カメラと両手のみを必要とする新たな入力システム

松本琉大桜

2024 年 2 月 4 日

1　背景

近年、拡張現実 (Augmented Reality) および仮想現実 (Virtual Reality) 技術の普及により、HMD(Head-Mounted Display) がますます一般化してきている。さらに新型コロナウイルスの影響により、非接触タッチディスプレイへの需要が急速に拡大している。

はじめに、HMD における文字入力に関しては、それぞれの製品が独自のアプローチを取っている。例えば、Meta Quest 3[*1]では手で持つデバイスを使用し、ユーザーはポインタを指すことで文字入力を行う。一方、Vision Pro[*2]では音声認識によって文字入力を行う。しかしこれらは現在最も一般的な入力方法である、キーボードによる入力方法とはかけ離れたものである。

また、言語によってキーボード入力の方法の違いがある。英語の場合、キーボードのキーの種類は文字だけでも 26 種類である。一方で日本語のフリック入力の場合、キーは 10 種類しかなく、そこからさらに 4 方向へと分岐することで 50 種類の入力を可能にする。

次に、非接触タッチディスプレイに関して述べる。非接触タッチディスプレイは、三菱電機が開発したもの[*3]や、DNP が開発したもの[*4]などがあるが、大掛かりなデバイスを必要としている。

2　実装したツール

実装したツールは、カメラを使用して手の動きを検出し、左右どちらかの手を仮想のキーボード、逆の手を入力デバイスとして利用することができるツールである。図 1 に使用例を示す。今回は、実験的に 0 から 9 の数字と Enter、Delete を入力できるようにした。

2.1　外部仕様

ユーザは以下の手順で入力を行う。

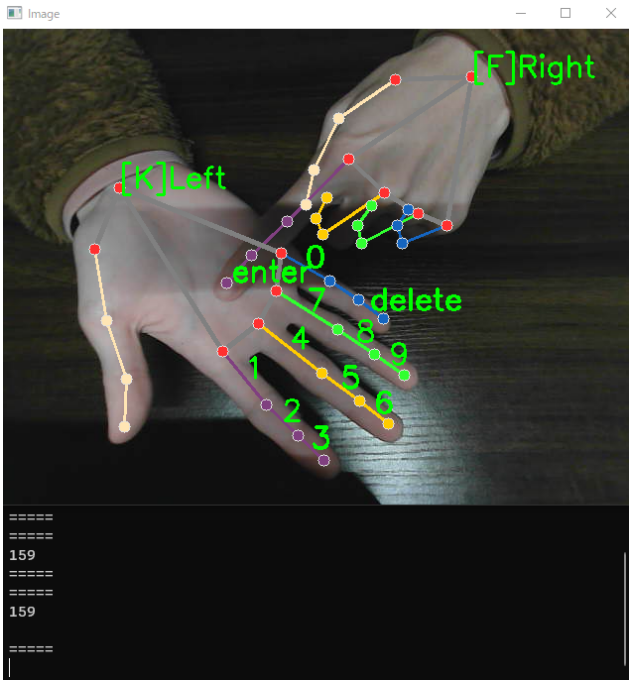

図 1　”1”, ”5”, ”9”, ”Eenter”と入力している例

1. プログラムを起動する。
2. カメラに向かってどちらかの手をパーの形にし、キーボードの検出を行う (図 2)。検出されれば手首のあたりに表示されている文字が F から K へ変化する。
3. 2 でもし逆の手を認識してしまった場合には、5 秒間カメラから両手が見えないようにする。
4. キーボードと認識した手とは逆の手の人差し指を使って、キーボードの任意の箇所に近づけることで文字入力を行う。どの数字がどの箇所に対応しているかは図 2 の通りである。

2.2　内部仕様

2.2.1　キーボード認識のためのプログラム

キーボード認識のために機械学習を用いた。具体的には、手の形がパーに近いかどうかを判定する決定木のモデルを作成した。決定木のモデルを作成するために scikit-learn 内のクラスである”sklearn.tree.DecisionTreeClassifier”を用いた。ただし、木の最大の深さは 3 に設定した [3]。

[*1] Meta Quest 3, https://www.meta.com/jp/quest/quest-3/
[*2] Vision Pro https://www.apple.com/apple-vision-pro/
[*3] https://www.mitsubishielectric.co.jp/business/biz-t/contents/pro-eye/pick016.html
[*4] https://www.dnp.co.jp/media/detail/10161948_1563.html

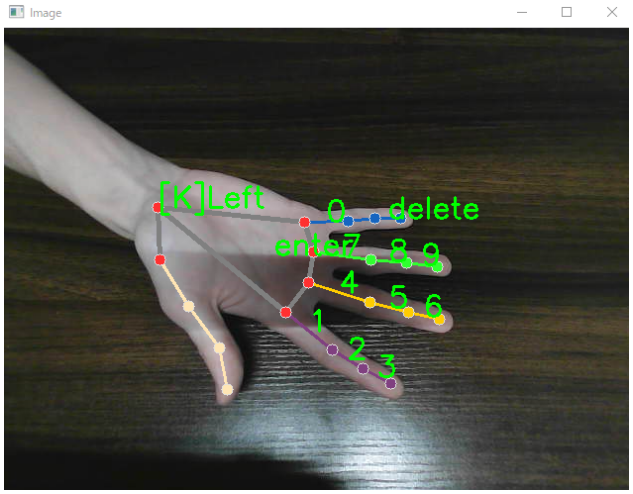

図 2　どの数字が手のどの箇所に対応しているか

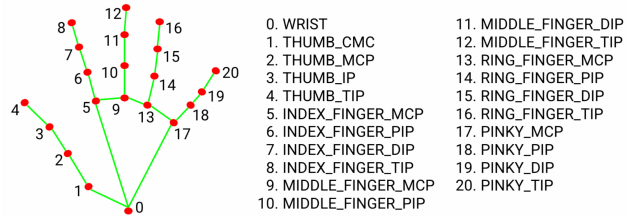

図 3　手の検出点 [4]

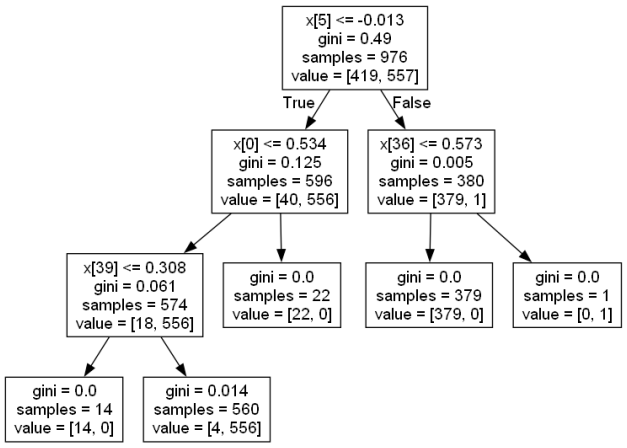

図 4　学習済みの決定木モデル

決定木の入力には、ラベルとしてパーであれば 1、そうでなければ 0、ターゲットとして手の各検出点の 3 次元座標計 63 種類の変数を使用した。ただし手の各検出点は図 3 に示すとおりであり、今回は簡単のため左手をキーボード用、右手を入力用とした訓練データのみを使用した。また、図 4 は graphbiz を用いて学習した決定木を可視化したものである [5]。

学習したモデルは pickle という Python の標準ライブラリを用いて、”models/clf_model.pickle”というファイルに保存し、メインの実行プログラムではそのファイルを読み込むことで学習済みモデルを使用できるようにした。

以下、2.2.2 で説明したファイルとその説明である。

- ”model/clf_model.pickle” = 学習済みモデル
- ”make_model.py” = 決定木のモデルの学習を行うプログラム
- ”make_dataset.py” = 訓練データを集めるためのプログラム

2.2.2　メインの実行プログラム

まず各クラスについて図 5 で示す。それぞれの役割を簡単に説明する。

- DrawingUtils = 描画用のメソッドを集めたクラス
- KeyboardDetector = キーボード認識を行うクラス
- KeyboardManager = キーボードの座標管理を行うクラス
- InputManager = 入力文字列を管理するクラス
- TouchDetector = 入力判定を行うクラス

”KeyboardManager”オブジェクトは、”KeyboardDetector”オブジェクトを常に監視している。また、”TouchDetector”オブジェクトは”KeyboardDetector”オブジェクトを常に監視し、入力文字に更新があれば”InputManager”オブジェクトへとデータの受け渡しを行う。

また、メインの実行部分の処理は以下の流れで行う。

1. 各クラスのインスタンスの初期化を行う
2. カメラの画像を読み込む
3. 手の座標を検出する
4. もし手が検出されればキーボード検出を行う
5. キーボードの座標を更新する
6. 入力の判定を行う
7. 描画する
8. もし ESC キーが押されたら終了、そうでなければ 2 へ戻る

2.3　使用した関数やライブラリ・実行環境

- Python 3.11.7
- graphviz==0.20.1
- matplotlib==3.8.2
- mediapipe==0.10.9
- numpy==1.26.3
- opencv-python==4.9.0.80
- pandas==2.2.0
- scikit-learn==1.4.0

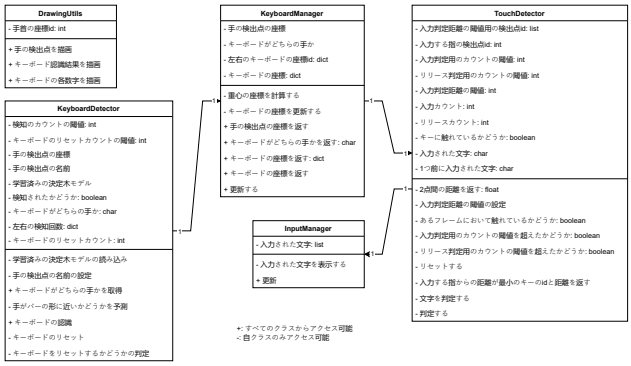

図 5　メインの実行プログラムのクラス図

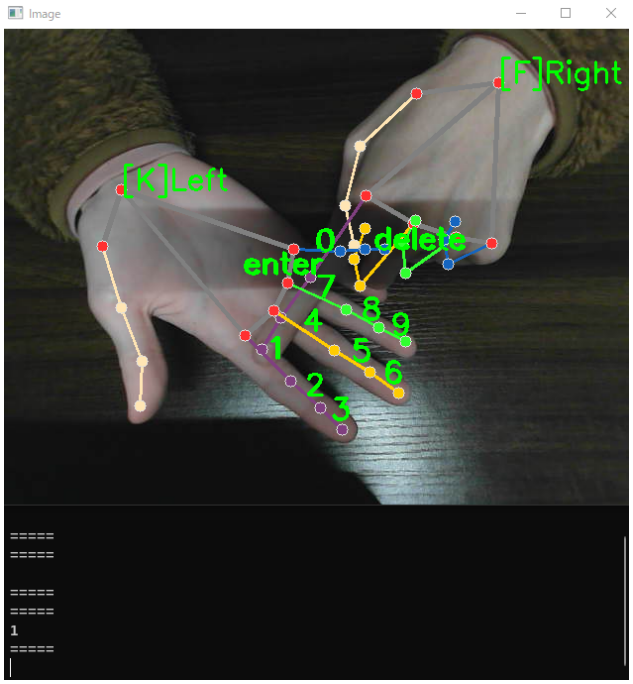

図 6　”1”と入力している例

3　実行結果

”1”と入力している例を図 6 で示す。図の上半分が入力画像であり、右手人差し指で、左手人差し指の付け根を指していることが分かる。さらに図の下半分は出力結果で、確かに”1”と入力できていることが読み取れる。

また、図 1 は”1”, ”5”, ”9”, ”Enter”と入力している例である。入力画像では最後の文字である”Enter”を入力したことが読み取れ、図の下半分を見ると確かに”1”, ”5”, ”9”, ”Enter(改行)”と入力できていることが分かる。

一方で、左右の手が重なりやすい”1”, ”2”, ”3”などを入力するときには、手の座標が検出されないことがあり、入力に時間がかかることがあった。

4　考察・感想

想定よりもスムーズに入力することができた。手の座標が検出されない問題については、mediapipe よりも性能の良いモデルを使用することで解決ができないかと考えた。

開発についての感想としてはまず、学習済みモデルの使い方を知ることができたため良かった。また、この課題を通してオブジェクト指向を意識したプログラムの作成方法が身に付いたため良い経験になったと感じた。

今後はこれを応用してフリック入力で日本語の 50 音を入力ができるようにしていきたい。その際には手首に加速度センサーを付けて情報量を増やしたりすることで、より精度を上げることができないか実験したい。

参考文献

[1] Mediapipe で手の形状検出を試してみた (Python) https://qiita.com/bianca26neve/items/116814135739929759a0

[2]【mediapipe 入門】ほんとに簡単に動くね https://qiita.com/MuAuan/items/11d1e69d34f9e741f515

[3]【Python】決定木の理論と実装を徹底解説してみた https://qiita.com/renesisu727/items/844648d6c60e578ce944

[4] Hand landmarks detection guide https://developers.google.com/mediapipe/solutions/vision/hand_landmarker

[5] Windows10 に Graphviz をインストールする https://0222-nnn.com/windows10%E3%81%ABgraphviz%E3%82%92%E3%82%A4%E3%83%B3%E3%82%B9%E3%83%88%E3%83%BC%E3%83%AB%E3%81%99%E3%82%8B/