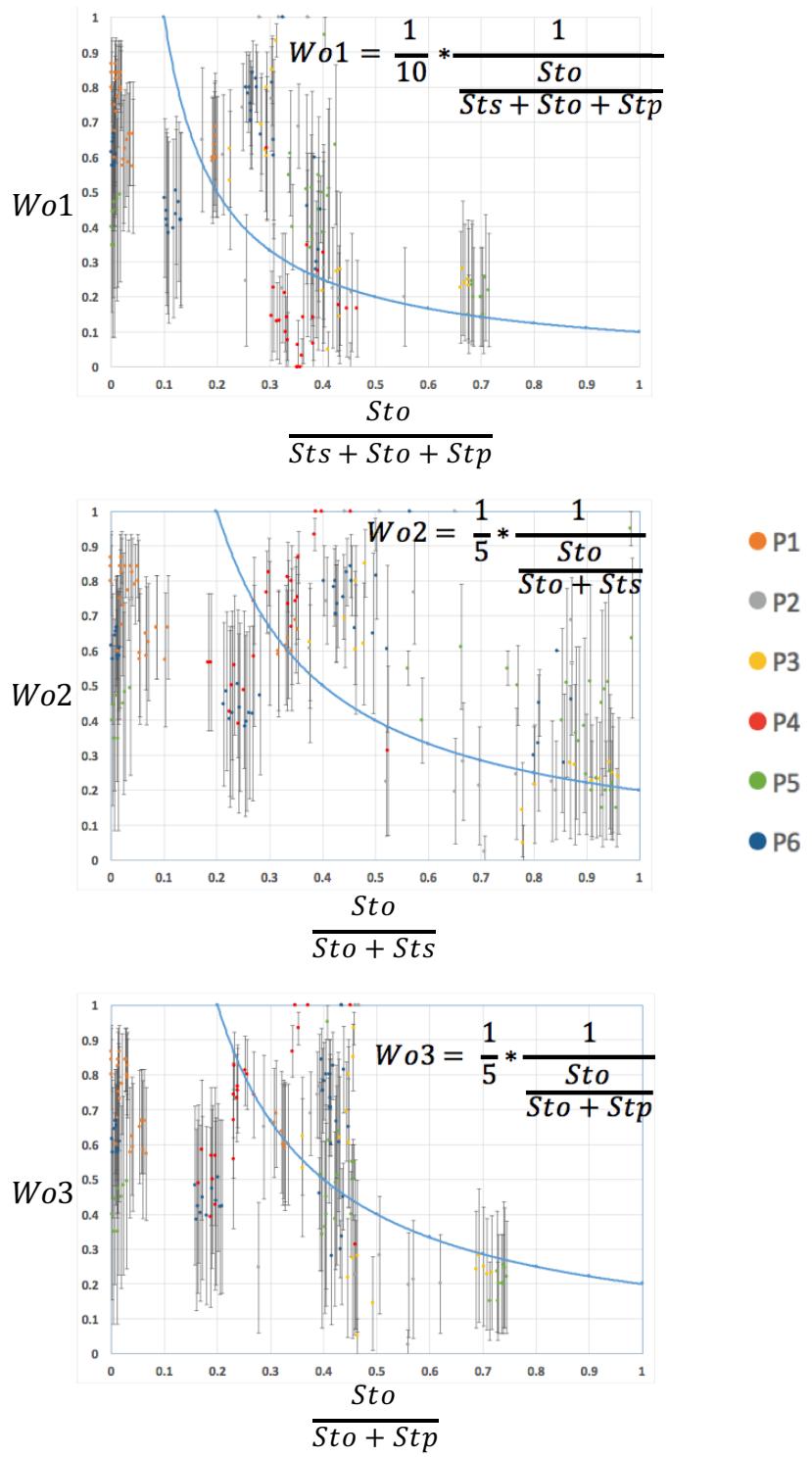


図 5.19: ジェスチャグループ内の学習データ間の類似度と向きの重みの関係の被験者ごとの結果 .

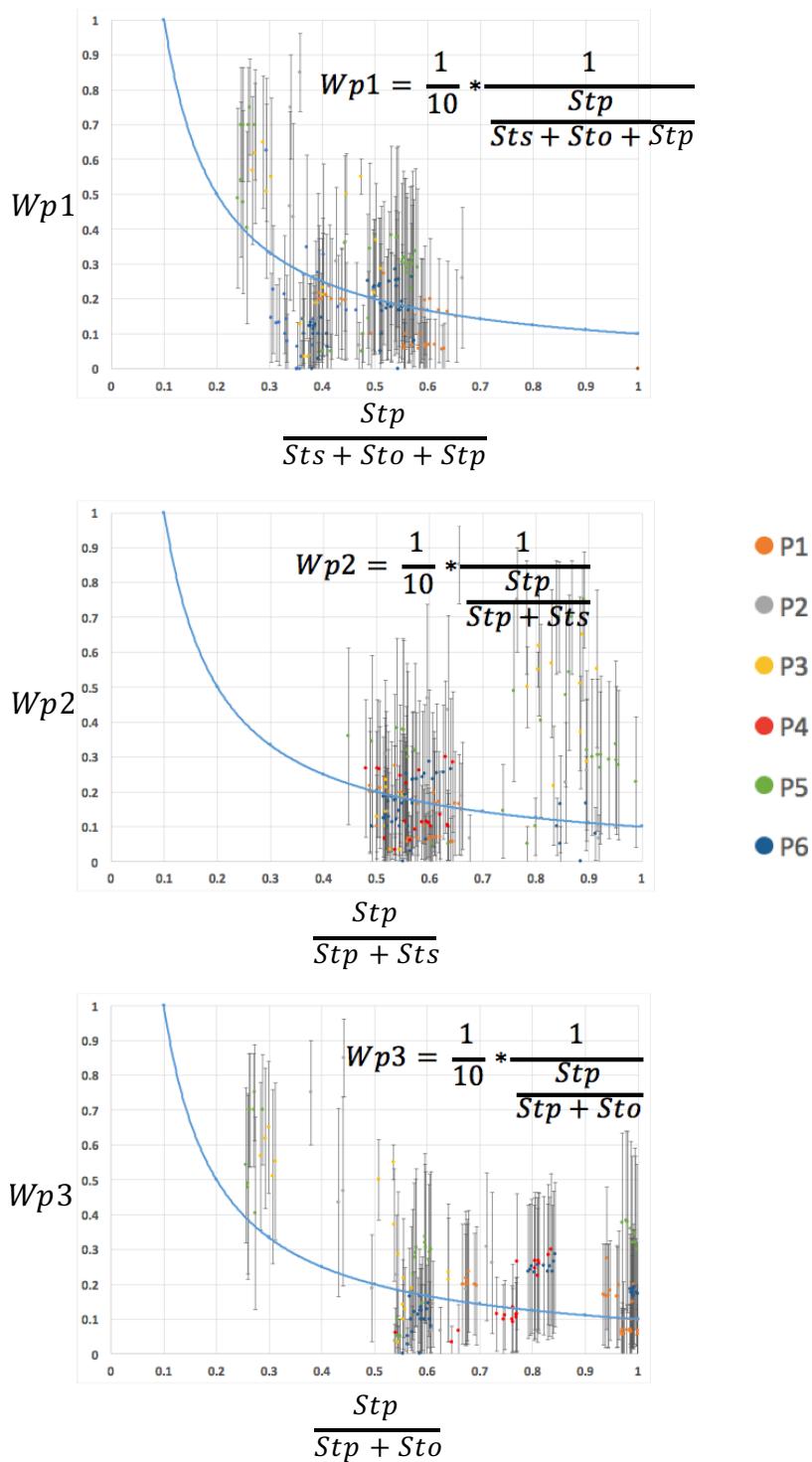


図 5.20: ジェスチャグループ内の学習データ間の類似度と位置の重みの関係の被験者ごとの結果 .

5.5 重みを用いたジェスチャの認識

以上を踏まえ、重みを用いたジェスチャ認識が可能となる。

まず、学習データを追加する際に、ジェスチャグループ内の学習データ間の類似度を元に重みを求める。この際、図 5.18 ~ 図 5.20において示される近似式を元に重みを求める。これは、以下の式に統合される。

$$W_s = \frac{1}{90} (5.5 + 3.5 \frac{S_{to} + S_{tp}}{S_{ts}}) \quad (5.10)$$

$$W_o = \frac{1}{30} (5.0 + 3.0 \frac{S_{ts} + S_{tp}}{S_{to}}) \quad (5.11)$$

$$W_p = \frac{1}{30} (3.0 + 2.0 \frac{S_{ts} + S_{to}}{S_{ip}}) \quad (5.12)$$

ここで、 S_{ts} は学習データ間の大きさの類似度、 S_{to} は学習データ間の向きの類似度、 S_{tp} は学習データ間の位置の類似度を示している。

次に、実際に入力データを認識させる手順を述べる。

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對し類似度を求め、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。
2. 1. によって判別された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。

この際、ジェスチャグループ内において、入力データと学習データの類似度 (S_{cs} , S_{co} , S_{cp}) を求め、学習データを追加した際に式 5.10 ~ 式 5.12 によって求めた重みを用い、式 5.8 によって最終的な類似度を求める。この類似度が最も高かった時のジェスチャが判別されるジェスチャとなる。

この重み付けを考慮したアルゴリズムを、\$V の最終的なアルゴリズムとする。

第6章 評価実験

本章にて，重み付けを考慮した\$Vアルゴリズムの性能評価実験を述べる．

6.1 実験設計

\$Vの認識率，認識速度，類似度のN-best Listの1番目と2番目のスコアの差，ジェスチャが一致した時の類似度，ジェスチャが一致した時の類似度の最小値を測定することによってアルゴリズムの性能を評価した．また，認識率，認識速度に関しては，\$Vの拡張元である\$1及び，\$Vと同様に，大きさ，向き，位置に関して異なる手書きジェスチャを識別可能なRubine，DTWと比較した．

実験に用いたジェスチャは，5章と同様，ユーザ調査によって得られたジェスチャを用いる．また，それぞれの測定方法については，5章において述べた通りに行うが，\$1については，大きさ，向き，位置に関して手書きジェスチャを識別できないため，形状と書き順さえ正しく認識されれば，正しく認識されたとみなした．

6.2 実験結果と考察

6.2.1 認識率

図6.1に各手法ごとの認識率の結果を示す．それぞれの被験者の平均は，\$Vは93% (SD=4.55)，\$1は98% (1.04)，DTWは98% (0.86)，Rubineは88% (2.14)であった．\$Vは\$1とDTWに対し，有意に低かった($p < 0.01$)が，被験者P1，P3，P4，P5，P6に関しては認識率の平均は95% (1.27)と高く，\$1と比べて認識率の低下を最小限に抑えることができた．\$Vは\$1とは異なり，大きさ，向き，位置に関して異なる手書きジェスチャを識別可能であるにもかかわらず，被験者P1，P3，P4，P5，P6に関しては，認識率はおよそ3%しか低下しなかった．Rubineと比較した場合は，P1，P3，P4，P5，P6に関しては\$Vは有意に高かった($p < 0.001$)．また，重み付けをしない場合と比べても，\$Vは全被験者について認識率は向上した．

P2の認識率が低かったのは，P2は別の名前のジェスチャで，形状が酷似したジェスチャが複数存在したため，識別が困難になったことが要因であると考えられる．また，\$Vは学習データに比例して認識率が高くなるとは言えない．これは，同じジェスチャの学習データを追加するたびに，大きさ，向き，位置それぞれの特徴量を算術平均するため，必ずしも入力データに類似する学習データが存在する可能性が高くなるとは言えないからである．しかしながら

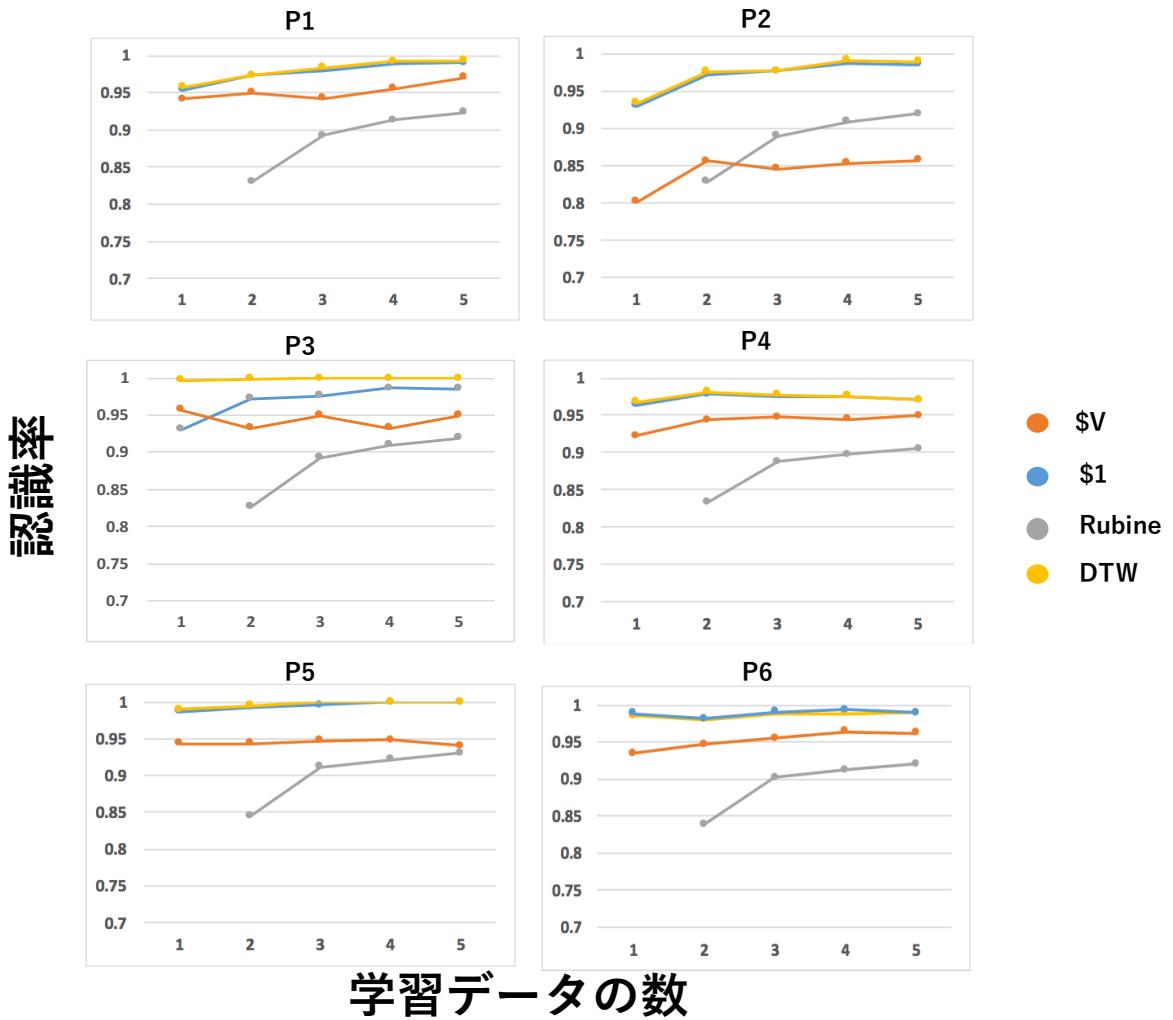


図 6.1: 各手法における、被験者ごとの認識率の平均 .

ら、ほとんどの被験者において、少ない学習データにおいて高い認識率を示し、学習データが1つの場合における、全ジェスチャセットの認識率の平均は91.2%(SD=0.07)、学習データが2つの場合は92.2%(0.04)、学習データが3つの場合は92.7%(0.05)、学習データが4つの場合は93.3%(0.04)、学習データが5つの場合は93.2%(0.05)となった。

6.2.2 認識速度

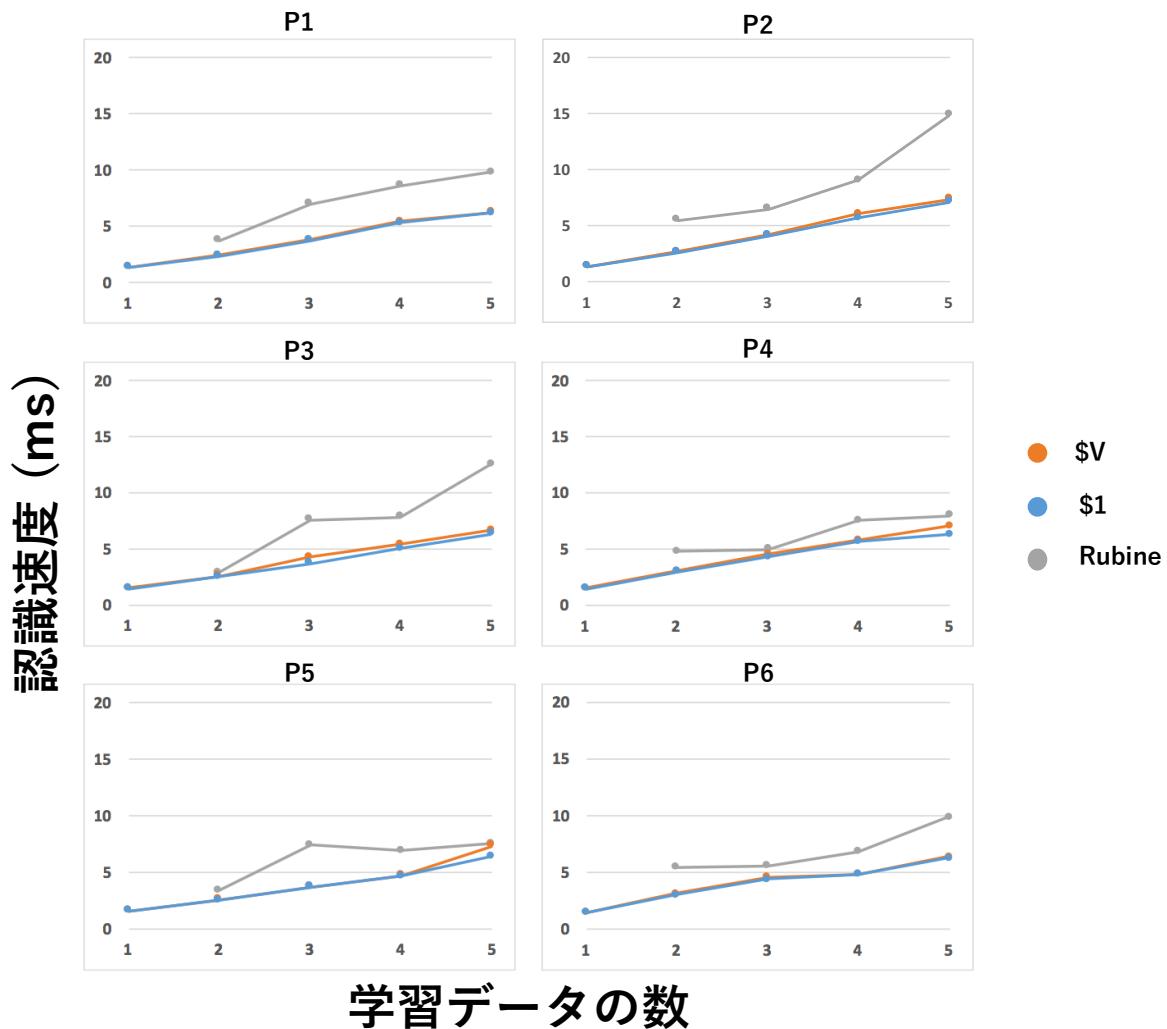


図 6.2: \$V , \$1 , Rubine における , 被験者ごとの認識速度の平均 .

図 6.2 に , \$V , \$1 , Rubine の , 図 6.3 に , DTW のジェスチャ 1 つを認識するまでの認識速度の結果を示す . \$V の認識速度は , 全被験者の平均が , 学習データの数が 1 つの場合 2.6ms (SD=0.2) , 学習データの数が 2 つの場合 3.5ms (0.3) , 学習データの数が 3 つの場合 4.1ms (0.3) , 学習データの数が 4 つの場合 4.9ms (0.3) , 学習データの数が 5 つの場合 6.1ms (0.4) となり非常に速いと言える . また , \$1 と認識速度に有意差はなく ($p < 0.001$) , \$V は \$1 と比べて認識速度の低下を抑えることに成功した . Rubine と比べると有意に速かった ($p < 0.005$) . DTW の認識速度は図 6.3 に示すように非常に遅く , \$V の認識速度は DTW のおよそ 100 分の 1 であった . また , 学習データに比例して認識速度が遅くなることも分かった .

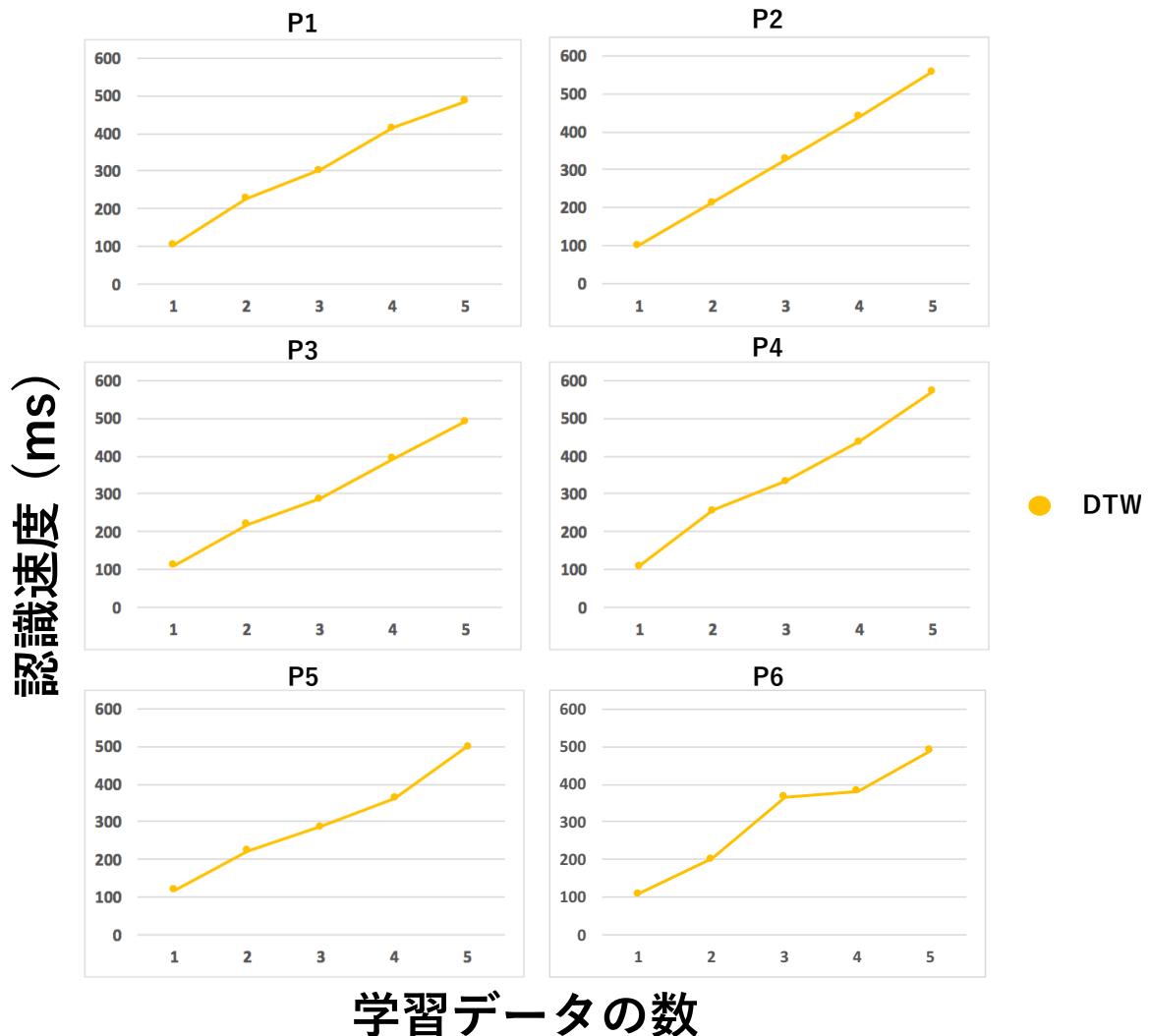


図 6.3: DTW における，被験者ごとの認識速度の平均．

6.2.3 識別性能の結果

\$V の識別性能の結果を , N-best List の 1 番目と 2 番目のスコアの差及びジェスチャが正しく認識された時の類似度及びジェスチャが正しく認識された時の類似度の最小値を示すことによって述べる .

図 6.4 に \$V における N-best List の 1 番目と 2 番目のスコアの差 , 図 6.5 に \$V におけるジェスチャが正しく認識された時の類似度 , 図 6.6 に \$V におけるジェスチャが正しく認識された時の類似度の最小値の結果を示す . N-best List の 1 番目と 2 番目のスコアの差について , 全ジェスチャセットの平均値はおよそ 0.24 (SD = 0.10) となった . また , ジェスチャが正しく認識された時の類似度の平均値は 0.94 (SD=0.04) となり非常に高いと言えるが , ジェスチャが正しく認識された時の類似度の最小値は被験者によってばらつきが生じ , 平均値はおよそ 0.83 (0.14) であった . また , N-best List の 1 番目と 2 番目のスコアの差は , 重み付けをしない場合と比べて有意に高く ($p < 0.01$) , 識別性能が向上したと言える . しかしながら , ジェスチャが正しく認識された時の類似度の最小値は , 重み付けをしない場合と比べて有意に低かった ($p < 0.01$) . これは , ジェスチャグループによっては , 適した重み付けがされていない場合があり , その場合類似度が低くなるからであると考えられる . 以上を踏まえ , 重み付けは多くのジェスチャグループにおいて適用可能であるが , 適用することによって類似度が低下する場合もあることがわかった . また , ジェスチャが一致した時の類似度の最小値の平均は 0.65 以上であり , N-best List の 1 番目と 2 番目のスコアの差の平均値は 0.15 以上であることから , 認識のための類似度の閾値を 0.5 とすることによって , ジェスチャが一致するか否かを判別することが可能となると言える .

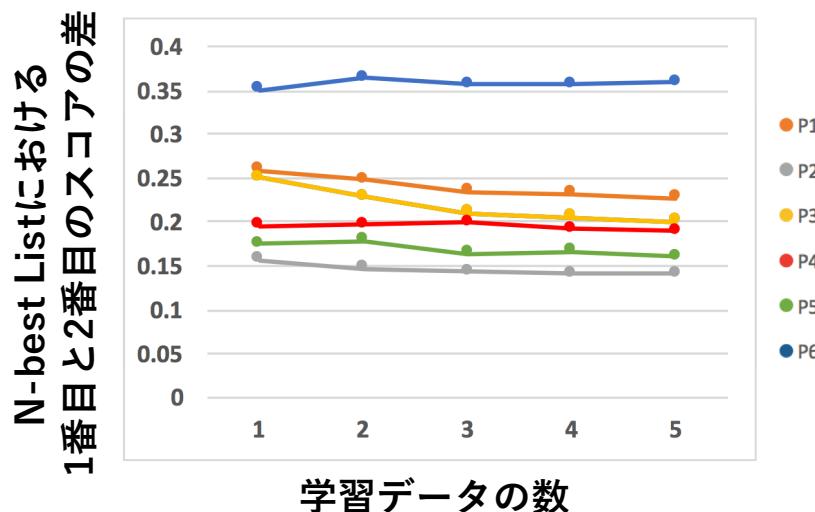


図 6.4: \$V における , N-best List の 1 番目と 2 番目のスコアの差の平均 .

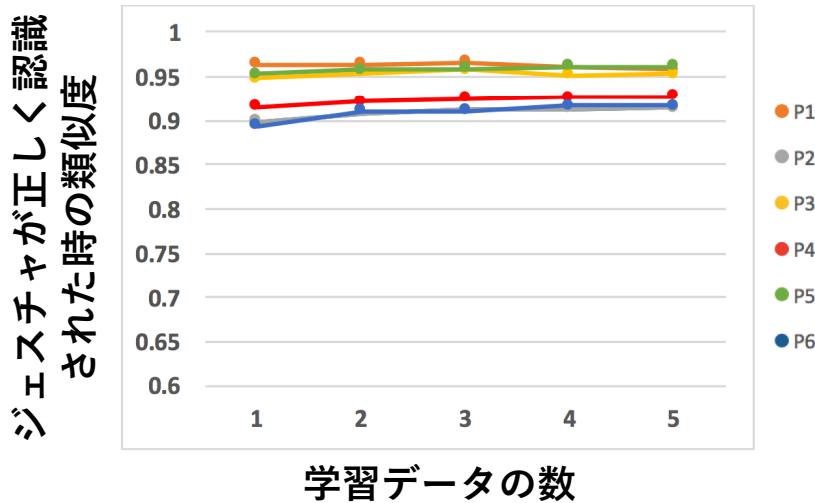


図 6.5: \$V における，ジェスチャが正しく認識された時の類似度の平均 .

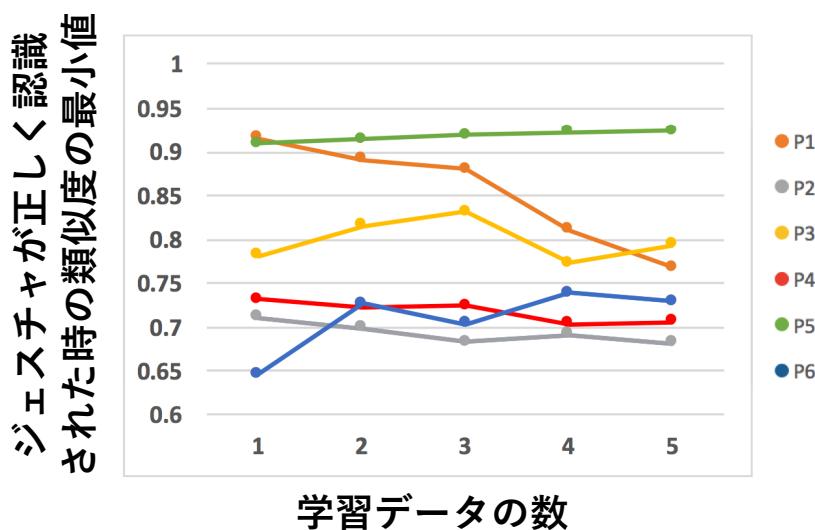


図 6.6: \$V における，ジェスチャが正しく認識された時の類似度の最小値の平均 .

以上を踏まえ，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及び，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は検証され，\$V\$は，認識率及び認識速度において高いパフォーマンスを示し，かつ，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なアルゴリズムであると言える．

第7章 アプリケーション例

\$Vを利用したアプリケーション例を本章にて示す。\$Vは少ない学習データにおいて、高い認識率及び識別性能を示すため、ユーザ定義手書きジェスチャを利用したアプリケーションを開発することができる。また、同じ形状及び書き順の手書きジェスチャを大きさ、向き、位置に関して識別可能であるため、それらを利用したアプリケーション例を示す。

7.1 手書きジェスチャを利用したアプリケーション開発ツールキット

手書きジェスチャを利用したアプリケーションを開発するためのツールキットを示す。

まず、ユーザは、入力として用いたい手書きジェスチャを実際に手書きジェスチャを入力する端末を用いて学習データとして1つずつ追加していく(図7.1a)。すると、\$Vが実装された本ツールキットは、追加された学習データを自動的にジェスチャグループに分類し、それぞれのジェスチャグループごとに、大きさ、向き、位置の特徴量に対する最適な重みを自動計算する(図7.1b)。最後に、ユーザは、利用したいアプリケーションにおける処理を実行するAppleScriptあるいはボタンと利用したい手書きジェスチャを対応付けることによって(図7.1c)、手書きジェスチャを利用したアプリケーションを開発することができる。

このツールキット使うことによって、例えば、図7.2のようなメディアプレイヤを開発することができる。このメディアプレイヤにおいて、再生、巻き戻し、前のメディア、次のメディアなどに対し、同じ書き順及び同じ形状の手書きジェスチャが複数割り当てられており(図7.2a)、それぞれ、大きさ、向き、位置の違いが利用されている。ユーザはスマートフォンを手書きジェスチャの入力端末として用いることによって、PC上のメディアプレイヤを操作することが可能である(図7.2b)。このようにして、学習データを1つ追加するのみによって、手書きジェスチャを入力として用いるアプリケーションを開発できる。また、大きさ、向き、位置の違いを利用することによって、より実際の動作に即した手書きジェスチャを割り当てることが可能となる。大きさ、向き、位置の違いを利用できない場合と比べて、利用可能な手書きジェスチャの種類が大きく拡大し、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を考案しやすくなったと言える。

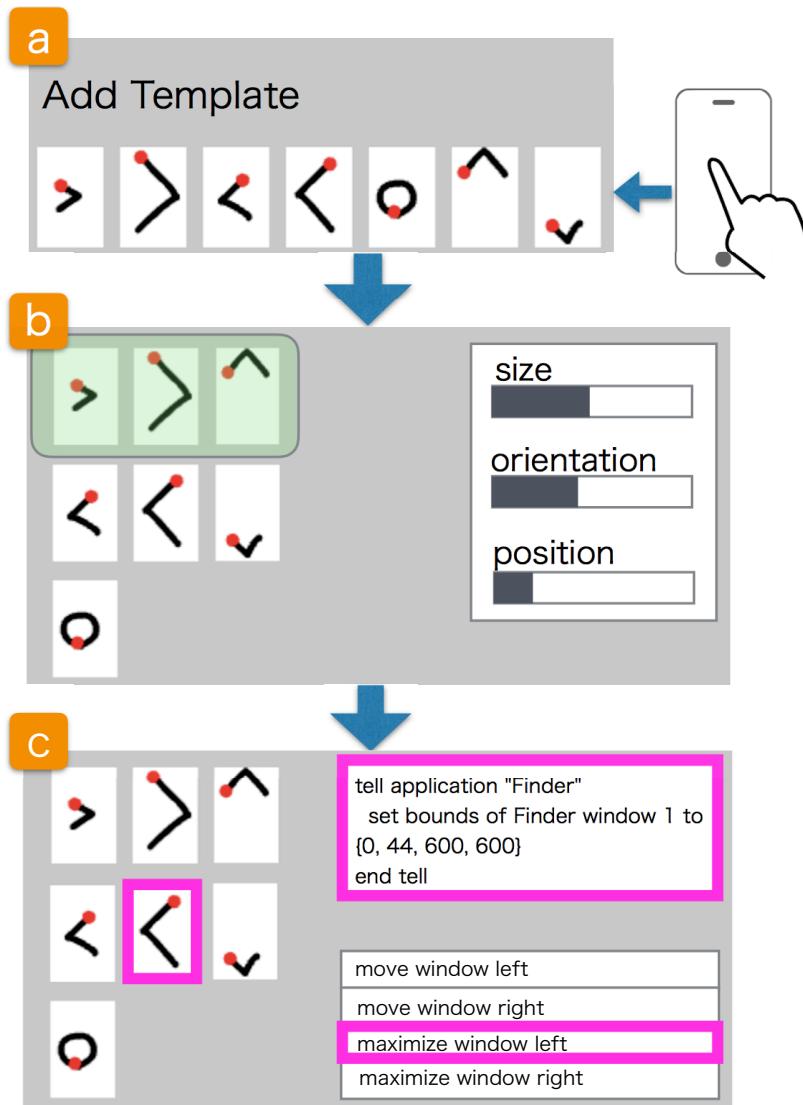


図 7.1: 手書きジェスチャを利用したアプリケーション開発ツールキット . (a) まず , ユーザはスマートフォンなどの手書きジェスチャを入力する端末を用いて学習データを追加することによって , (b) が実装されたツールキットが追加された学習データを自動的にジェスチャグループに分類し , ジェスチャグループごとに重みを計算する . (c) 最後にユーザは , アプリケーションにおける処理と手書きジェスチャを対応付ける .

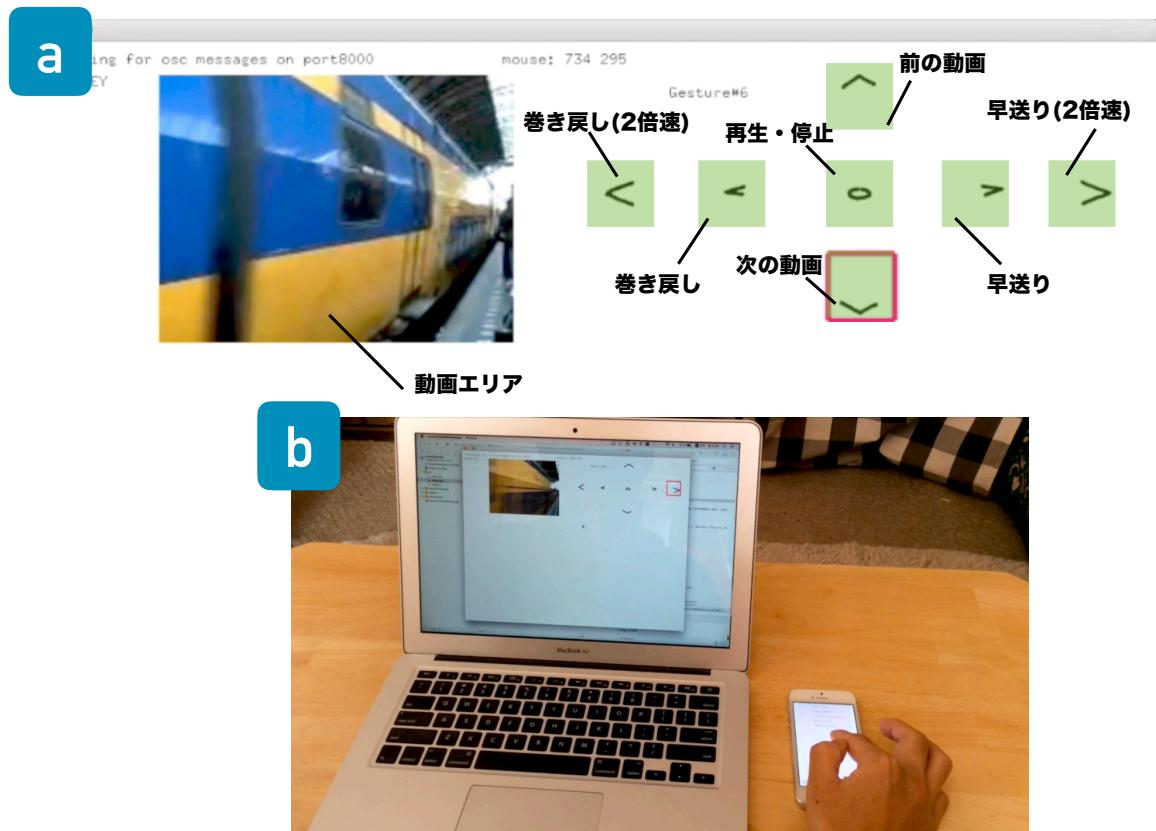


図 7.2: ツールキットを使って開発されたメディアプレイヤの例 . (a) アプリケーションのスクリーンショット , (b) 実際にスマートフォンを用いて入力している場面.

第8章 議論

7章までにおいて，\$Vアルゴリズムの詳細及び性能評価を述べた．本章において，\$Vが今後どのように改善されうるか，あるいは\$Vをどのように発展利用できるかについて議論する．

8.1 最適な重み付けの再定義

5.4節において，認識率及びN-best Listの1番目と2番目のスコアの差を向上させるための重みを実験的に求めた．しかしながら，ジェスチャグループ内の学習データ間の類似度と重みの関係を示す近似式の決定係数 R^2 は高いとは言えず，今回は，式5.9において示されるように変数を1つにし，段階的に変化させることによって近似式を求めた．このように決定係数 R^2 が低くなった要因は，ジェスチャグループ内の学習データ間の類似度と重みは，ある程度相関を示したものの，データにはばらつきが存在したことであると考えられる．

今回，このデータは，あるジェスチャグループ内の学習データ間の類似度において，ジェスチャが一致した時の類似度の平均値が0.9以上，N-best Listの1番目と2番目のスコアの差が0.2以上となる重みの平均値としてプロットされている．しかしながら，当然ながらこの条件を満たす多くの重みにおいて，具体的な類似度の平均値及びN-best Listの1番目と2番目のスコアの差には違いがある(例えば，類似度が0.9，N-best Listの1番目と2番目のスコアの差が0.2である重みも存在すれば，類似度が0.99，N-best Listの1番目と2番目のスコアの差が0.4である重みも存在する)．このことから，条件を満たす重みを平均するのではなく，高いスコアを示す重みをより重要視することによって，データのはばらつきを抑えられる可能性がある．データのはばらつきを抑えることによって，近似式の決定係数 R^2 は高くなり，よりデータを忠実に表す近似式を得られ，認識率の向上あるいは識別能力の向上を実現できるかもしれない．

8.2 ユーザに依存しない手書きジェスチャの精度評価

本研究において，ジェスチャを定義したアプリケーションユーザが，自身が定義したジェスチャを入力した時に，\$Vがどれだけ認識できるかを評価した．今回はアプリケーションユーザが手書きジェスチャ定義する場合を想定しているため，ユーザに依存した手書きジェスチャの評価を行った．しかしながら，アプリケーション開発者が手書きジェスチャを定義する場合，実際に入力するのはアプリケーションユーザであるため，ユーザに依存しない手書きジェスチャにおいても，高い認識率及び高い識別性能が求められる．\$Vは識別に必要のない特徴

量の重みを小さくするなどの処理を施すことによって、ロバスト性を極力維持しているため、ユーザに依存しない手書きジェスチャにおいて、重み付けをしない場合と比べて、高い認識率及び高い識別能力を示す可能性が高い。また、\$1, DTW, Rubineとの比較を含め、ユーザに依存しない手書きジェスチャの精度評価を行うことによって、さらなる改善の余地を発見できるであろう。

8.3 書き順に依存しないアルゴリズムの導入

$\$V$ は $\$1$ の拡張であり、 $\$1$ と同様に形状と書き順を同じジェスチャを認識することができるので、図8.1のような形状が同じであっても書き順の異なるジェスチャは別のジェスチャとして認識される。双方を同じジェスチャとして認識してほしい場合には、それぞれのジェスチャを同じ名前で登録する必要がある。このようなアプリケーションユーザへの負担を解消すべく、書き順に依存しない、つまり形状さえ同じであれば同じジェスチャとみなすアルゴリズムは $\$N$ [AW10]において開発されているため、このアルゴリズムを活用することによって書き順に依存しないアルゴリズムを実現することができる。

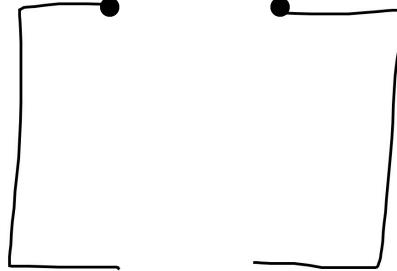


図 8.1: 形状が同じであるが書き順の異なる手書きジェスチャの例。

8.4 スマートフォン以外の端末への応用

$\$V$ は、スマートフォンへの手書きジェスチャの入力を想定した、手書きジェスチャ認識アルゴリズムである。そのため、スマートフォン以外の端末、例えばスマートウォッチやテーブルトップ端末などへの手書きジェスチャの認識アルゴリズムとしても応用できる。しかしながら、 $\$1$ とは違い、大きさ、向き、位置に関して識別可能であるため、それぞれに対する重みを定義する必要がある。例えば、スマートウォッチはスマートフォンと比べ入力領域が小さいため、大きさや位置の違いを利用してしづらく、大きさや位置への重みが小さくなる可能性がある。

高い，反対に，テーブルトップ端末はスマートフォンと比べ入力領域が大きいため，大きさや位置の違いを利用しやすく，大きさや位置への重みが大きくなる可能性が高い．

このように，入力領域を変数として重みを定義することによって，スマートフォン以外の端末への大きさ，向き，位置の違いを利用した手書きジェスチャを認識することが可能になる．

8.5 空中手書きジェスチャへの応用

\$V は，スマートフォンのような端末への入力を想定しており，認識できる手書きジェスチャは 2 次元のストロークからなる手書きジェスチャである．これを 3 次元に応用することによって，Kinect などの深度センサを用いた [TQXW13] あるいは，端末の加速度を用いた [RLL11] ユーザ定義の空中手書きジェスチャを認識することが可能になる．\$3 [KR10] は \$1 を 3 次元に応用し，空中手書きジェスチャを認識可能にした．このアルゴリズムを活用することによって実現することができる．

第9章 結論

本研究にて、ユーザ定義手書きジェスチャを認識するアルゴリズムである\$Vを開発した。本論文にてまず、ユーザ調査を行い、アプリケーションユーザは、手書きジェスチャの形状や書き順が同じでも、大きさ、向き、位置の違いを利用したジェスチャを入力したいという要望があることがわかった。これらユーザが定義する手書きジェスチャを認識するためのアルゴリズムを開発した。\$Vは\$1を拡張し、\$1のように、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行った特徴を持ちながら、\$1とは違い、大きさ、向き、位置に関して識別可能にした。その際、大きさ、向き、位置に関して不变な\$1と比べて、認識速度及び認識率の低下を抑えるために、ジェスチャグループを作成し、類似度計算するジェスチャの数を減らす、かつ保管されている学習データの類似度をもとに、識別するために必要な特徴量を選定するという手法を用いた。また、識別するために必要な特徴量を選定する上で、それらの特徴量が識別するためにどれくらい必要であるかを重みによって表現することによって、より尤度の高い認識を可能にした。

評価実験においては、93%の認識率を示し、N-best List の 1 番目と 2 番目のスコアの差は 0.24 であり、高い認識率、識別性能を示した。また、認識速度は\$1と有意差がなく高速であることを示した。また、大きさ、向き、位置が識別可能な既存アルゴリズムと比較し、認識率及び認識速度において高いパフォーマンスを示した。\$Vは\$1に簡単な数式からなるアルゴリズムを追加するだけであるため、ジェスチャ認識アルゴリズムのライブラリが提供されていない開発環境においても実装可能であるだけでなく、特に手書きジェスチャ認識への深い知識を持たないアプリケーション開発初学者にも、自身のシステムに容易に組み込むことが可能である。また、\$Vの手書きジェスチャ認識としての改善点及び応用可能性を示した。

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 Multi-stroke Recognizer. In *Proceedings of Graphics Interface 2012*, GI '12, pp. 117–120, Toronto, Canada, 2012. Canadian Information Processing Society.
- [AZ09] Caroline Appert and Shumin Zhai. Using Strokes As Command Shortcuts: Cognitive Benefits and Toolkit Support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 2289–2298, New York, NY, USA, 2009. ACM.
- [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.
- [DS94] David Skalak Department and David B. Skalak. Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 293–301. Morgan Kaufmann, 1994.

- [FVKS13] Andreas Fischer, Muriel Visani, Van Cuong Kieu, and Ching Y. Suen. Generation of Learning Samples for Historical Handwriting Recognition Using Image Degradation. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, HIP ’13, pp. 73–79, New York, NY, USA, 2013. ACM.
- [GFMdOg09] Javier Galbally, Julian Fierrez, Marcos Martinez-diaz, and Javier Ortega-garcia. Synthetic Generation of Handwritten Signatures Based on Spectral Analysis, 2009.
- [GKN⁺05] Basilis. Gatos, Thomas. Konidaris, K. Ntzios, Ioannis. Pratikakis, and Stavros. J. Perantonis. A Segmentation-free Approach for Keyword Search in Historical Type-written Documents. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR ’05, pp. 54–58, Washington, DC, USA, 2005. IEEE Computer Society.
- [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] James. Herold and Thomas. F. Stahovich. The 1cent 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 *Proceedings 15th International Conference on Pattern Recognition*, pp. 691–694, 2000.
- [HZS⁺07] Ken Hinckley, Shengdong Zhao, Raman Sarin, Patrick Baudisch, Edward Cutrell, Michael Shilman, and Desney Tan. Inkseine: In Situ Search for Active Note Taking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’07, pp. 251–260, New York, NY, USA, 2007. ACM.
- [KB93] Gordon Kurtenbach and William Buxton. The Limits of Expert Performance Using Hierarchic Marking Menus. In *Proceedings of the INTERCHI ’93 Conference on Human Factors in Computing Systems*, INTERCHI ’93, pp. 482–487, Amsterdam, The Netherlands, The Netherlands, 1993. IOS Press.

- [KR10] Sven Kratz and Michael Rohs. The \$3 Recognizer: Simple 3D Gesture Recognition on Mobile Devices. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pp. 419–420, New York, NY, USA, 2010. ACM.
- [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.
- [LGHH08] Chunyuan Liao, François Guimbretière, Ken Hinckley, and Jim Hollan. Papiercraft: A Gesture-based Command System for Interactive Paper. *ACM Transactions on Computer-Human Interaction*, Vol. 14, No. 4, pp. 18:1–18:27, January 2008.
- [Li10a] Yang Li. Gesture Search: A Tool for Fast Mobile Data Access. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pp. 87–96, New York, NY, USA, 2010. ACM.
- [Li10b] 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.
- [LKJ02] Emilie Lundin, Håkan Kvarnström, and Erland Jonsson. A Synthetic Fraud Data Generation Methodology. In *Proceedings of the 4th International Conference on Information and Communications Security*, ICICS '02, pp. 265–277, London, UK, 2002. Springer-Verlag.
- [LL11] Hao Lü and Yang Li. Gesture Avatar: A Technique for Operating Mobile User Interfaces Using Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 207–216, New York, NY, USA, 2011. 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.
- [MCvM97] Thomas P. Moran, Patrick Chiu, and William van Melle. Pen-based Interaction Techniques for Organizing Material on an Electronic Whiteboard. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, UIST '97, pp. 45–54, New York, NY, USA, 1997. ACM.
- [MMM⁺97] Brad A. Myers, Richard G. McDaniel, Robert C. Miller, Alan S. Ferency, 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.
- [NFC07] Ramanan Navaratnam, Andrew W. Fitzgibbon, and Roberto Cipolla. The Joint Manifold Model for Semi-supervised Multi-valued Regression. In *International Conference on Computer Vision*, pp. 1–8. IEEE Computer Society, 2007.
- [Per85] Ken Perlin. An Image Synthesizer. *SIGGRAPH Computer Graphics*, Vol. 19, No. 3, pp. 287–296, July 1985.
- [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.
- [PJ09] Hae-Sang Park and Chi-Hyuck Jun. A Simple and Fast Algorithm for K-medoids Clustering. *Expert System With Application*, Vol. 36, No. 2, pp. 3336–3341, March 2009.
- [PS00] Réjean Plamondon and Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 1, pp. 63–84, January 2000.
- [PTIL16] Corey Pittman, Eugene M. Taranta II, and Joseph J. LaViola, Jr. A \$-Family Friendly Approach to Prototype Selection. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, IUI '16, pp. 370–374, New York, NY, USA, 2016. ACM.
- [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.

- [RLL11] Jaime Ruiz, Yang Li, and Edward Lank. User-defined Motion Gestures for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’11, pp. 197–206, New York, NY, USA, 2011. ACM.
- [RSH11] J. Reaver, Thomas. F. Stahovich, and James. 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.
- [RSP12] Jose A. Rodriguez-Serrano and Florent Perronnin. Synthesizing Queries for Handwritten Word Image Retrieval. *Pattern Recognition*, Vol. 45, No. 9, pp. 3270–3276, September 2012.
- [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.
- [SFC⁺11] James. Shotton, Andrew. Fitzgibbon, Mat. Cook, Toby. Sharp, Mark. Finocchio, Richard. Moore, Alex. Kipman, and Andrew. Blake. Real-time Human Pose Recognition in Parts from Single Depth Images. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR ’11, pp. 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.
- [Sig06] A multi-level representation paradigm for handwriting stroke generation. *Human movement science* 25, 4, pp. 586–607, 2006.
- [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] Charles. C. Tappert. Cursive Script Recognition by Elastic Matching. *IBM Journal of Research 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, Canada, 2015. Canadian Information Processing Society.
- [TMPL16] Eugene M. Taranta, II, Mehran Maghoumi, Corey R. Pittman, and Joseph J. LaViola, Jr. A Rapid Prototyping Approach to Synthetic Data Generation for Improved 2D Gesture Recognition. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pp. 873–885, New York, NY, USA, 2016. ACM.
- [TQXW13] Jing Tian, Chengzhang Qu, Wenyuan Xu, and Song Wang. KinWrite: Handwriting-Based Authentication Using Kinect. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, 2013.
- [TSW90] Charles. C. Tappert, Ching. Y. Suen, and Toru. Wakahara. The State of the Art in Online Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, pp. 787–808, August 1990.
- [Vat12a] Radu-Daniel Vatavu. 1F: One Accessory Feature Design for Gesture Recognizers. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, pp. 297–300, New York, NY, USA, 2012. ACM.
- [Vat12b] 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.
- [ZBLF08] Robert C. Zeleznik, Andrew Bragdon, Chu-Chi Liu, and Andrew Forsberg. Lineogrammer: Creating Diagrams by Drawing. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pp. 161–170, New York, NY, USA, 2008. 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.
- [増井 97] 増井俊之. 予測/例示インターフェースの研究動向 (特集インタラクティブソフトウェア). コンピュータソフトウェア, Vol. 14, No. 3, pp. 220–235, may 1997.

付録A \$Vアルゴリズムの擬似コード

\$Vアルゴリズムにおいて，学習データを追加する際のアルゴリズムと，入力データを認識する際のアルゴリズムの擬似コード示す．本擬似コードにおいて，templates は同じ名前の手書きジェスチャの学習データ，group は1つのジェスチャグループ，groups は複数のジェスチャグループ(すなわち全学習データ)を示している．

\$V-ADD-TEMPLATE(template)

```
1: n  $\leftarrow$  64
2: templatenewPoints  $\leftarrow$  REAMPLE(templatepoints, n) ▷ Refer to $1
3: tempatesize  $\leftarrow$  SIZE(templatenewPoints)
4: tempateorientation  $\leftarrow$  ORIENTATION(templatenewPoints)
5: tempateposition  $\leftarrow$  POSITION(templatenewPoints)
6: < templates, group >  $\leftarrow$  SEARCH-SAME-NAME-TEMPLATE(template)
7: if templates then
8:   templates.push(template)
9:   SET-TEMPLATE-FEATURE(templates)
10:  SET-GESTURE-GROUP-SIMILARITY(templates, group)
11:  SET-GESTURE-GROUP-WEIGHT(group)
12: else if
13:   then < group, score >  $\leftarrow$  $1-RECOGNIZER(template, groups)
14:   if score  $\geq$  0.8 then
15:     templates  $\leftarrow$  MAKE-TEMPLATES(template)
16:     group.push(templates)
17:     SET-GESTURE-GROUP-SIMILARITY(templates, group)
18:     SET-GESTURE-GROUP-WEIGHT(group)
19:   else if
20:     then templates  $\leftarrow$  MAKE-TEMPLATES(template)
21:     group  $\leftarrow$  MAKE-GESTURE-GROUP(templates)
22:     groups.push(group)
23:   end if
24: end if
```

\$V-RECOGNIZER(gesture, groups)

```
1:  $n \leftarrow 64$ 
2:  $gesture_{newPoints} \leftarrow REAMPLE(template_{points}, n)$  ▷ Refer to $1
3:  $gesture_{size} \leftarrow SIZE(template_{newPoints})$ 
4:  $gesture_{orientation} \leftarrow ORIENTATION(template_{newPoints})$ 
5:  $gesture_{position} \leftarrow POSITION(template_{newPoints})$ 
6:  $<group, score> \leftarrow \$1-RECOGNIZER(template, groups)$ 
7:  $bestScore \leftarrow 0$ 
8: for  $i \leftarrow 1, groupLength$  do
9:    $templates \leftarrow group_i$ 
10:   $sizeSimilarity \leftarrow SIZE-SIMILARITY(templates_{size}, gesture_{size})$ 
11:   $\theta_1 \leftarrow templates_{orientation}$ 
12:   $\theta_2 \leftarrow gesture_{orientation}$ 
13:   $orientationSimilarity \leftarrow ORIENTATION-SIMILARITY(\theta_1, \theta_2)$ 
14:   $x \leftarrow templates_{positionx}$ 
15:   $x' \leftarrow gesture_{positionx}$ 
16:   $y \leftarrow templates_{positiony}$ 
17:   $y' \leftarrow gesture_{positiony}$ 
18:   $positionSimilarity \leftarrow POSITION-SIMILARITY(x, x', y, y')$ 
19:   $score \leftarrow sizeSimilarity \times group_{sizeWeight} + orientationSimilarity \times$ 
      $group_{orientationWeight} + positionSimilarity \times group_{positionWeight}$ 
20:  if  $score > bestScore$  then
21:     $bestScore \leftarrow score$ 
22:     $T \leftarrow templates$ 
23:  end if
24: end for
25: return  $< T, bestScore >$ 
```

SET-TEMPLATE-FEATURE(templates)

```
1: for  $i \leftarrow 1, \text{templates.length}$  do
2:    $\text{templates.size} \leftarrow \text{templates.size} + \text{templates.i.size}$ 
3:    $\text{templates.orientation} \leftarrow \text{templates.orientation} + \text{templates.i.orientation}$ 
4:    $\text{templates.positionx} \leftarrow \text{templates.positionx} + \text{templates.i.positionx}$ 
5:    $\text{templates.positiony} \leftarrow \text{templates.positiony} + \text{templates.i.positiony}$ 
6: end for
7:  $\text{templates.size} \leftarrow \text{templates.size}/\text{templates.length}$ 
8:  $\text{templates.orientation} \leftarrow \text{templates.orientation}/\text{templates.length}$ 
9:  $\text{templates.positionx} \leftarrow \text{templates.positionx}/\text{templates.length}$ 
10:  $\text{templates.positiony} \leftarrow \text{templates.positiony}/\text{templates.length}$ 
```

SET-GESTURE-GROUP-SIMILARITY(templates, group)

```
1:  $\text{minSizeSimilarity} \leftarrow 1$ 
2:  $\text{minOrientationSimilarity} \leftarrow 1$ 
3:  $\text{minPositionSimilarity} \leftarrow 1$ 
4: for  $i \leftarrow 1, \text{group.length}$  do
5:    $\text{sizeSimilarity} \leftarrow \text{SIZE-SIMILARITY}(\text{templates.size}, \text{group.i.size})$ 
6:    $\text{minSizeSimilarity} \leftarrow \text{MIN}(\text{minSizeSimilarity}, \text{sizeSimilarity})$ 
7:    $\theta_1 \leftarrow \text{templates.orientation}$ 
8:    $\theta_2 \leftarrow \text{group.i.orientation}$ 
9:    $\text{orientationSimilarity} \leftarrow \text{ORIENTATION-SIMILARITY}(\theta_1, \theta_2)$ 
10:   $\text{minOrientationSimilarity} \leftarrow \text{MIN}(\text{minOrientationSimilarity}, \text{orientationSimilarity})$ 
11:   $x \leftarrow \text{templates.positionx}$ 
12:   $x' \leftarrow \text{group.i.positionx}$ 
13:   $y \leftarrow \text{templates.positiony}$ 
14:   $y' \leftarrow \text{group.i.positiony}$ 
15:   $\text{positionSimilarity} \leftarrow \text{POSITION-SIMILARITY}(x, x', y, y')$ 
16:   $\text{minPositionSimilarity} \leftarrow \text{MIN}(\text{minPositionSimilarity}, \text{positionSimilarity})$ 
17: end for
18:  $\text{group.sizeSimilarity} \leftarrow \text{minSizeSimilarity}$ 
19:  $\text{group.orientationSimilarity} \leftarrow \text{minOrientationSimilarity}$ 
20:  $\text{group.positionSimilarity} \leftarrow \text{minPositionSimilarity}$ 
```

SIZE-SIMILARITY(size1, size2)

```
1: if  $size1 \geq size2$  then
2:    $sizeSimilarity \leftarrow size2 / size1$ 
3: else if
4:   then  $sizeSimilarity \leftarrow size1 / size2$ 
5: end if
6: return  $sizeSimilarity$ 
```

ORIENTATION-SIMILARITY(orientation1, orientation2)

```
1:  $orientationSimilarity \leftarrow ABS(templateOrientation - groupOrientation)$ 
2: if  $orientationSimilarity > \pi$  then
3:    $orientationSimilarity \leftarrow 2\pi - orientationSimilarity$ 
4: end if
5:  $orientationSimilarity \leftarrow 1 - orientationSimilarity / PI$ 
6: return  $orientationSimilarity$ 
```

POSITION-SIMILARITY(x, x', y, y')

```
1:  $width \leftarrow inputScreenWidth$ 
2:  $height \leftarrow inputScreenHeight$ 
3:  $positionSimilarity \leftarrow 1 - (\sqrt{(x - x')^2 + (y - y')^2} / \sqrt{width^2 + height^2})$ 
4: return  $positionSimilarity$ 
```

SET-GESTURE-GROUP-WEIGHT(group)

```
1:  $Ss \leftarrow groupSizeSimilarity$ 
2:  $So \leftarrow groupOrientationSimilarity$ 
3:  $Sp \leftarrow groupPositionSimilarity$ 
4:  $groupSizeWeight \leftarrow (5.5 + 3.5 \times (So + Sp)) / Ss$ 
5:  $groupOrientationWeight \leftarrow (5.0 + 3.0 \times (Ss + Sp)) / Sp$ 
6:  $groupPositionWeight \leftarrow (3.0 + 2.0 \times (Ss + So)) / Sp$ 
7:  $sumWeight \leftarrow groupSizeWeight + groupOrientationWeight + groupPositionWeight$ 
8:  $groupSizeWeight \leftarrow groupSizeWeight / sumWeight$ 
9:  $groupOrientationWeight \leftarrow groupOrientationWeight / sumWeight$ 
10:  $groupPositionWeight \leftarrow groupPositionWeight / sumWeight$ 
```

MAKE-TEMPLATES(template)

```
1: templates  $\leftarrow$  new TEMPLATES
2: templates.push(template)
3: templates.size  $\leftarrow -\text{templates.size}$ 
4: templates.orientation  $\leftarrow -\text{templates.orientation}$ 
5: templates.position  $\leftarrow -\text{templates.position}$ 
6: return templates
```

MAKE-GESTURE-GROUP(templates)

```
1: group  $\leftarrow$  new GROUP
2: group.push(templates)
3: group.sizeSimilarity  $\leftarrow -1$ 
4: group.orientationSimilarity  $\leftarrow -1$ 
5: group.positionSimilarity  $\leftarrow -1$ 
6: group.sizeWeight  $\leftarrow 0$ 
7: group.orientationWeight  $\leftarrow 0$ 
8: group.positionWeight  $\leftarrow 0$ 
9: return group
```

SIZE(points)

```
1: B  $\leftarrow$  BOUNDING-BOX(points)
2: return Bwidth  $\times$  Bheight
```

ORIENTATION(points)

```
1: c  $\leftarrow$  CENTROID(points)
2: return ATAN(cy - points0y, cx - points0x)
```

POSITION(points)

```
1: return CENTROID(points)
```

\$1-RECOGNIZER(gesture, groups)

```
1: b  $\leftarrow$  0
2: size  $\leftarrow$  250
3: NORMALIZE(gesture)
4: for i  $\leftarrow$  1, groups.length do
5:   group  $\leftarrow$  groupsi
6:   for j  $\leftarrow$  1, group.length do
7:     templates  $\leftarrow$  groupj
8:     for k  $\leftarrow$  1, templates.length do
9:       NORMALIZE(templatesk)
10:      d  $\leftarrow$  DISTNCE-AT-BEST-ANGLE(gesture, templatesk)     $\triangleright$  Refer to $1
11:      if d  $\downarrow$  b then
12:        b  $\leftarrow$  d
13:        bestGroup  $\leftarrow$  group
14:      end if
15:    end for
16:  end for
17: end for
18: score  $\leftarrow$   $1 - b / 0.5\sqrt{size^2 + size^2}$ 
19: return < bestGroup, score >
```

NORMALIZE(template)

```
1: ROTATE-TO-ZERO(templatepoints)                                 $\triangleright$  Refer to $1
2: SCALE-TO-SQUARE(templatepoints)                                 $\triangleright$  Refer to $1
3: TRANSLATE-TO-ORIGIN(templatepoints)                             $\triangleright$  Refer to $1
```

付録B ユーザ調査により得られた手書きジェスチャ

...
.のとき同じ

・ファイル
.別のアクリル板

・メモ
.記入内容クリア

・ラウザ
.前のページへ
戻る

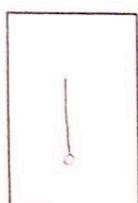
・スクロール
.新しいタブを開く

・音楽再生
.音量上げる

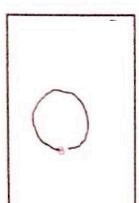
・音楽
.音量下げる



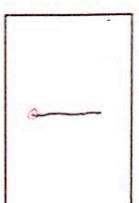
音楽
.次の曲へ



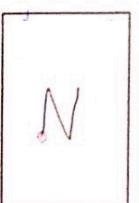
音楽
.前の曲へ



音楽
.早送り



音楽
.巻戻し



・スクロール
.新しいタブ



・カメラ
.拡大



・カメラ
.縮小



電子書籍
.しおり



音楽
.ミート



音楽
.一時停止



電子書籍
.最初のページ



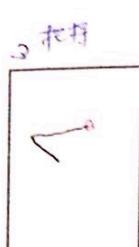
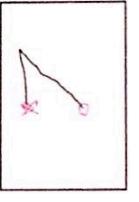
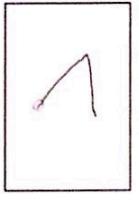
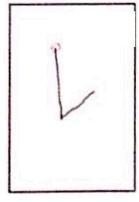
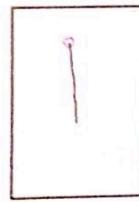
電子書籍
.最後のページ



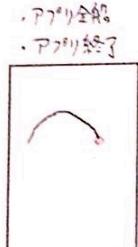
北極



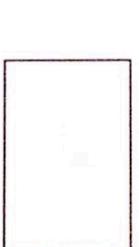
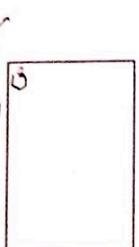
地獄



右折



アクリル
.アクリル終了



・ブーラクリ

・開く(アガ)

開(ハタフ)

ブックマーク
を開く

ブックマーク
に入れると

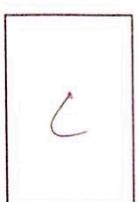
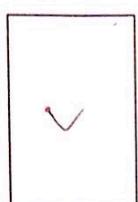
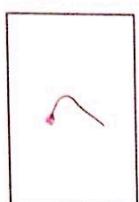
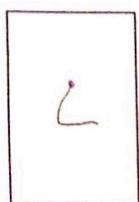
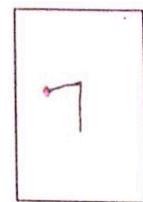
LINE

写真録取

LINE

写真動画録
音楽削除

music player



・アラート

・アラート

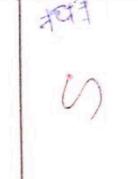
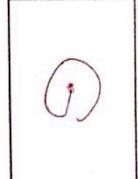
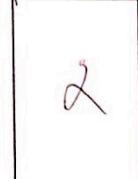
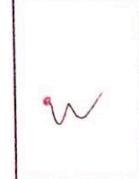
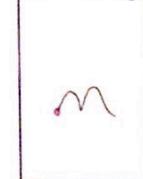
全体的
スクリーンショット

全体的
すべてのアラートを削除・重複なし

全体的
特定のアラートを削除・リストの削除

全体的
特定のアラートを削除

全体的
特定のアラートを削除



PC側
すべてのタスクを削除する

PC側
すべてのタスクを削除する

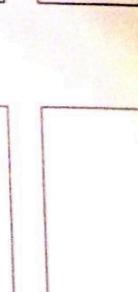
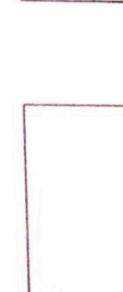
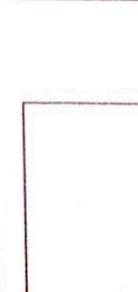
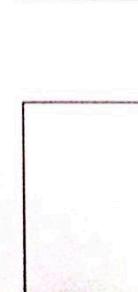
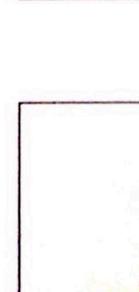
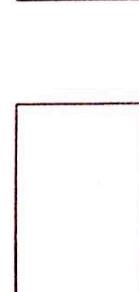
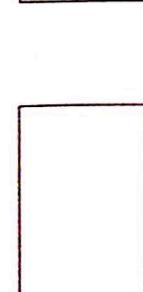
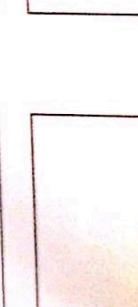
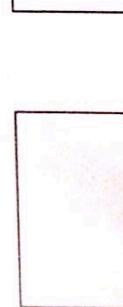
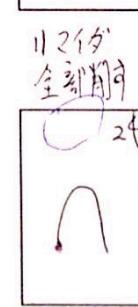
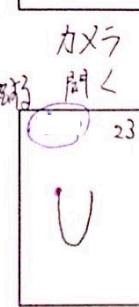
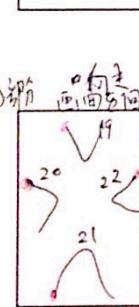
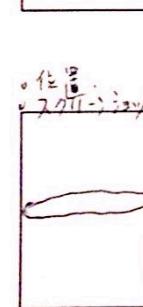
PC側
オシャッターボタン

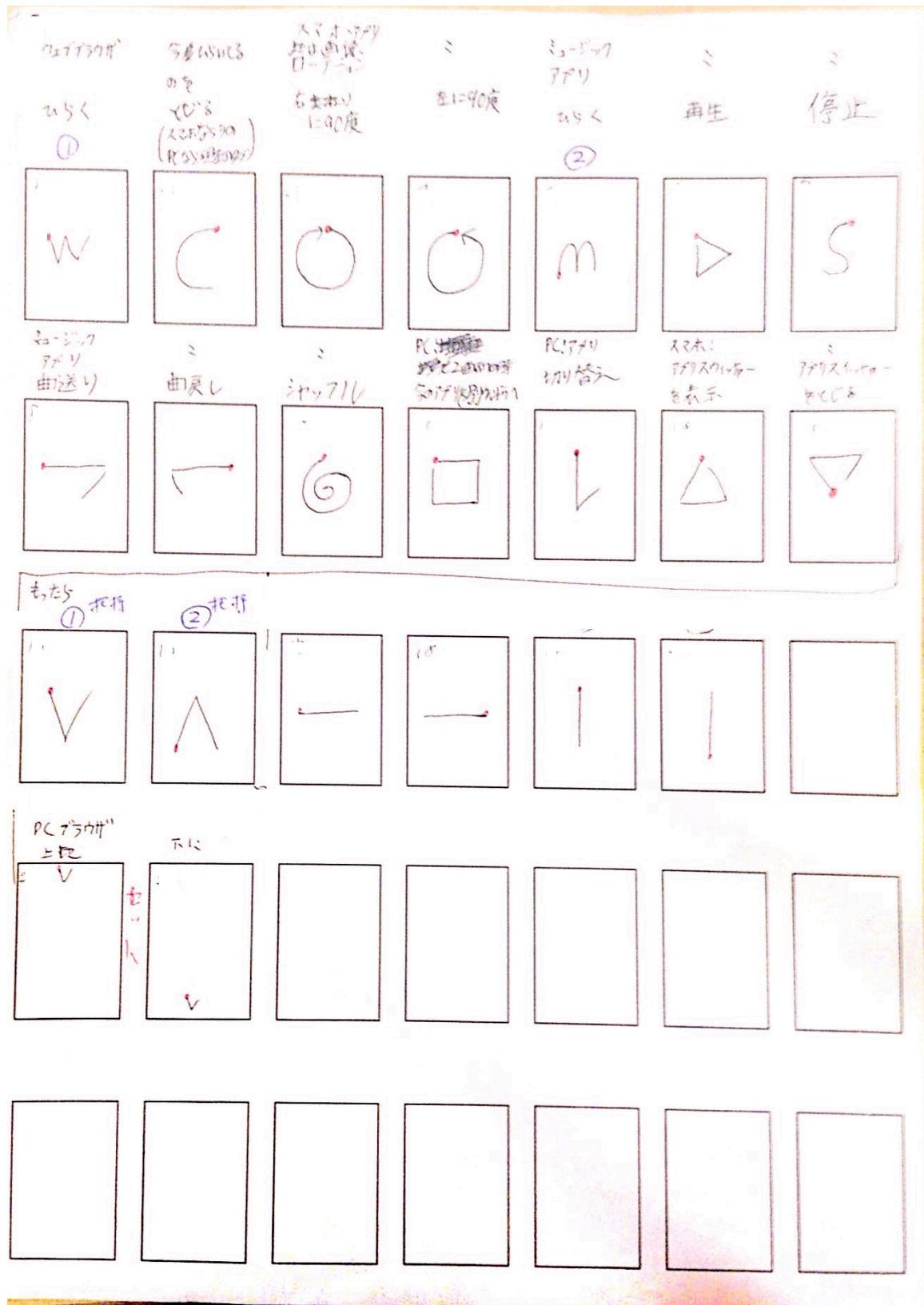
PC側
スリーブ

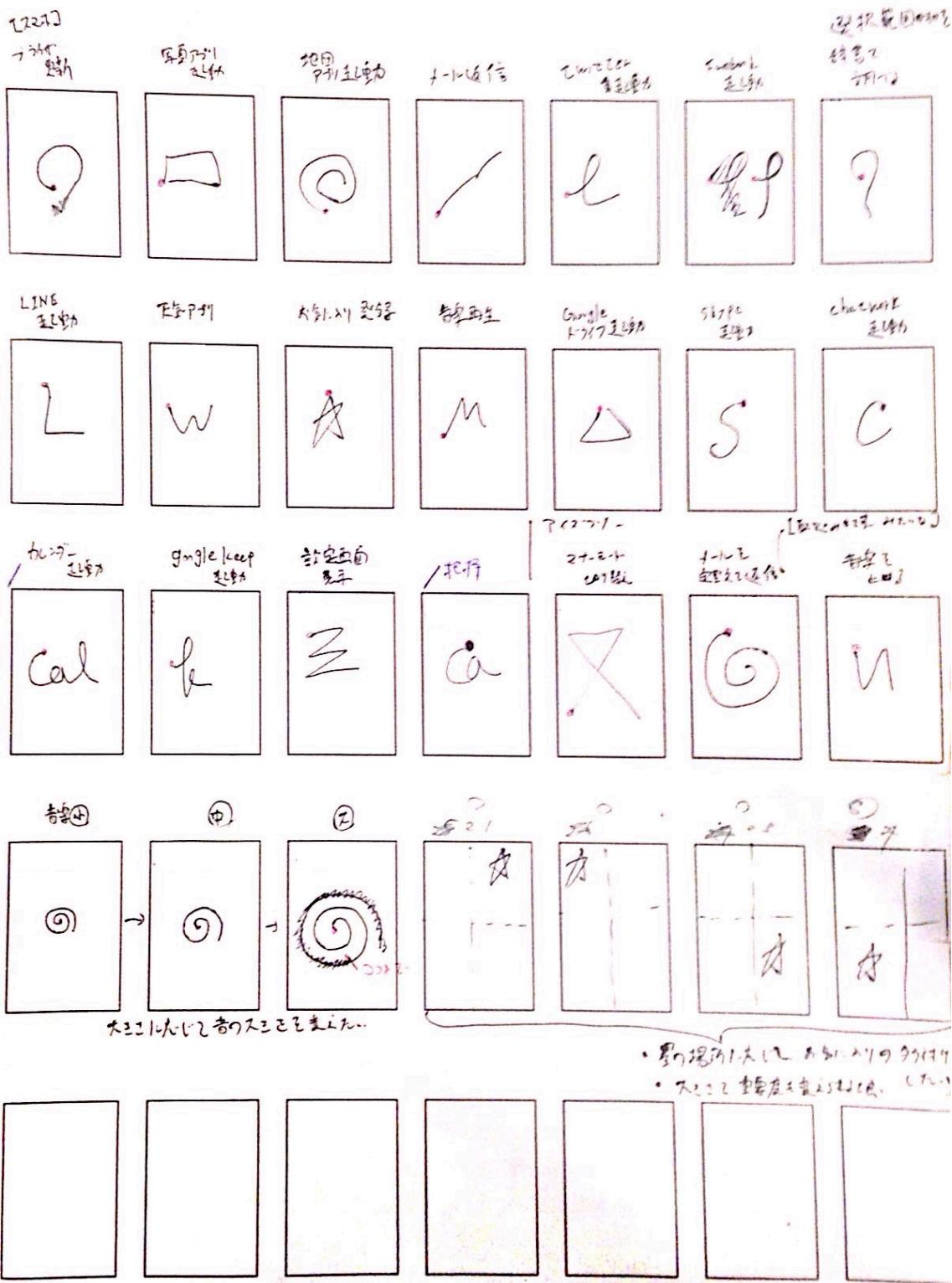
S

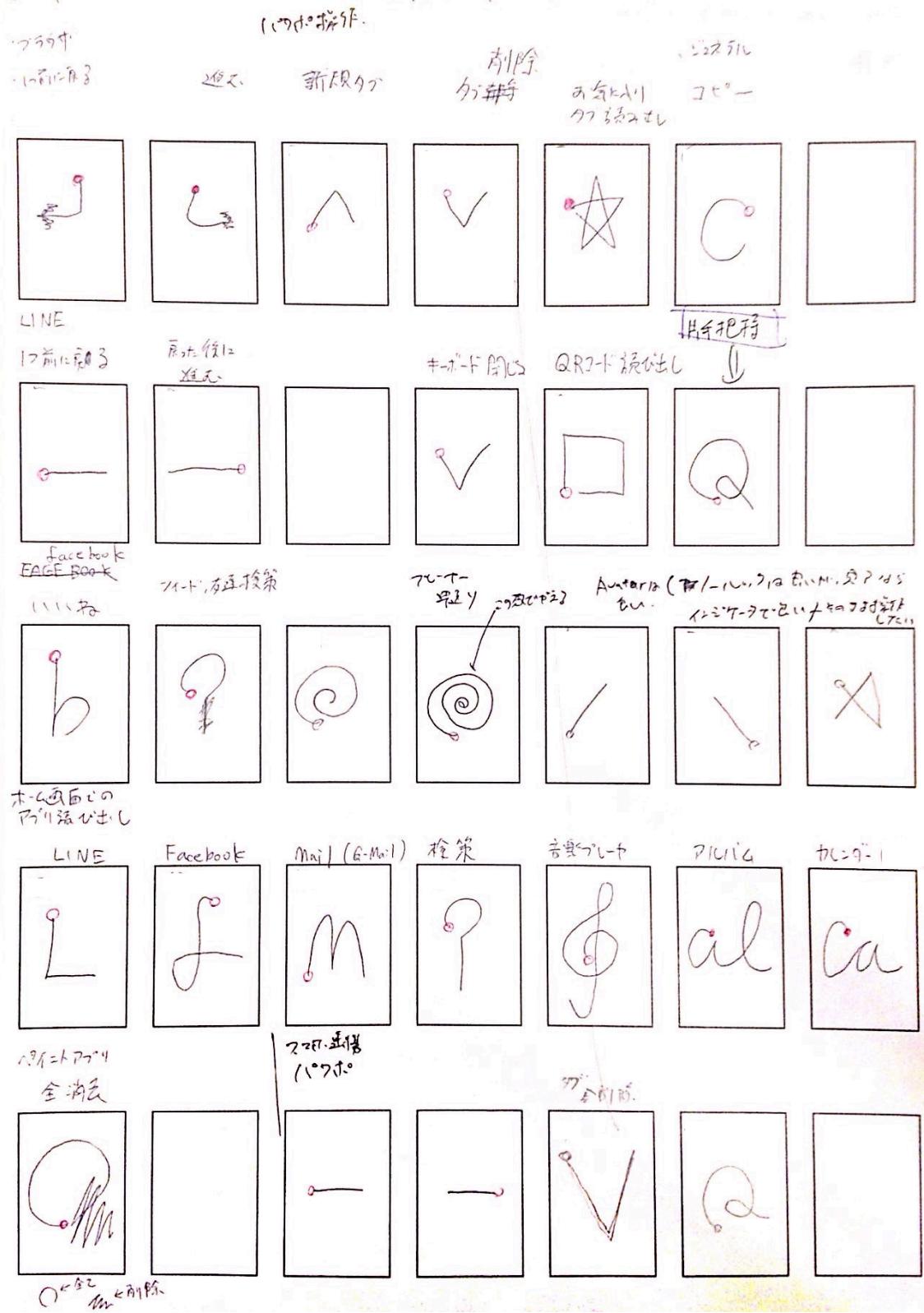
マウント

マウント



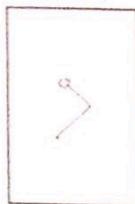




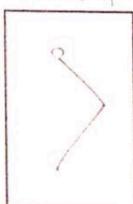


音量上げ

早送り



次の音楽



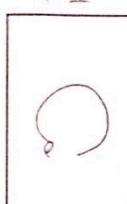
巻き戻し



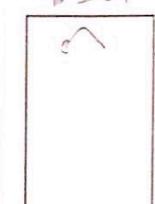
前略奪



再生



音量UP



音量down



グラフ表示

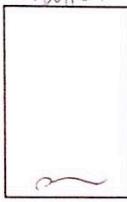
TOP



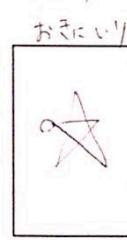
途中



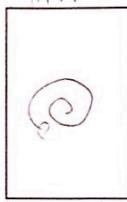
BOTTOM



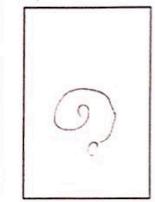
グラフ



前略奪



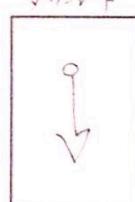
再生



音量-f



音量-f



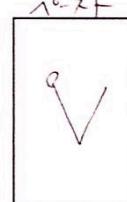
テキスト入力



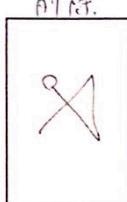
TOP



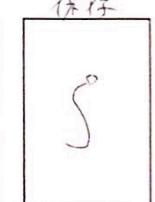
入力



削除



保存



音量-f



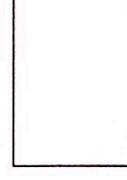
テキスト入力



TOP



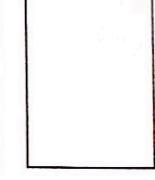
入力



削除



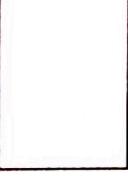
保存



音量-f



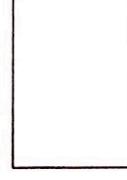
テキスト入力



TOP



入力



削除



保存

