平成 25 年度

筑波大学情報学群情報科学類

卒業研究論文

題目
LEDマトリクス光センサによる影画像の取得とジェスチャ認識への応用

主専攻　　知能情報メディア主専攻

著者　　梶孝行

指導教員　高橋伸　志築文太郎　三末和男　田中二郎

要　旨

遠隔で直感的に電子機器の操作を行うため、リモコンの代わりにユーザのジェスチャを利用する手法が研究されている。また部屋にいるユーザの状態を取得し支援を行うスマートルームを実現する場合においても、ユーザのジェスチャを取得することは重要である。近年では深度センサとカメラ等を組み合わせた Kinect を利用することで、簡単に人体のジェスチャを取得することができるため、これを利用することで大画面操作等の様々なインタラクションが実現されている。しかし Kinect やカメラを用いたジェスチャ認識ではユーザは監視されていると感じてしまう場合があり、ジェスチャの方向の制限せず認識を行うためにはカメラの数も増えその感覚は大きくなる。そこで床に生じる影に着目し、影の二次元画像を光センサによって取得することでプライバシーを重視したジェスチャ認識を行う手法を提案する。

本研究ではこのアプローチの有効性を実証するため、手によって生じる影画像を取得しジェスチャ認識を行うシステムを開発した。そして手を近づける、手を振るなどのジェスチャを実装しその動作確認と性能評価を行った。その結果素早く手を振った場合でもジェスチャ認識が行うことができ、アプローチの有効性を実証することができた。

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

遠隔でより直感的に電子機器の操作を行うため、リモコンの代わりに音声入力やユーザの動き、ジェスチャを利用する手法が研究されている。例えば、テレビの音量調節やチャンネル切り替えを手を振るなどの直感的なジェスチャによって操作したり、指の細かい動きを取得してマウスのように使用し操作するものなどがある。また部屋にいるユーザの状態を取得し支援を行うスマートルームを実現することにおいても、ユーザの状態を取得し、ジェスチャを認識することは重要な要素である。ユーザのジェスチャを取得するには深度センサやカメラなどを組み合わせた Kinect を利用することで、簡単にトラッキングを行うことができ様々なジェスチャインタラクションの実装に利用されている。他にも様々なインタラクションを実現する際にジェスチャ認識の技術が利用されている。

1.2　問題点と本研究の目的

人体のジェスチャ認識を行う手法としては、一般的にはカメラで撮った映像を解析する手法がある。また深度センサとカメラを等を組み合わせた Kinect などの 3 次元情報を利用したジェスチャ認識手法もある。しかしこれらの手法ではユーザは監視されていると感じる場合がある。またスマートルームを実現する際にはジェスチャを行う際の体の向きが制限されていない方が望ましい。しかし、これを実現するためには様々な角度からの映像が必要になり、使用するカメラの数が増え監視されている感覚を大きくしてしまう。本研究ではこれらの問題を解決する、つまりユーザに監視されている感覚を与えないような、ジェスチャ認識を行うことを目的とする。

1.3　アプローチ

本研究では上記の問題を解決するために人体によって床に生じる影に着目するジェスチャ認識手法を提案する。床に生じる人体の影はユーザの姿勢を反映しており、光源によっては様々な方向に影が生じるので死角の少ないジェスチャ認識が行えると考えられる。これを実現するためには床一面の影、つまり光の強さを取得する必要があるが、一般的に床一面の光の強さを取得するようなセンサは存在していない。そこで光センサをマトリクス上に配置し

て一面の光の強さを取得することを考える。本研究では光センサとして LED に着目し、入出力の可能な LED マトリクス光センサの実装を行う。

1.4 構成

本章では、カメラを用いたジェスチャ認識手法での問題点を挙げ、それを解決する手法の提案と本研究の目的を述べた。続いて第 2 章では関連研究について述べる。第 3 章では本研究で提案するシステムの説明を行い、第 4 章ではシステムの実装について詳細に述べる。第 5 章ではジェスチャ認識アプリケーションの性能評価を行った結果を述べ、最後に第 6 章で結論と今後の課題について述べる。

第2章　関連研究

2.1　プライバシー重視のジェスチャ認識に関する研究

プライバシーを重視したジェスチャ認識を行う研究として、床に伝わる圧力に着目し、足によるジェスチャを行うアプローチがある。圧力による足の入力は Multitoe[1] により提案されており、これを発展させることで Branzel らは床に生じる圧力を取得することでユーザの識別とトラッキングを行う GravitySpace を構築した [2]。GravitySpace では、床に伝わるユーザの靴跡や尻、手の圧力を分析することによりユーザの識別、また姿勢や体の傾きの推定を行い、その結果を利用することでインタラクションを行う。また床に生じる圧力に着目した研究には Srinivasan らの研究 [3] や Paradiso らの The Magic Carpet[4]、Robert の The Smart Floor[5] などがある。しかしこれらの圧力に着目した研究では、床に接していない空中の手を用いたジェスチャ等を認識することができない。これに対して本研究では、ユーザの影を使用することで体全体のジェスチャ認識を行うことができる。

一方で、手や体による影を用いることでジェスチャ認識を行うアプローチがある。Segen らは手と手の影よって 3 次元ジェスチャを行う Shadow Gesture を提案している [6]。Shadow Gesture では斜め方向から光を当て反対方向から手と手によって生じた影をカメラで撮る。この影の大きさや形を分析することで手の形と 3 次元位置を推定しジェスチャ認識を行う。また Cowan らはプロジェクタを手でふさぐことで生じる影によってジェスチャ認識を行う ShadowPuppets を提案している [7]。ShadowPuppets ではプロジェクタとカメラを組み合わせることでプロジェクタをさえぎる手の影を取得し、生じた影の形、大きさから手の 3 次元的な動きを取得しジェスチャ認識を行う。また、天井に取り付けたカメラから影を識別することで人物のトラッキングを行う Raheja らによる研究もある [8]。これらの研究は影をカメラでとってジェスチャ認識を行っているのに対して、本研究ではカメラを用いず、影を平面光センサで取得するためプライバシーを重視したジェスチャ認識を行うことができる。

2.2　光センサマトリクスを用いた研究

Dalton は LED マトリクスを使用することで入出力を行う TapTiles を提案している [9]。TapTiles では LED マトリクスによってテキストの出力を行う一方で、LED を光センサとして利用することでユーザの入力を行う。また秋田らは同様に LED マトリクスを入出力に用いる LEDtile を構築している [10]。TapTiles が影の取得に LED マトリクスを用いている一方で LEDtile では LED マトリクスに光を当てることを認識し、オルゴールなど様々なインタラクションに応用

している。また Echtler らは液晶スクリーンにおいてマルチタッチを実現するため赤外線 LED マトリクスを利用する手法を提案している [11]。この手法では液晶スクリーンのバックライトの間に赤外線 LED を使用することで液晶スクリーン上の光の変化を取得し、指がタッチされることで生じる影を認識することで、マルチタッチを実現している。

本研究では関連研究と同様に LED マトリクスを光センサとして利用することで、床に生じる影を取得しジェスチャ認識へと応用する。また LED マトリクスを光センサの入力として使用する一方で出力に用いる。

第3章　LEDマトリクスによる影画像取得システムとジェスチャ認識への応用

3.1　床の影によるジェスチャ認識

本アプローチでは床に生じる影に着目することで、カメラを用いないジェスチャ認識を行うことを目的としている。ユーザが部屋にいる場合、床には図 3.1 のようにユーザの姿勢が反映された影が生じる。これにより床の影画像を取得することでユーザの姿勢を識別することができる。そのため床に生じる影を追跡することでユーザの動きを取得することができ、手を振る、足を上げるなどのジェスチャ認識を行うことが可能になる。

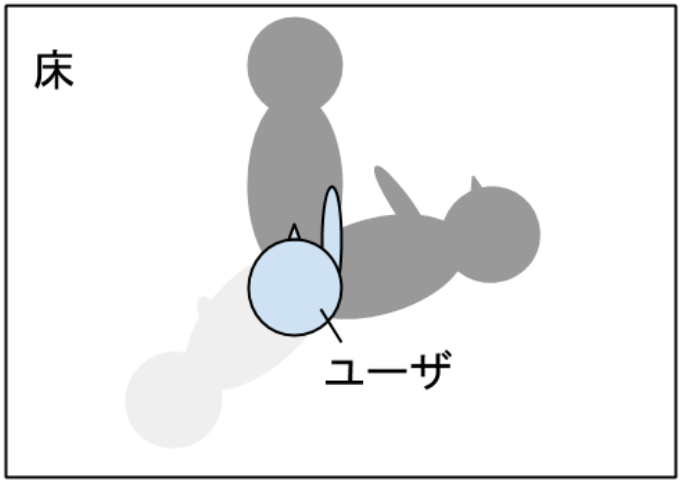

図 3.1: ユーザにより床に生じる影

また影は光源の位置と強さによって複数の方向に生じるため、一方向では死角になるような位置で手を動かすようなジェスチャであっても認識することができる。例えば本研究で想定するジェスチャの一つに手を振る動作がある。体の前方で手を振る場合、手の影は図 3.2 の左と右の画像のように右方向に生じる影を用いることで手の位置を取得することができる。しかし図 3.2 の中央の画像のように手が体の横に位置する場合、右方向の影においては手の影が体の影と重なり手の位置を取得することができない。そのため、右方向の影を分析するだけでは手を振るジェスチャの認識ができない。しかし手が体の横に位置する場合は上方向に

生じる影に手の影が映るため、この二つの影を用いることで手を振るジェスチャを認識することができる。

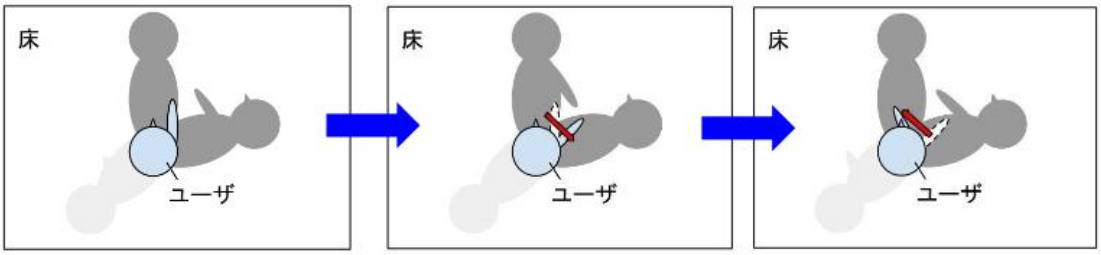

図 3.2: 手を振るジェスチャ

3.2　システム概要

本システムの全体像を図 3.3 に示す。本システムは、床面 LED マトリクスの一つ一つの LED から発生される電圧を読み取ることによって床に生じる影による影画像を取得し、ジェスチャ認識などに応用する。また床面 LED マトリクスをジェスチャ認識に利用する一方で、出力に利用することも可能である。

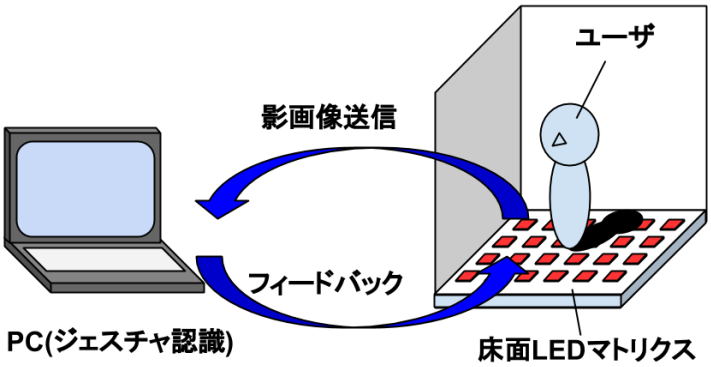

図 3.3: システム全体像

本システムの利用例として、床に画像を表示し、画像に向かって手を振るジェスチャをすることで、画像の切り替えを行うなどのインタラクションを考える。このインタラクションの実現に Kinect とディスプレイを利用する手法では、ユーザのジェスチャは Kinect がおいてある方向に制限される。反対に全方位からジェスチャが取れるような環境にする場合、周囲に Kinect を複数台設置する必要がある。またユーザが複数人いる場合に Kinect の正面にいるユーザによって奥にいるユーザの手の動きが死角になる場合があり、これも考慮する必要が

ある。しかし本システムを使用することで、ディスプレイとして利用しているLEDを用いてジェスチャ認識を行うことができる。そのためジェスチャ認識のためにカメラを置くスペースをとらず、また複数人のユーザが使用する場合でも影が重ならない限りジェスチャを認識することができる。

3.3 床面LEDマトリクスによる影画像の取得

影画像の取得を実現するために床一面の光の強さを取得する必要があるが、一般的に床一面の光の強さを取得するような巨大な平面光センサは存在していない。そこで光センサをマトリクス上に配置して一面の光の強さを取得することを考える。一般的に可視光に対して利用される光センサには受光素子であるフォトダイオードやフォトトランジスタ等があるが、本研究ではLEDに着目した。LEDは電流を流すことで光を発生させる電子素子だが、反対にLEDに対して光を当てると電子の移動が生じ、これを外部につなぐことで微量の電流を得ることができる。受光素子であるフォトダイオードなどと比較すると発生電流の大きさや応答速度は劣るものの、比較的安価で大量に利用することができる。安価で光センサとして同様に動作する電子素子としてはCdS等もあるが、LEDは出力としても利用できるという利点もあり、LEDを使用して実装を行う。

本研究で考える人体のジェスチャは体の姿勢、特に足や手の位置を使用したものである。またLEDは直上の光だけでなく斜めからの光にも反応している。そのため、影画像を取得するためにLEDを隙間なく配置する必要はなく、$20cm$程度の間隔を空けてLEDをマトリクス状に配置することで人体の影画像を取得することができ、期待するジェスチャが認識できると考える。本研究ではこれを床面LEDマトリクスと定義して扱う。

第4章　実装

4.1　LED光センサの特性調査

始めにLEDの光センサとしての性能を確かめるため評価実験を行った。LEDの光センサとしての性能はLEDによって違いがあるため、より適したLEDを選択するためいくつかのLEDで端子間の電圧を計測した。LEDは赤色LEDのOSDR5113Aと、高輝度赤色LEDのOSHR5161A-QR、OS5RCA5B61Pの三つを用意した。評価は部屋にいる状態を意識した複数の光源下での電圧、一方向から強い光を当てた場合の電圧、影を作った場合の電圧を計測することで行う。具体的にはLEDの両端にテスタをつなぎ、通常状態(研究室自席、蛍光灯下)での電圧、携帯によるライトを至近距離で当てた場合の電圧、通常状態から5cm上で手による影を作った場合の電圧、光が入らないように手でLEDをふさいだ場合の電圧を計測した。また各LEDの結果はそれぞれLEDをランダムに5つ選択し、計測した値を平均したものを用いている。計測結果は表4.1のようになった。

表4.1: LED特性調査結果

LED名	通常状態[V]	携帯による光[V]	手による影[V]	手によるふさぎ[V]
OSDR5113A	0.019	1.45	0.006	0.001
OSHR5161A-QR	0.045	1.61	0.017	0.008
OSR5CA5B61P	0.110	1.58	0.043	0.013

結果から強い光を当てた場合の電圧値は差が少なく、OSHR5161A-QRが一番高い電圧値を示している。しかし今回は影の濃さをより細かく取得することがより適しているため、通常状態から影を作った場合の変異が大きいOS5RKA5B61Pの5mm高輝度赤色LEDを使用し実装を行った。

4.2　開発環境

LEDマトリクスの実装には、LEDの電圧計測とPCとの通信が簡単に行えるArduinoを使用するため、開発環境、開発言語にArduinoを使用した。また開発には、Arduino Mega2560、先に述べたようにOS5RKA5B61PのLEDを使用した。

PC 側のジェスチャ認識アプリケーションの開発環境は Visual Studio2012、開発言語には C#を使用した。PC は CTO のデスクトップ型 PC で CPU は Corei7 Quad i7-920、メインメモリは DDR3 1333 1GB 3 本のものを使用した。

4.3 システム構成

本システムは LED マトリクスで読み取った影画像に対して、ジェスチャ認識を行うものである。今回は本システムの部分的な実装として手のひらサイズの LED マトリクスにおける、手のジェスチャ認識を行う。ユーザは LED マトリクス上で手を近づける、手を振るなどの動作を行うことでジェスチャを認識することができる。本研究ではこれを実現するための LED マトリクス光センサ、ジェスチャ認識アプリケーションのプロトタイプを作成した。

システムは LED マトリクスと通信用の Arduino、PC のアプリケーションから構成される。Arduino を用いることで LED マトリクスの個々の LED の電圧値を読み取り、読み取られた電圧値は定期的に PC のアプリケーションへ送信される。PC のアプリケーションは複数の Arduino からそれぞれ LED の電圧値を受け取り、これを画像として扱い合成することで大きな影画像を取得し、ジェスチャ認識へ応用する。

4.4 LED マトリクス光センサの実装

LED マトリクスは Arduino 一つで構築できる 4 × 4 の LED マトリクスを複数組み合わせることで部屋一面に配置できる巨大な LED マトリクスを構築し、ユーザの影を取得することを想定している。今回は Arduino を 4 つ使用することで図 4.1 のような 8 × 8 の LED マトリクス光センサを構築した。LED の配置はそれぞれの LED の間隔を $2.7cm$ とし、縦横 $18.9cm$ の大きさで実装した。LED の間隔と LED マトリクス全体の大きさは、手の影が取得でき、かつある程度動かしても LED マトリクスの領域内に入るような大きさで、さらに指を開いた際の隙間程度の間隔を各 LED に与えるような値とした。

LED マトリクス光センサは影により変化する LED の電圧値を読み取り、その結果を PC へ送信する。各 LED の電圧値を取得し、その結果を PC へと送信するため、LED マトリクス光センサの実装には Arduino Mega2560(以下、Arudino) を使用した。各 LED の電圧値は Arduino のアナログ入力を用いることで計測する。使用した Arduino は 16 本のアナログ入力を搭載しているため、16 個の LED の入出力が行うことができる。そのため一つの Arduino で 4 × 4 の LED マトリクスの計測を行い、これを組み合わせることで大きな LED マトリクスを構築する。各 Arduino の 4 × 4 の LED マトリクスは図 4.2 のような回路で構成される。

Arduino のプログラムは図 4.3 のようなアルゴリズムで動作するようにプログラムを作成する。また図 4.3 中の cnt は LED を読み取る回数であり、今回は通信速度とのトレードオフから 20 回とした。

Arduino は $0V$〜$2.56V$ の範囲の電圧を 0〜1024 に分解して扱うことができる。これにより各 LED の電圧をそれぞれ取得する。しかし、影が生じた場合の Arduino で計測した LED の

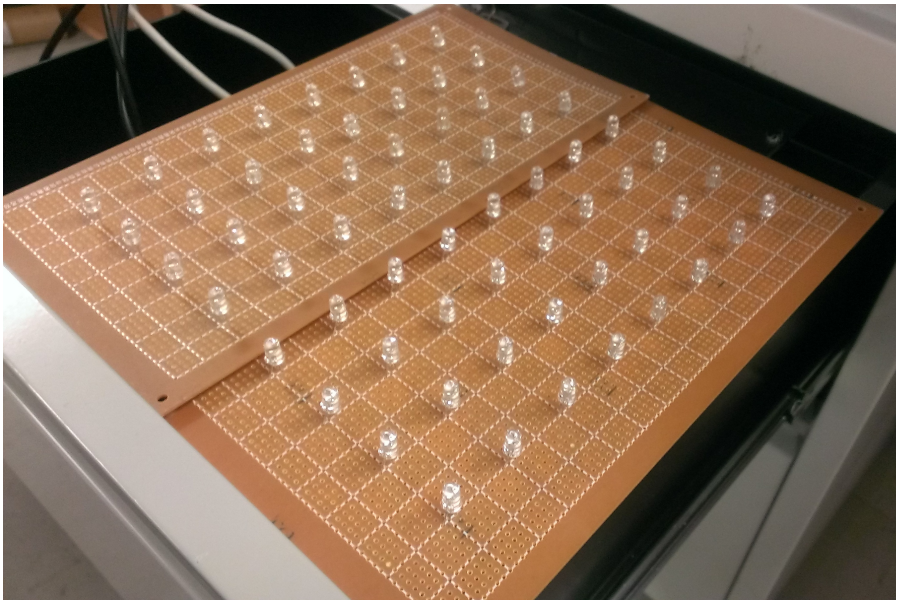

図 4.1: LED マトリクス

電圧値は影が移動しない場合でも一定ではなく、ある程度ぶれが生じている。そのため複数回 LED の電圧値を読み取り、その平均値を現在の電圧値として扱う。今回は通信速度とのトレードオフから、20 回の取得を行った。また LED の電圧を読み取る際、一つの LED を複数回読み取ってから次の LED を複数回読み取るという手順では LED の間でラグが生じてしまう恐れがある。これを最小限に抑えるため、16 個の LED をラスター走査の手順で一度読み取り、その次に二回目の読み取りを行うようにする。

Arduino と PC の通信はシリアル通信を用いて行う。このとき LED の電圧値を一つずつ送信した場合、PC のアプリケーションはシリアル通信によって得たデータがどの LED の電圧値であるかを判断する必要がある。これを解決するため、16 個の LED の電圧値を空白区切りで一つの文字列にまとめて送信する。PC のアプリケーションは受け取ったデータを空白で分解することで LED マトリクスのデータを取得する。また PC と Arduino の通信を同期させるために、PC から通信可能の合図である文字列「send」を受け取った場合のみシリアル通信を開始する。このとき一定時間 PC から合図がない場合は LED マトリクスの電圧を再取得する。

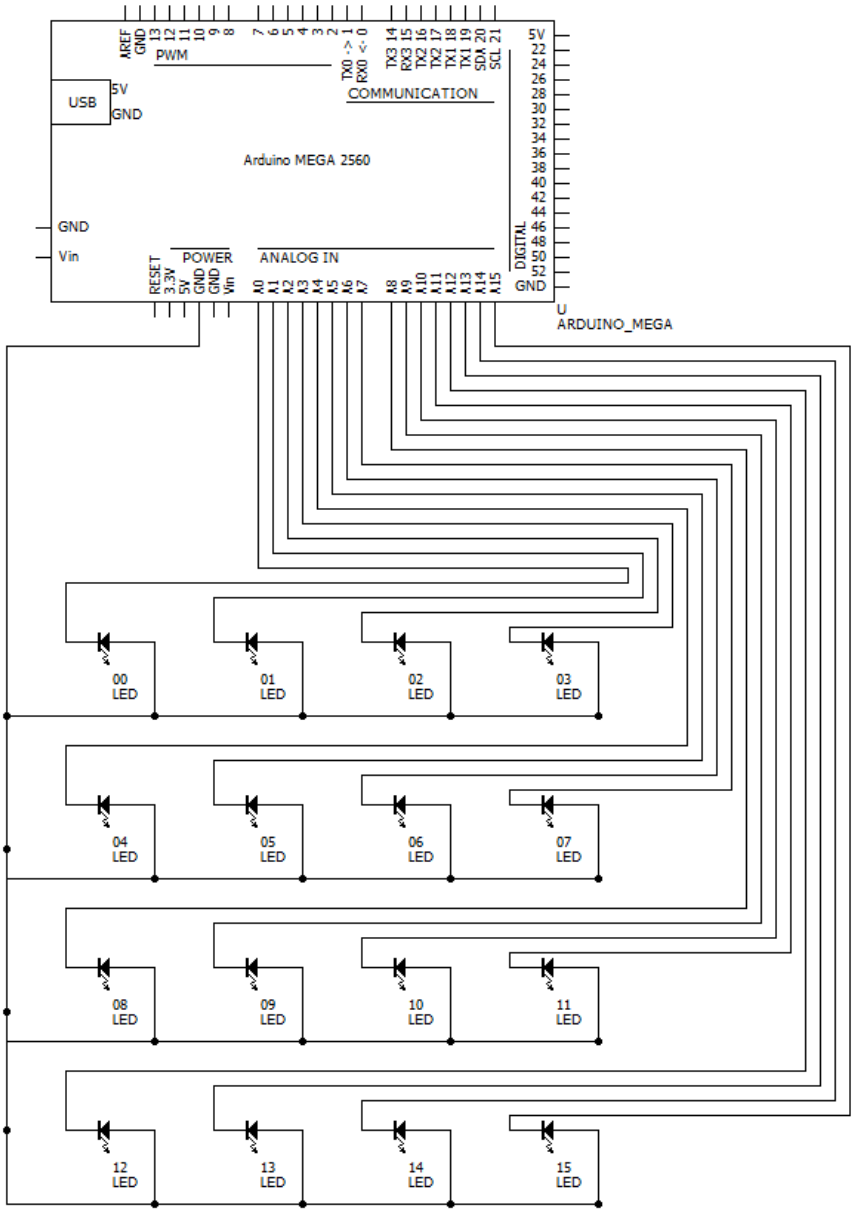

図 4.2: LED マトリクス回路図 (4 × 4)

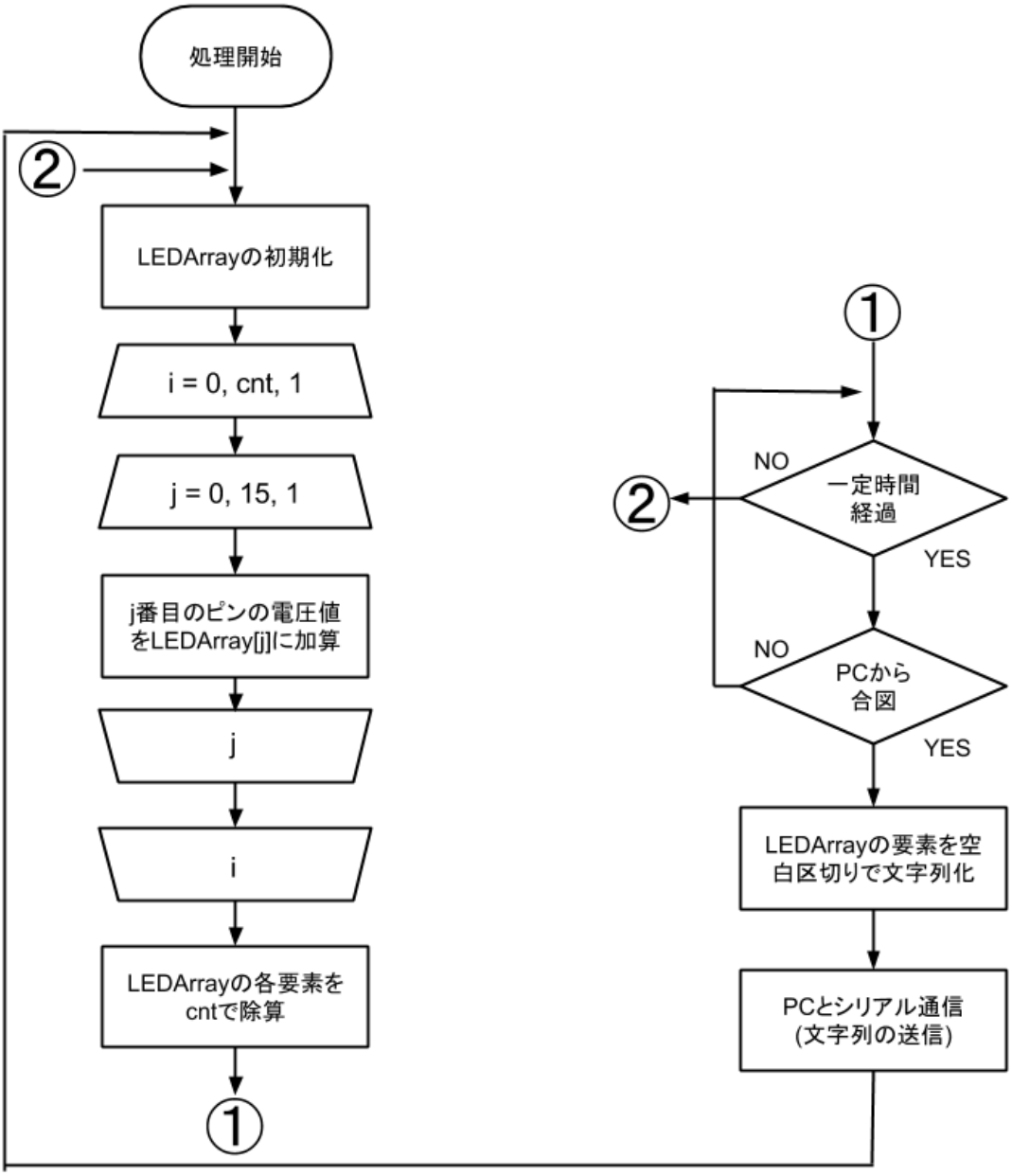

図 4.3: 影画像取得フローチャート

4.5　ジェスチャ認識アプリケーションの実装

4.5.1　Arduino とのデータ通信と影画像の作成

PC におけるジェスチャ認識アプリケーションでは、まず各 Arduino から LED の電圧値を受け取り、8 × 8 の影画像を作成する。一つの Arduino では 16 個の LED を計測できるためそれぞれ 4 × 4 の LED マトリクスの電圧取得を行う。そして 4 つの Arduino から受け取った電圧値を、図 4.4 のように画像として扱い合成することで 8 × 8 の影画像を作成する。

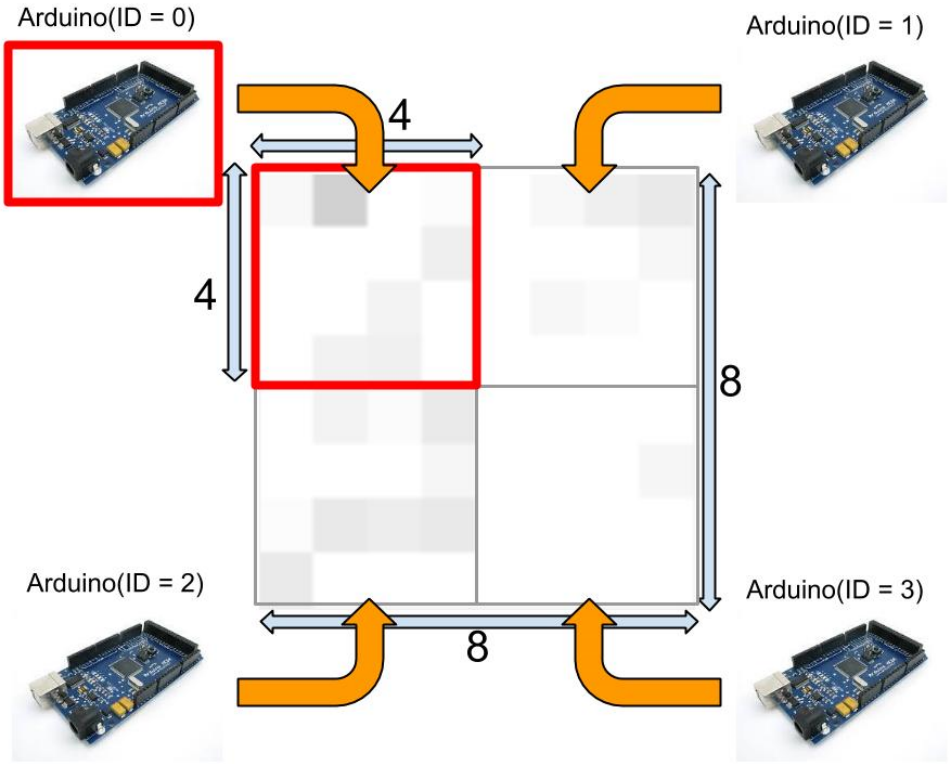

図 4.4: 影画像の合成

Arduino と PC の通信にはシリアル通信を用いるため、プログラムでは通信用に別々のシリアルポートを使用する。このときプログラムにおいて、各 Arduino がどの位置の LED を取得しているかを判断する必要があるため、接続の際にシリアルポートごとに ID を 0 から割り振る。この ID を用いることで、各 Arduino が持つ 16 個の LED の配列を　8 × 8 の影画像の対応した座標に代入していく。

使用する Arduino の数が N × N でそれぞれの Arduino は 16 個の LED を接続している場合、ID 番目の Arduino が持つ位置 (i, j) の LED の電圧値 Arduino[i][j] は式 (4.1)、式 (4.2) によって影画像 ShadowImage に代入される。なお式中の ul は ShadowImage における Arduino[0][0]、

つまり影画像の左上の位置であり、Arduino、ShadowImage はそれぞれ 4 × 4、8 × 8 の配列で画像を扱っているとする。

$$ul = 4 \times (ID \bmod N) + 16 \times N \times (ID/N) \tag{4.1}$$

$$ShadowImage[ul + i \times 4 \times N + j] = Arduino[i][j] \tag{4.2}$$

また影画像としてジェスチャ認識などに応用する場合には、できるだけ同じ時間に取得した 4 × 4 の影画像を合成する必要がある。これを解決するために、プログラムは Arduino からデータを受け取るごとに値の更新を行う一方で、タイマーをによって $50ms$ ごとに各 Arduino の最新の値を取得し画像化する。また動作確認を行ったところ Aruduino は LED の電圧値を読み取り PC に送信してから、次の結果を送信するまで平均 $15ms$ 程度の時間がかかり、遅くとも $40ms$ を超えることはなかった。これによりタイマーで取得している値は常に更新された最新のデータを取得していることが分かる。なおフレーム数をあげることは可能であるが、LED は影が生じた場合でもすぐに電圧値に反映されず、電圧値は段階的に下がっていく、そのため LED の示す電圧値と実際の影とは差が生じてしまう。そのためフレーム数をあまり上げても影画像の精度への影響はそれほど多くないといえる。

LED マトリクスから取得した影画像は LED には個体差があるのと、影がない状態での床の光の強さには微量の差があることから、値にばらつきがある。この状態では手の影の識別が困難であるので、影がない状態の画像からの差分画像を作成することで、これをジェスチャ認識を行う影画像として利用する。実際に作成したジェスチャ認識アプリケーションの全体像を図 4.5 に示す。

アプリケーションの構成を使用の流れから説明する。アプリケーションを起動後、最初に Arduino 接続設定を行う。4 つあるコンボボックスには接続している Arduino の COM ポートの一覧が表示され、それぞれ LED マトリクスと対応した位置にある Arduino の COM ポートをコンボボックスから選択する。その後コンボボックスの下にある ShowShadowImg のボタンを押すことで、問題なく接続されればログ表示に「Connection Success」と表示され、各 Arduino から受け取った電圧値を画像化し合成した 8 × 8 の影画像が影画像表示の左部分に表示される。また接続ができた場合再接続のエラー回避のため ShowShadowImg ボタンは押せないようにする。なお図 4.6 のように Arduino を設定してないもしくは同じ Arduino を複数選択している場合には接続が行われず、ログ表示に「Connection Error」と表示される。

プログラムでは Arduino との通信が成功したと同時にタイマーを有効にする。これにより以降タイマーの設定時間 $50ms$ ごとに画像が更新され表示される。次に影が生じていない状態で SetBase のボタンを押すことで、ログに「setBaseImage」と表示され、現在の影画像を差分画像のベースとして保存する。以降保存したベースと現在の影画像の差分画像を影画像表示部の右に表示する (図 4.7)。またベース画像は SetBase ボタンを押すことで更新できる。

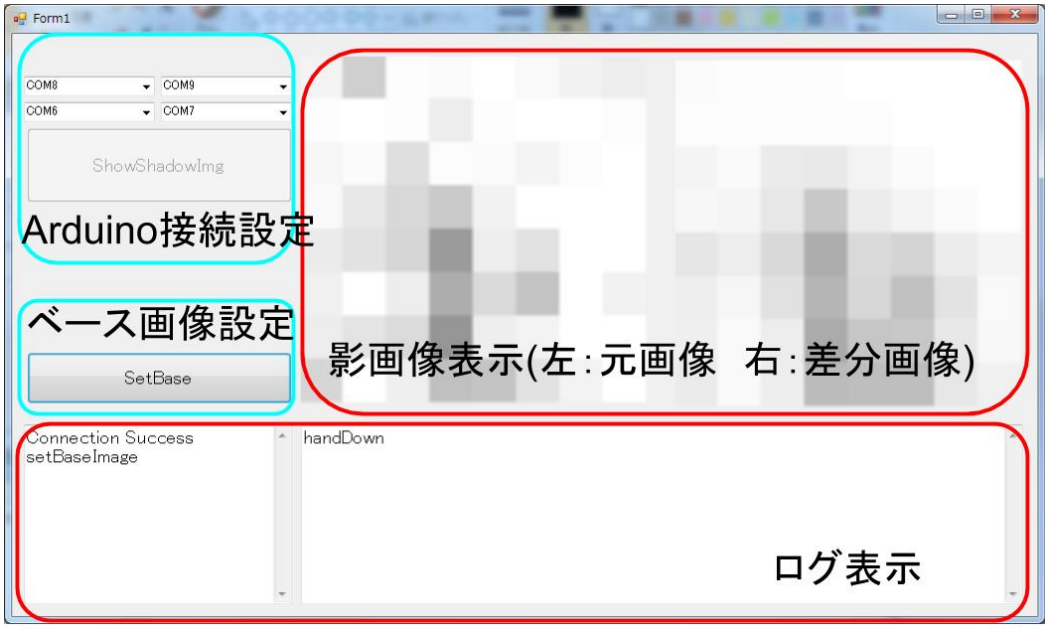

図 4.5: ジェスチャ認識アプリケーション

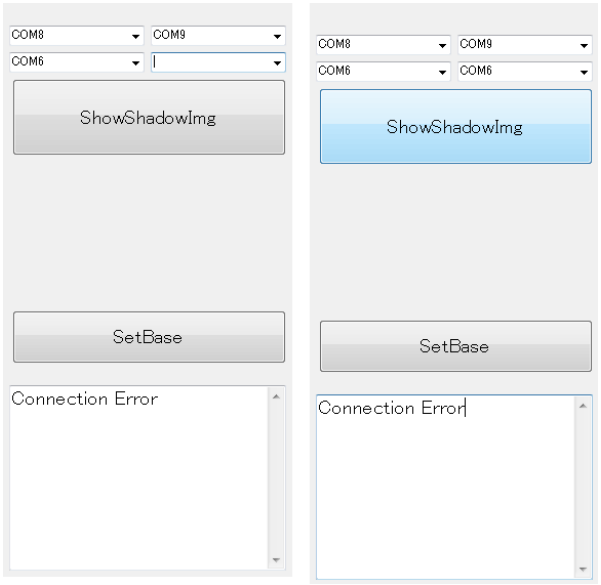

図 4.6: Arduino との通信失敗時 (左:Arduino 未指定　右:Arduino の複数指定)

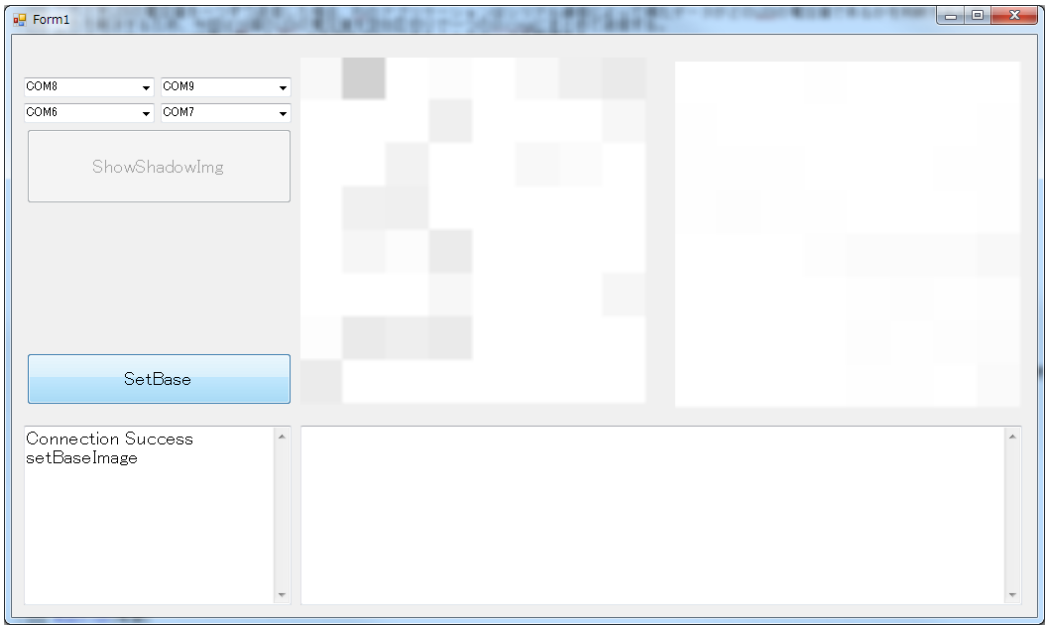

図 4.7: 差分画像の表示

4.5.2　ジェスチャ認識

本アプリケーションでは手の影画像を利用したジェスチャとして手を近づける、手を振るという動作に注目し、以下の 5 つのジェスチャを定義し実装を行った。

- handDown : 認識範囲内に手を近づける動作
- handUp : 認識範囲内から手を遠ざける動作
- waveRight : 手を左から右に振る動作
- waveLeft : 手を右から左に振る動作
- wave : 手を左右に振る動作

個々の LED は直上の光にだけ反応しているわけではなく、ある程度斜めの角度からの光にも反応している。そのため LED によって取得した影画像のうち、特に暗い部分は斜め方向の光も遮られる手のひらの中心部分である可能性が高い。つまり影画像のうち一番暗い領域を手のひらの位置として利用することができる。具体的なジェスチャの認識手法として、各ジェスチャは現在の状態と次の状態に遷移するための条件を持っておき、指定した条件を満たした場合に次の状態に遷移しすべての状態遷移を満たした場合にジェスチャ成功とする。条件は影画像の画素値に任意の閾値を設定し、指定した領域、例えば LED マトリクスの左から二列など、において閾値を超える画素値があるかどうかによって行う。またこれらのジェスチャ認識は差分画像を用いて識別を行う。

handDown handUp

「handDown」、「handUp」は LED マトリクス上に手を近づける、遠ざける動作を行うジェスチャである。手を近づけているかどうかは LED マトリクス全面のいずれかの領域で閾値を超えているかどうかで判定する。「handDown」のジェスチャは手を近づけていない状態から、手を近づける状態に遷移することで行う。反対に「handUp」のジェスチャは手を近づけている状態から、LED 全面のいずれの領域も閾値を超えていない、つまり手を遠ざける状態に遷移することで行う。

「handDown」と「handUp」を実際に行った場合の例を図 4.8 に示す。図は上の二枚が「handDown」を認識した時の手とアプリケーションの状態、下の二枚が「handUp」を認識した時の手とアプリケーションの状態を示す。

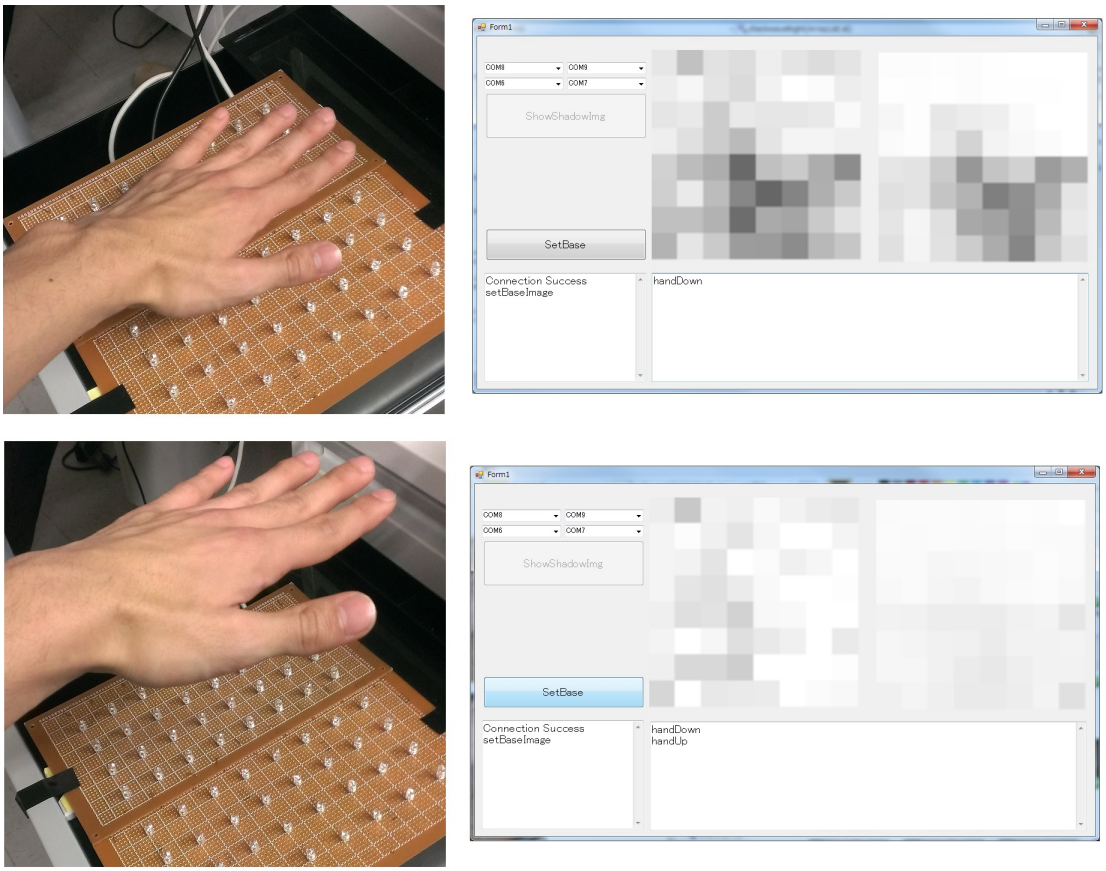

図 4.8: handDown 及び handUp の認識

図 4.8 の上の画像のように「handDown」は影がない状態から手を近づけて影を作る操作をすることで認識される。このときジェスチャの認識結果ログに「handDown」を表示される。そして図 4.8 の下の画像のように「hanDown」の状態から影がない状態に遷移すると「handUp」

のジェスチャとなり、ログに「handUp」と表示される。またこれから手を近づけた場合は「handDown」のジェスチャとなる。

なお図 4.8 は影画像で影が黒くわかりやすいような値を閾値としたが、閾値と光源の設定によっては 1m ほど上空からの影によるジェスチャ認識も行える。

waveRight waveLeft

「waveRight」、「waveLeft」はそれぞれ手を左、また右に振る動作を行うジェスチャである。「waveRight」のジェスチャははこれまでと同様に閾値を超える領域を判定し、以下の順番で状態を遷移させることで行う。

(a) 左 4 列の領域に手のひらの影があり、右 4 列の領域には閾値を超える影が存在しない。

(b) 中央 4 列の領域に手のひらの影があり、他の領域 (両端 2 列づつ) には影が存在しない。

(c) 右 4 列の領域に手のひらの影があり、左 4 列の領域には影が存在しない。

「waveRight」を行った際の動作を図 4.9 に示す。図 4.9 の画像は上から下の方向に手と、アプリケーションそれぞれの状態の遷移を示しており、同様に手を動かすことでジェスチャ認識が行える。まず図 4.9 の一段目のように左の領域に影があり右の領域に影がない場合に (a) の状態となり、ログに「waveRight: Start」と表示される。次に一段目の状態から二段目のように中央に影ができ両端に影ができていない場合に遷移したとき、(b) の状態に遷移しログに「waveRight: continue」と表示される。最後に二段目の状態から三段目のように右の領域に影ができ、左の領域に影ができていない状態になると (c) の状態に遷移し「waveRight: success」と表示され「waveRight」成功となる。そして図 4.9 の 4 段目は影が LED マトリクスの領域外に出ているため「handUp」のジェスチャが認識されている。また「waveRight」の認識途中、つまり (a) や (b) の状態で手を上げ「handUp」が認識されると「waveRight」は失敗となり、また (a) からジェスチャを行う必要がある。

「waveLeft」は「waveRight」を左右反転にした状態遷移によって認識する。「waveLeft」の状態の遷移は以下のようになる。

(a) 右 4 列の領域に手のひらの影があり、左 4 列の領域には閾値を超える影が存在しない。

(b) 中央 4 列の領域に手のひらの影があり、他の領域 (両端 2 列づつ) には影が存在しない。

(c) 左 4 列の領域に手のひらの影があり、右 4 列の領域には影が存在しない。

アプリケーションにおいては「waveLeft」も「waveRight」同様に、状態が遷移するとログに「waveLeft: Start」、「waveLeft: continue」、「waveLeft: Success」と表示する。図 4.10 は手を左から右に振り、その後左に振るという動作を行った後のアプリケーションの結果を示す。ジェスチャはまず「waveRight」を認識し、「waveRight」が完了したタイミング、つまり手が右にある状態で「waveLeft」は (a) の状態に遷移し、それから手を左に振ることで「waveLeft」の認識もできていることがわかる。

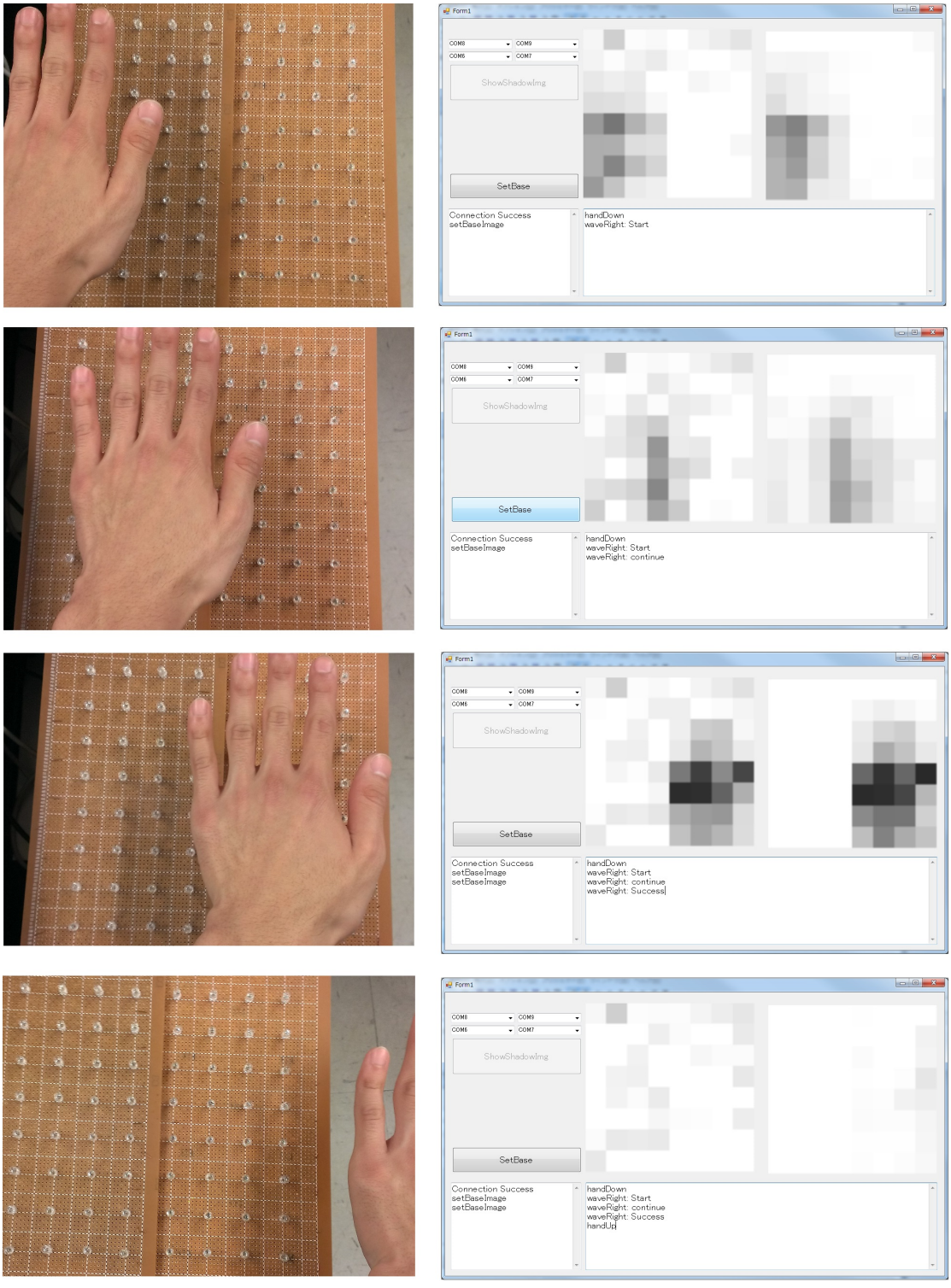

図 4.9: waveRight ジェスチャ認識の流れ

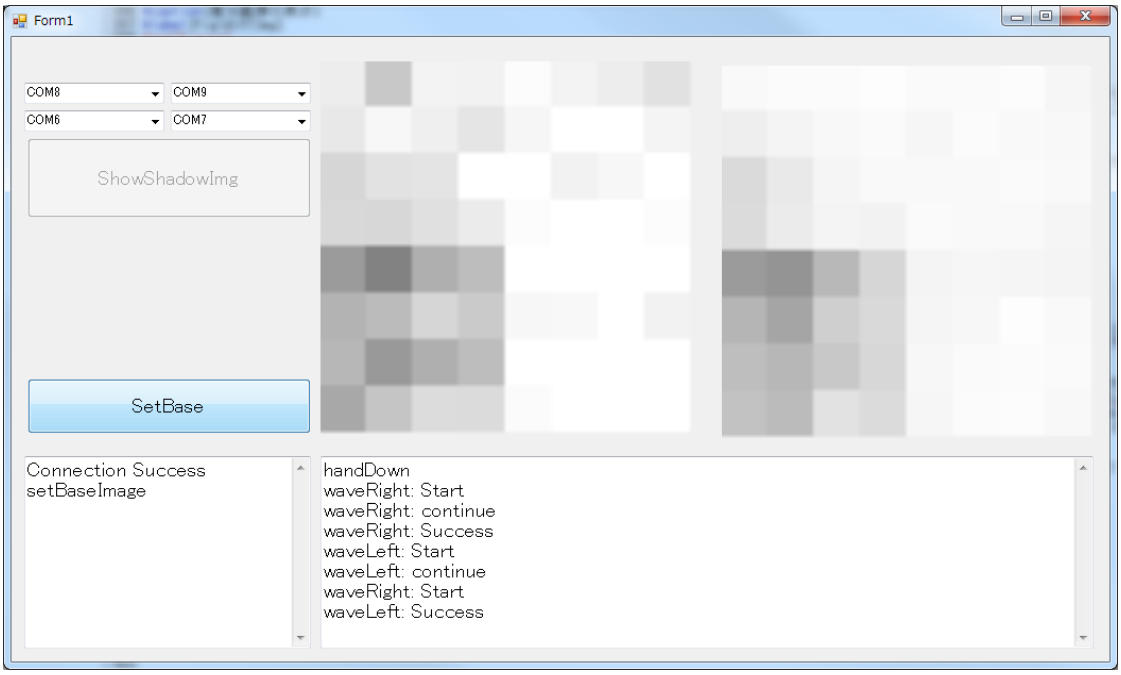

図 4.10: waveLeft ジェスチャ認識

wave

先に説明した「waveRight」「waveLeft」の応用として手を左右に振る動作「wave」のジェスチャを実装した。「wave」の認識は、「waveRight」「waveLeft」のどちらかが認識された場合に wave 開始とし、そこからは指定した時間内に反対に振る動作が続く限り「wave」継続、指定した時間を越えても反対の動作が認識できなければ「wave」終了とする。図 4.11 に「wave」を認識したときのアプリケーションを示す。「wave」の開始が認識されたときログには「wave: start」と表示される。それから指定した時間内に反対の動作のジェスチャが認識されればログに「wave: continue」と表示され、時間がリセットされる。指定した時間内に反対の動作がない場合はログに「wave: end」と表示される。なお図 4.11 では「handDown」「handUp」「waveRight」「waveLeft」のログ表示は省略している。

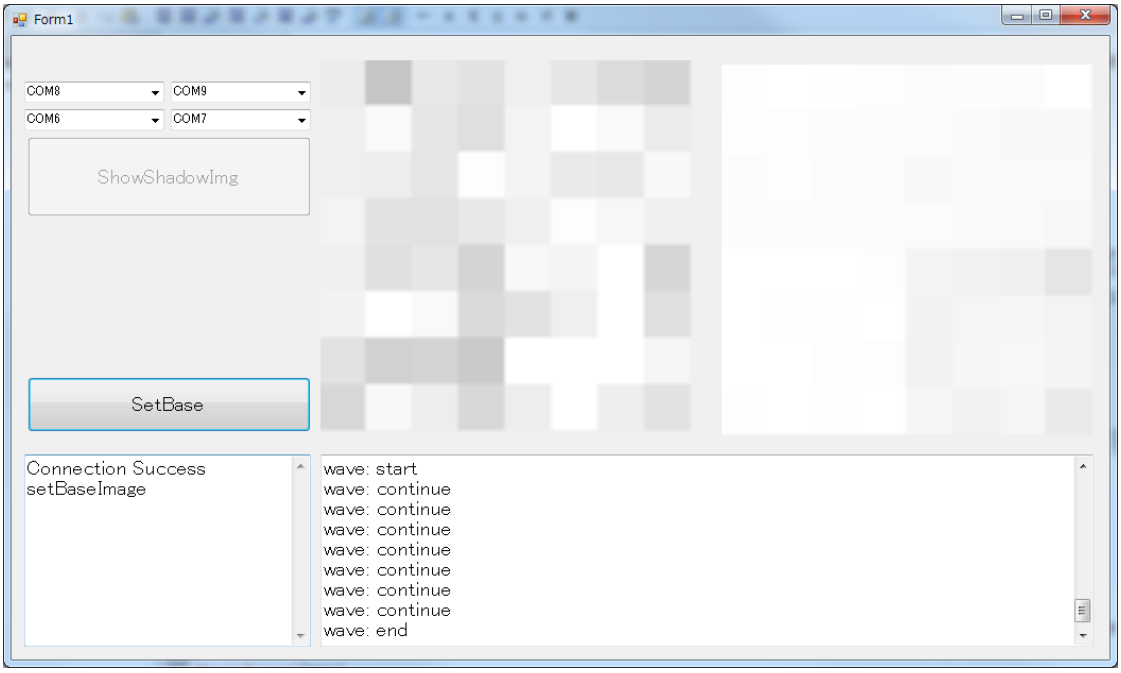

図 4.11: wave ジェスチャ認識

第5章　ジェスチャ認識精度の評価

作成したジェスチャ認識アプリケーションに対して、ジェスチャ認識の動作確認と、手の動きの速さを変化させた場合のジェスチャ認識の性能評価を行う。実験環境は複数の光源がある研究室の自席、被験者は著者 1 名で行った。

5.1　評価手法

本実験では手の動きを速くした場合にどこまでジェスチャ認識を行えるのかの評価を行う。認識するジェスチャには「waveRight」を使用する。「handDown」、「handUp」は手の速さにはあまり影響されないので、評価するジェスチャには含まない。また「waveLeft」は「waveRight」と対称のジェスチャであり、「wave」は「waveRight」「waveLeft」の繰り返しである。そのため手の動きの速さを変化させたときの影響は「waveRight」と変わらないので本実験では「waveRight」のジェスチャのみを用いる。

手の動きの速さを取得するために「waveRight」の開始時点、つまりログに「waveRight: Start」と表示した時点の時間から「waveRight: success」と表示するまでの時間を利用する。「waveRight」を行う際に手を左の領域外から右方向に動かした際に、「waveRight: Start」「waveRight: success」と表示されるときの手の位置は図 5.1 のようになる。図 5.1 のように手の移動量は LED マトリクスの大きさ、$19.8cm$ となり、これによって手の動きの速度を計算する。

実験は手の動きの速さをできるだけ固定して 20 回ジェスチャ入力を行い、どれだけジェスチャ認識が行えたかを評価する。ジェスチャ認識はログに「waveRight: success」と表示された場合に成功とし、その以外でジェスチャが止まる、あるいは始まらない場合はジェスチャ失敗とする。なおジェスチャ認識した結果の手の速度が基準から大きく異なる場合は無効としもう一度やり直しを行う。手の速さは何度か手の動きの速さを変えてジェスチャ認識を行った際に安定した値が出る 10、30、50、100[cm/sec] とした。

5.2　結果

評価結果を表 5.1 に示す。またジェスチャ失敗時の内訳を表 5.2 に示す。ジェスチャ失敗は continue まで (continue 状態から success 状態への遷移で失敗) と start まで (start から continue への遷移で失敗) に分けられる。

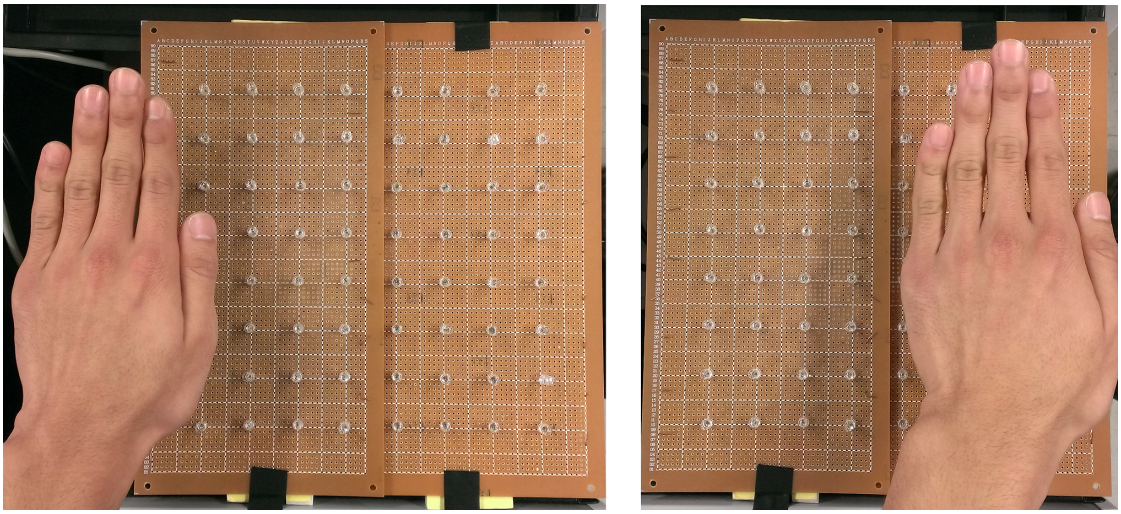

図 5.1: 左:「waveRight」開始時の手の位置　右:「waveRight」終了時の手の位置

表 5.1: 手の動きの速さによる精度評価結果

手の速度 [cm/sec]	ジェスチャ成功 [回数]	ジェスチャ失敗 [回数]	成功率 [%] (成功回数/総回数)
10	20	0	100
30	17	3	85
50	16	4	80
100	13	7	65

5.3　考察と議論

表 5.1 から手を速く動かすほどジェスチャ認識の精度は落ちていることが分かる。しかし手の速さが $100cm/sec$ でも半分以上はジェスチャ認識することができた。

表 5.2 より手の速度 30 での失敗は continue から success への遷移の部分であるのに対して、手の速度 50 や 100 は start から conitnue の遷移の時点で失敗する回数が増えている。これは手の動きに対して影による LED の電圧低下に時間差があるのが原因と考えられる。手の動きが速い場合、LED は影によって電圧が下がりきる前に影が移動してしまうことで元の電圧に戻ってしまう。その結果閾値をこえるまで影画像が暗くならず、アプリケーション側では「handUp」となり「waveRight」は失敗となる。この問題を解決するためには影による電圧低下の応答速度が速い LED を用意する、あるいは認識アルゴリズムの修正が必要である。

またジェスチャが失敗する理由の一つとして Arduino の 4 × 4 の LED マトリクスと他の Arduino との境界では他の箇所に比べて影の移動が滑らかでないという問題がある。そのため手の動きを速くした際に左半分から右半分に滑らかに影が移動せず「handUp」と判定されている状況がよく表れていた。この問題は今後改善する必要がある。

表 5.2: ジェスチャ失敗時の内訳

手の速度 [cm/sec]	continue まで [回数]	start まで [回数]
10	0	0
30	3	0
50	0	4
100	1	6

アプリケーションでの影画像の更新は時間を計測した結果平均 $60ms$ で行われている。そのため理論上は start から continue への遷移、continue から success への遷移が直ぐ行われれば $120ms$ 前後、$157cm/sec$ の速度のジェスチャも認識可能であるが、手を速く移動させても認識は一度もできなかった。

第6章　結論

本研究では、カメラを用いずジェスチャ認識を行う手法として床に生じる影に着目し、LED マトリクスによって取得された影画像を用いるアプローチを提案した。本手法 LED マトリクスで取得した影画像を用いることで、ユーザに監視されている感覚を与えずにジェスチャ認識を行うことが可能になる。また提案手法の実証として手の影画像を取得できる 8×8 の LED マトリクスとジェスチャ認識アプリケーションの実装を行った。その結果手を近づける、手を振るといったジェスチャ認識を行うことができた。また手を振る速さによるジェスチャ認識の性能評価を行い、素早い動きでもジェスチャ認識が可能であることを実証した。今後の課題として、影画像から手の形を識別する、入出力を利用した応用例の実装が挙げられる。

謝辞

本論分を執筆するにあたり、指導教員の高橋伸准教授、田中二郎教授をはじめ、三末和男准教授、志築文太郎准教授にはゼミやミーティングを通して大変貴重なご意見をいただきました。特に高橋准教授には研究の進め方、論文の執筆などに関して大変多くのご指導をいただきました。ここに深く御礼申し上げます。またインタラクティブプログラミング研究室の皆様には日常生活の中で様々な意見を頂きました。特にユビキタスチームの皆様には、ゼミだけでなく研究生活全般に渡って多くのご意見を頂きました。ここに深く感謝いたします。最後に自分の生活を支えてくださった家族、そして日常生活や研究生活において支えてくださった皆様に深く感謝いたします。

参考文献

[1] T. Augsten, K. Kaefer, R. Meusel, C. Fetzer, D. Kanitz, T. Stoff, T. Becker, C. Holz, and P. Baudisch. Multitoe: high-precision interaction with back-projected floors based on high-resolution multi-touch input. In Proceedings of the 23nd annual ACM symposium on User interface software and technology, PP. 209-218, 2010.

[2] A. Branzel, C. Holz, D. Hoffmann, D. Schmidt, M. Knaust, P. Luhne, R. Meusel, S. Richter, and P. Baudisch. GravitySpace: tracking users and their poses in a smart room using a pressure-sensing floor. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, PP. 725-734, 2013.

[3] P. Srinivasan, D. Birchfield, G. Qian, A. Kidane. A pressure sensing floor for interactive media applications. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer, PP. 278-281, 2005.

[4] J. Paradiso, C. Abler, K. Hsiao, M. Reynolds. The magic carpet: physical sensing for immersive environments. In Proceedings of the CHI'97 Extended Abstracts on Human Factors in Computing Systems, PP. 277-278, 1997.

[5] Robert J. Orr, Gregory D. Abowd. The smart floor: a mechanism for natural user identification and tracking. In Proceedings of the CHI'00 Extended Abstracts on Human Factors in Computing Systems, PP. 275-276, 2000.

[6] Jakub Segen, Senthil Kumar. Shadow Gestures: 3D Hand Pose Estimation using a Single Camera. Computer Vision and Pattern Recognition, IEEE Computer Society Confe-rence on, Volume: 1, PP. 479-485, 1999.

[7] Lisa G. Cowan, Kevin A. Li. ShadowPuppets: supporting collocated interaction with mobile projector phones using hand shadows. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, PP. 2707-2716, 2011.

[8] J. L. Raheja, Sishir Kalita, Pallab Jyoti Dutta, Solanki Lovendra. A Robust Real Time People tracking and Counting incorporating shadow detection and removal. International Journal of Computer Applications, Volume:46, No.4, 2012.

[9] Nicholas S. Dalton. TapTiles: LED-based floor interaction. In Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces , PP. 165-174, 2013.

[10] Junichi Akita. Interactive block device system with pattern drawing capability on matrix leds. In Proceedings of the CHI'12 Extended Abstracts on Human Factors in Computing Systems, PP. 1059-1062, 2012.

[11] Florian Echtler, Thomas Pototschnig, Gudrun Klinker. An LED-based multitouch sensor for LCD screens. In Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction, PP. 227-230, 2010.