1　アプリケーションの概要

LeapMotion という人の手と指の動きを感知し、各関節の座標を取得できる新たな入力デバイスを用いて、手のジェスチャーによる生体認証アプリケーションを作成した。手を用いた生体認証の実例として、指紋や静脈といった複雑な模様を本人しか持ちえない固有の情報とし認証を行うシステムがある。しかしその静的性故に模様が正確に他者に知られると偽装される可能性が少なからずある。また、手は人間が非常に精密かつ自由に動かすことのできる部位であるという身体的特徴を持つ。そこで手の持つ複雑な静的情報認証システムに動的な認証システムを加えことが出来るとセキュリティは増大するはずである。以上の考えから、手の動的な認証方法の提案と勉強のため、LeapMotion を用いた簡易動的生体認証システムを作成した。

2　機能と実装方法

初めに、LeapMotion から取得できる関節の座標データを以下に示す。

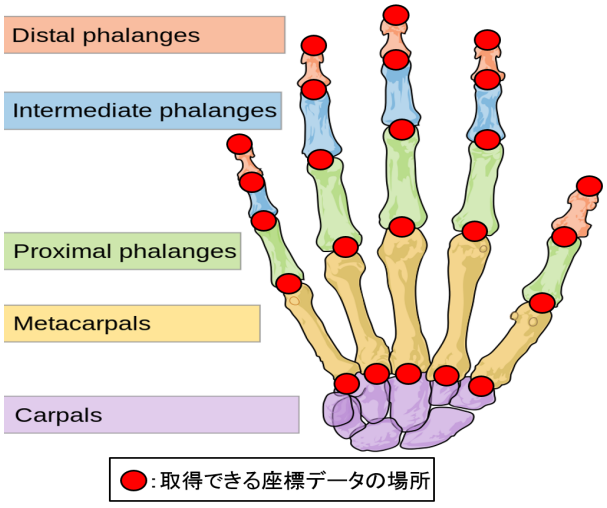

図 1: LeapMotion から取得できる関節の座標データ (参照文献 [6])

2.1　ユーザーの登録

認証鍵を手の静的、動的それぞれのデータから作成する。静的キーには指の長さと太さをデータとして登録した。動的キーは指を動かしているか停止中かを判別し、動作中から停止中に遷移したタイミングで次々に各骨の向きを保存して数フレームの連続的なキーデータ作成した。実装方法としては、静的キーは LeapMotion から値を参照するだけである。動的キーのための動作中か停止中の判定は、指定した時間間隔ごとに現在と一時刻前の全ての骨の向きの内積をとり総和を求める。その値が経験的に設けた閾値を超えれば動作中、越えなければ停止中と判定するよう機能を実装した。骨の座標で無く向きの変化で判定を行っているのは手ぶれなどで誤作動するのを防ぐためである。

2.2　ユーザー認証

保存された動的キーをロードし、本人が登録した動きを LeapMotion 上で再現することでユーザ認証機能を作成した。実装方法は各指ごとで骨の向きの内積の平均をとり一致率とする。全ての指で一致率が閾値を超えればそのフレームを突破できる。こうして保存したフレームを全て突破するとキーが開く。手を適当に動かしてキーを突破しようとするのを防ぐため制限時間を設定し、認証開始から一定時間内にキーを解ききらないと認証失敗とした。

3　アプリの使用方法

以下にアプリの画面と状態の遷移図を示す。

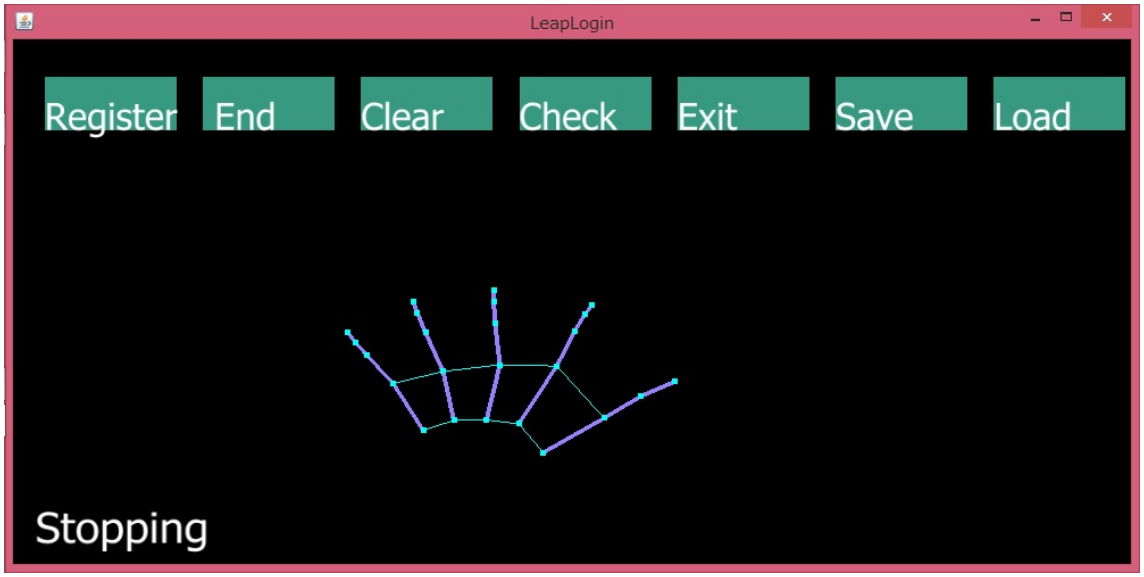

図 2: メニュー画面の様子。ボタンと LeapMotion かざした手が描写されており、実際の手を動かして画面の手を操作する。ボタンは人差し指の先端をユーザの正面にあるパソコンのモニタ側に押し込むと押すことが出来る。

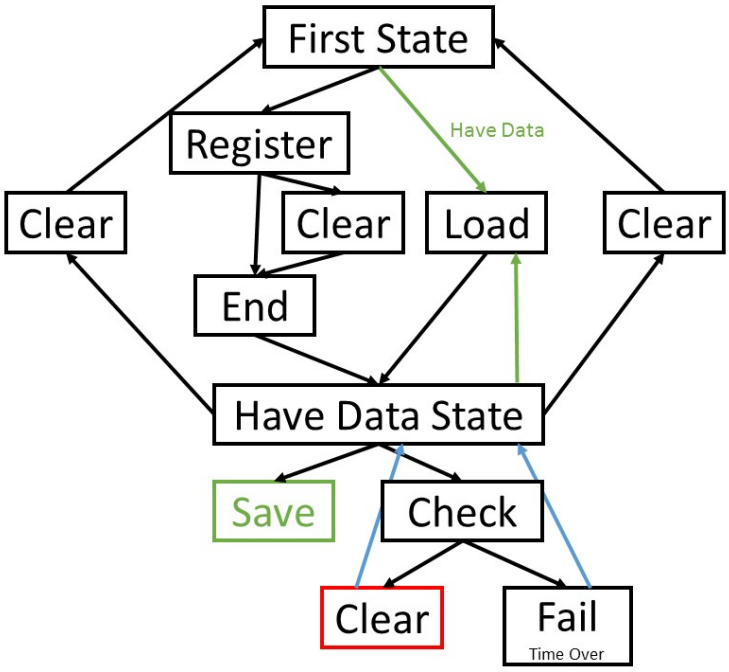

図 3: 状態図

以下ではアプリケーションで認証対象となるキーを一時キーとする。使用方法は大きく次の 5 項に分けられる。

3.1 一時キーの登録 (Start ボタン)

一時キーを新たに保存開始する。ユーザの登録の項で述べた通りにフレーム群の保存が開始され、適当なフレームが確保されると End ボタンを押して登録を終了する。この時 Clear キーを押すと初めから登録しなおすことが出来る。また Start ボタンを押した瞬間や End ボタンを押す前などで指を動かすとフレームが保存されてしまう可能性があるので、Start ボタンを押した後、End ボタンを押す前の最終フレームを保存し終えた後は手を制止させておく必要がある。

3.2 キーの保存 (Save ボタン)

一時キーに保存されたキーデータをテキストファイルに書き出す。また今回はユーザー数は一人の時に限定したので、データベースはキー一つ分である。

3.3 データロード (Load ボタン)

ボタンが押されたときに一時キー初期化し、テキストファイルからデータを読み込み一時データにセットする。この時はセキュリティのため保存されたフレームの赤い描写は行わない。

3.4 認証開始 (Check ボタン)

一時キーがセットされた状態でボタンが押されるとユーザ認証の項で述べた通りに認証が開始し、制限時間以内にキーを解ききると Clear が表示される。また認証中に手を抜くと強制的に認証失敗となる。これはこの認証システムが動的キーの作成を目標としているからである。

3.5 クリアー (Clear ボタン)

今保有している一時キー全てリセットする。