平成 29 年度 卒業論文

Kinect を用いたプライバシー保護に配慮した

防犯システムの開発

指導教官 大野亙 准教授

畔柳拓実

2018 年 2 月 27 日

独立行政法人国立高等専門学校機構
豊田工業高等専門学校
電気・電子システム工学科

要旨

既存の防犯システムの問題点として、高度なシステムを持ち合わせながらも実際に利用されない点が挙げられる。この問題に対して本研究室では Kinect を用いた防犯システムを開発することで個人情報保護問題の解決を図った。プライバシー保護機能として、人を検出した時に人物の領域をシルエットで表示する。防犯機能として、危険行動をあらかじめ設定し、その行動を検出した時にカラー画像で表示する。通行者の個人情報を保護し、不審者は特定できるような防犯システムを構築した。

結果、目的の動作を行うプログラムの作成ができた。今後は危険行動を後から再生できる録画機能や、外出の際にも状況が分かるライブ配信機能などの追加を考えている。このような改善によって、Kinect を用いた防犯システムが実際に社会で使われることが期待できる。

Abstract

Surveillance cameras don’t easily become utilized despite its advanced systems. Our laboratory tried to solve the problem, privacy protection, by developing a security system using Kinect. As a privacy protecting function, when a person is detected, the person is displayed in silhouette. As a crime preventing function, threatening movement is set in advance, when the action is detected, it allows the threat to be displayed as a color image. I developed a crime prevention system that protects personal information of pedestrians while identifying crime scenes.

As a result, I have created a program that cleared my previously mentioned aim. I am planning to add some functions such as video recording, which enables the user to replay the video afterwards and live viewing function which enables the user to check on the system while outside. With such improvements, I expect that the security system using Kinect actually become utilized in our society.

目次

第 1 章 はじめに

1.1 背景

監視カメラ需要の大幅な伸びが予想されている。監視カメラ需要は過去四年間増加傾向にあり、2019 年には約 500 億円に達すると見込まれている。

日本国内では 2020 年の東京五輪に向けて、競技場だけではなく鉄道、道路など周辺のセキュリティ需要が高まっている。西欧では相次ぐテロリズムにより同様にテロ対策が課題となっている。東南アジアの国々をはじめとする諸外国では、軽犯罪と警官の不正防止のために、店舗や商業施設などに監視カメラを設置する動きが高まっている。監視カメラの需要が増加するとともに、より高度なシステムへの移行が考えられている。

既存の防犯システムは人工知能を用いたものが流行している。例えば、NEC では顔認証防犯システムの運用が開始されている。ライブ映像、過去の映像、画像ファイルから不審者の検索や絞込みが可能であり、監視用カメラ映像から切り出した顔画像と事前に登録した特定人物の顔画像を自動照合しカメラが類似の顔を捉えたら発報などでスマートフォンに通知する仕組みだ。本研究で大量の学習データを持つ企業に対抗できるシステムを生み出すことは困難である。そこで、対抗するのではなく、共存できるようなシステムを開発したいと考えた。

このような高機能な防犯システムの問題点として、高度なシステムを持ち合わせていても実際に利用されない点が挙げられる。これは、監視カメラに付加機能を与えるばかりに人の行先や感情、興味が判別されるようになるかもしれないという監視社会への抵抗感があるためである。個人の保護とユビキタス社会[1]の推進にとってはプライバシー、個人情報の保護が重要な課題となる。本研究では、Kinect を用いたプライバシー保護に配慮した防犯システムを開発した。

1.2 目的

「犯行現場を全て映しだす」ことを目的にする防犯と、「個人に帰属する情報を映さない」ことを目的にするプライバシー保護は相反するものである。そのため、両立した防犯システムの開発は難しいと考えられてきた。しかし、この問題は Kinect を用いることで解決できるのではないかと考えた。

個人情報保護委員会(2017)は、「防犯カメラの撮影により得られる容姿の映像により、

[1] ユビキタス社会…総務省「情報通信白書（平成 16 年版）」によれば、「『いつでも、どこでも、何でも、誰でもアクセスが可能』なネットワーク環境」を、ユビキタス社会として定義付けしている。

特定の個人を識別することが可能な場合には、原則として個人情報の利用目的を本人に通知又は公表しなければなりません。」と公表している。また、人物の顔や服装が隠され個人が特定できない状態では個人情報の保護に関する法律を違反しているとはいえず、責任問題を問われないと考えた。

実際に、個人情報保護委員会(2017)が発行する『「個人情報の保護に関する法律についてのガイドライン」及び「個人データの漏えい等の事案が発生した場合等の対応について」に関するＱ＆Ａ』では、次のように指摘している。

Q1-13
カメラ画像から抽出した性別や年齢といった属性情報や、人物を全身のシルエット画像に置き換えて作成した移動軌跡データ（人流データ）は、個人情報に該当しますか。
A1-13
個人情報とは、特定の個人を識別することができる情報をいいます。性別、年齢、又は全身のシルエット画像等による移動軌跡データのみであれば、抽出元の本人を判別可能なカメラ画像や個人識別符号等本人を識別することができる情報と容易に照合することができる場合を除き、個人情報には該当しません。

従って、Kinect を用いたシルエット表示を行う本システムは、プライバシー保護の問題を一気に解決する手段として一考の余地があると考えられる。

また、予め危険行動[2]を設定しておくことで不審者を検出した時のみ人物を表示するプログラムが作成できる。警備員や万引き G メンに代表される、犯罪者の行動特性に熟知した人は長年の勘に従って不審者を特定することが多い。それを勘ではなくパターンから学習させることができれば、システムの大幅なコスト削減を達成することができる。Kinect という新たなテクノロジーを用いることで問題の解決を図った。

[2] 危険行動…本研究では「犯罪や暴力事件の発生、そのほか他人の損害に繋がる不審行動」のことを危険行動と定義する

第 2 章 システム概要

2.1 Kinect とは

Kinect とは、Microsoft から発売されたナチュラルユーザーインターフェイス[3]を特徴としたコントロールデバイスである。本来は Xbox というゲーム機器に接続して専用のソフトウェアでゲームを楽しむためのものであるが、PC にも接続することが可能である。Microsoft 社から配布されている Kinect 開発キット Kinect for Windows SDK 2.0 を用いることで Kinect を純粋なセンサとして使用し、ただゲームを楽しむだけではなくジェスチャーを用いたコントロールデバイスとして使用することができるようになる。

2.2 システム動作

本研究室では Kinect を用いることで個人情報保護の問題解決を図った。

本システムは 3 つの動作部で構成される。データ取得を行う Kinect、データ処理を行うデバイス、そしてデータを保存しておくローカルサーバーである。図 1 にシステム構成を示す。

図 1　システム構成

[3] NUI…コンピュータのユーザーインターフェース（UI）のうち、人間にとってより自然な・直感的な動作で操作可能な仕組みや方法のことである。より直接的に対象を指示し、その分だけ直感的に操作できるといった点を特徴とする。

開発したシステムには大きく二つの機能が備わっている。

一つ目の機能はプライバシー保護機能である。前述した個人情報保護に関する法律や、監視社会への懸念の払拭、ユビキタス社会の推進を目的として設定した。これらの問題を解決するために、Kinect の人物領域認識技術を利用する。Kinect は二つの赤外線センサと RGB カメラによって人物の領域を認識することができ、これにより、人物の範囲を塗りつぶして表示する。

二つ目の機能は防犯機能である。危険行動をあらかじめ設定し、その行動を検出した時にカラー画像で表示する。歩行者の個人情報を保護し、不審者は特定できるような防犯システムを構築した。

2.3 利用したツール

Windows 用の Kinect 開発ツールである Kinect for Windows SDK v2、Kinect から取得できるデータを記録・再生できるツールである Kinect Studio、そしてジェスチャー認識のための機械学習ツールである Visual Gesture Builder（以降 VBG)を用いた。VGB は、機械学習によってジェスチャーの特徴を学習・認識するために用いるツールである。これによって、以前は人が経験則に基づき設定しなければならなかった関節の位置情報を、開発者が直接閾値を決定することなく決めることができる。

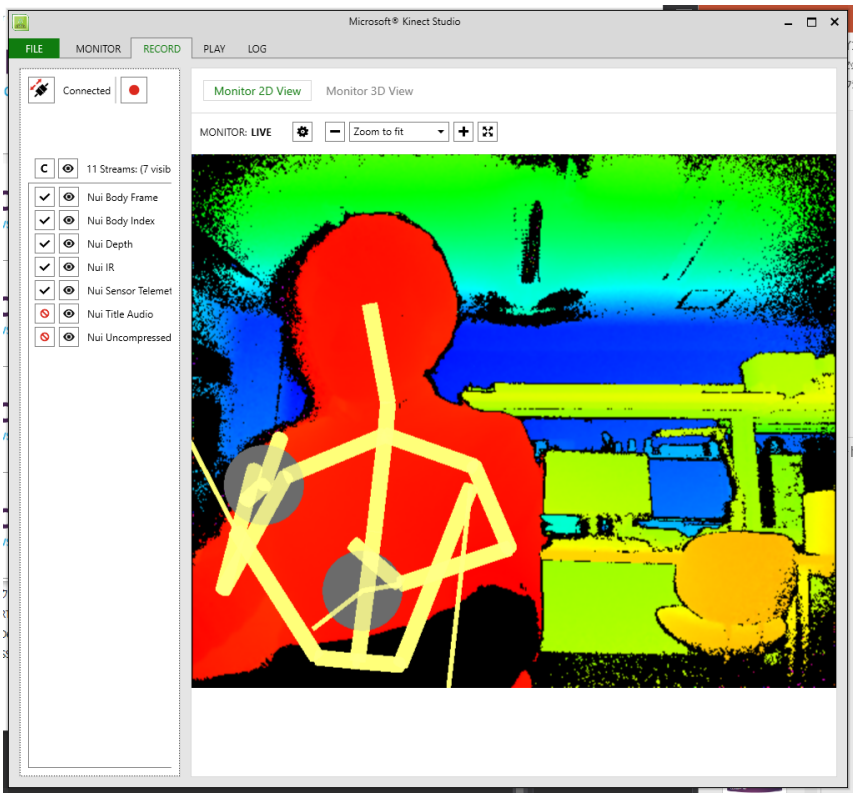

図 2　Kinect Studio 実行画面

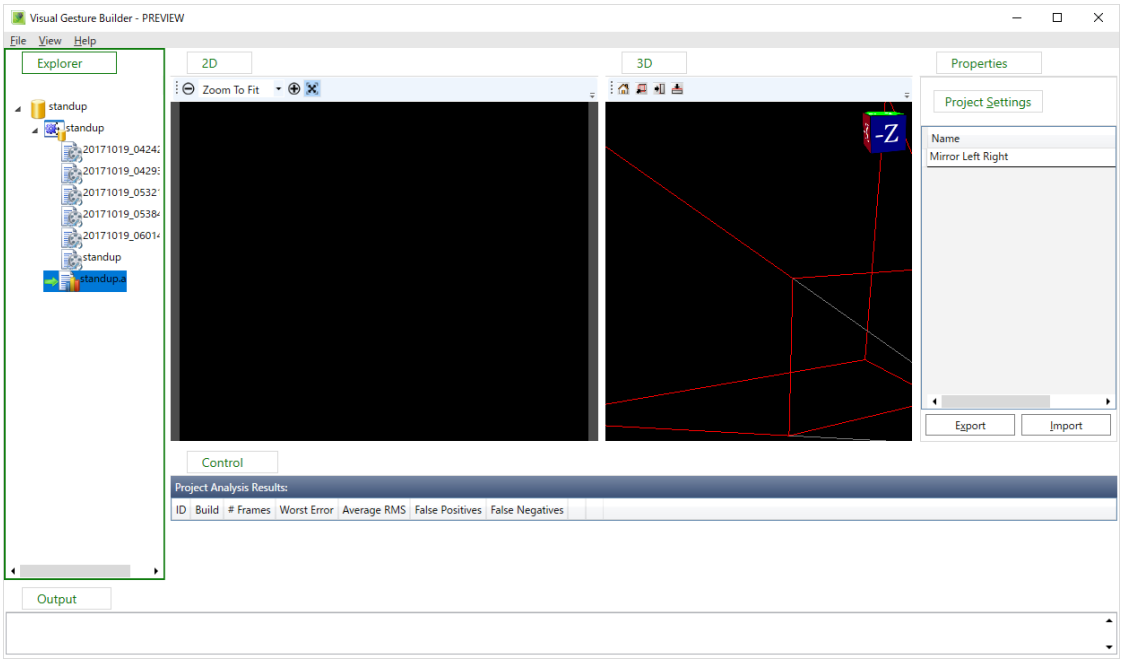

図 3　Visual Gesture Builder 実行画面

2.4 プログラミング言語

システム内部には C#を、ユーザーインターフェイス実装部分には Xaml を用いた。

Kinect は C++で開発されることも多いが、参考文献が充実していたことから、C#を選択した。C#とは、Microsoft 社が Java などをベースに独自に開発した言語である。

Xaml は、Microsoft が開発した XML ベースのマークアップ言語である。アプリケーションのビジュアルプレゼンテーションの背後に存在する。

2.5　先行研究

先行研究においても Kinect を用いた防犯システムの開発を行っている。怪しい行動をユーザー側が設定することによって怪しい行動の検出を可能にし、行動が怪しいかどうかによって警報を鳴らす夜間防犯システムを目的に開発されていた。しかし、ここで使用されたる Kinect v1 は、本研究で使用した Kinect v2 と互換性がない。先行研究のプログラムやノウハウが使用できないにもかかわらず私が Kinect v2 を採用したのは、デバイスやソフトウェアの性能が向上したためである。Kinect v1 と Kinect v2 センサーの仕様比較を表 1 に示す。

表 1　Kinect v1 と Kinect v2 センサー仕様比較

		Kinect v1	Kinect v2
色（Color）	解像度（Resolution）	640×480	1920×1080
	fps	30fps	30fps 15fps（暗所）
深度（Depth）	解像度（Resolution）	320×240	512×424
	fps	30fps	30fps
人物領域（Player）		6 人	6 人
人物姿勢（Skeleton）		2 人	6 人
関節（Joint）		20 関節／人	25 関節／人
手の開閉状態（Hand State）		△	○（グー、チョキ、パー）
深度の取得範囲（Range of Depth）		0.8～4.0m （Near Mode 0.4m～） （Extended Depth ～10.0m）	0.5～8.0m
人物の検出範囲（Range of Detection）		0.8～4.0m （Near Mode 0.4～3.0m）	0.5～4.5m
角度（Angle）（Depth）	水平（Horizontal）	57 度	70 度
	垂直（Vertical）	43 度	60 度
USB バージョン		USB 2.0	USB 3.0
1 台の PC で複数 Kinect 同時利用		×	○

仕様性能の比較において特に重視したのが、画像の解像度である。カラー画像の解像度が従来の 640×480 から 1980×1080(Full HD)に大幅に上昇した。これによって、人物認識精度が大きく向上した。さらに、深度画像の解像度も向上した。従来の近赤外線光を用いた Light Coding とは異なり、赤外線光を用いた Time of Flight 方式が採用されたことにより日光などの影響を受けにくくなった。

さらに、VGB が Kinect v2 向けに公開された。先行研究では怪しい行動の設定をユーザーが設定する方法を取っていた。課題は、怪しい行動というものをシステム管理者が決めなければいけないため非常に面倒なことが挙げられていたが、この課題は VGB の機械学習を用いることで容易に解決することができると考えた。

第 3 章 Kinect プログラミング理論

3.1 データの取得

Kinect SDK v2 では、Kinect Sensor から Frame Source を取得、Source から Reader を開き、Reader から Frame Reference を取得、Frame からデータを取得するという流れになった。Sensor とはメイン機体のことであり、Frame Source はデータの流れを決める５つのデータ群のことである。Reader は一つの Source に対して複数開くことができるため、観測物を指定するために用いる。短時間で上書きされる Frame 画像から、生データを取り出すことができる。データの取得方法について、図 4 に示す。

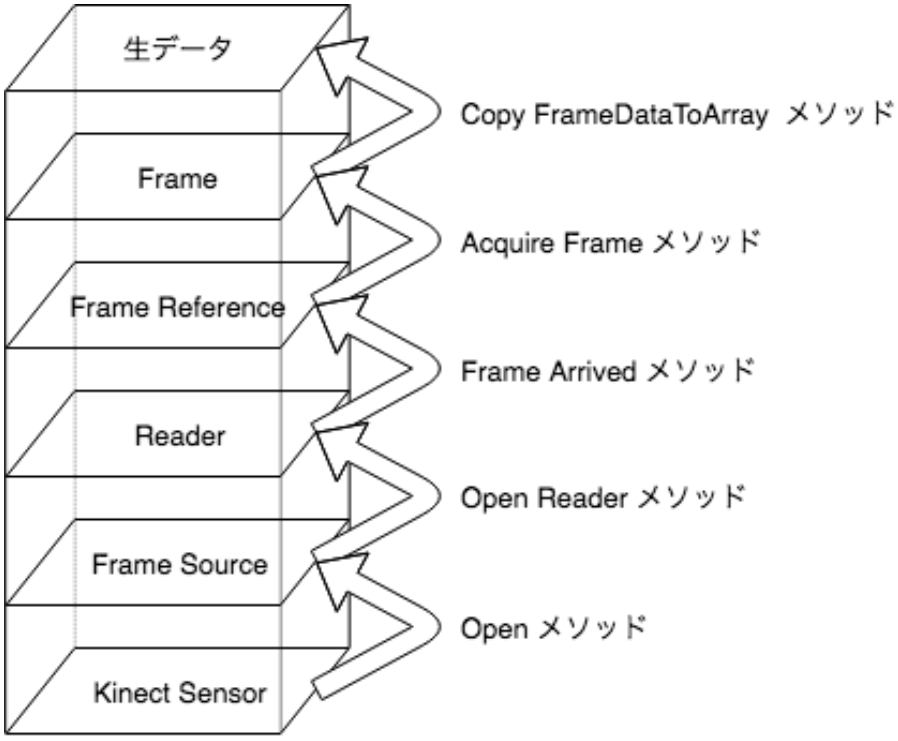

図 4　Kinect データの取得方法

3.2 座標変換

座標変換 Kinect v2 から取得できる基本ストリームのデータは 5 種類存在する。Color（色画像）, Depth（深度画像）, Camera（３次元座標）, Body（骨格）, BodyIndex（人物領域）, Infrared（赤外線）であり、それぞれ座標系が異なる。座標系の特性の違いや各ストリームから得られる情報について表 2 に示す。

表 2 ストリームデータと座標系の対応

座標系	ストリームデータ	概要
カラー座標系	Color	1920×1080 ピクセル 2 次元座標
深度座標系	Depth Infrared BodyIndex	512×424 ピクセル二次元座標
カメラ座標系	Body	メートル単位三次元座標

最初読み込んだ Color データは、4 バイトの BGRA 形式で保存される。A（透過）チャンネルは画面表示に関係ないので、データを小さくすることを考える。byte 型の配列を用意し、3 バイト RGB 形式に変換する。

Depth データの解像度は、Kinect v1 では 320×240 であったが、Kinect v2 では 512×424 に向上している。また、奥行き方向の分解能も向上している。Depth データの単位は mm（ミリメートル）であり、フォーマットは一種類だけなので形式の変換は必要ない。1 ピクセルあたり 16 ビットのデータが解像度の大きさだけ並んでいるおり、配列の要素にアクセスすることで任意の座標の距離を取得できる。

BodyIndex データは 1 ピクセルあたり 8 ビットのデータが 512×424 個並んでおり、配列の要素にアクセスすることで任意の座標の BodyIndex データにアクセスすることができる。BodyIndex データは人物領域を表す。大量の人物画像で学習しておくことによって Kinect は人物領域を判断する。人を検出していない場合には 255、人を検出している場合には 0~5 の値が代入される。

Body ストリームは人物の数を判定することにのみ利用したため、座標変換や画面出力をする必要がなかった。

Infrared ストリームは本研究では使用しなかった。

上記のような取得した座標系間のフレーム、座標単位の対応関係をもとにそれぞれのデータの位置合わせと、座標変換を行なった。

3.3 機械学習

KinectV2 には、機械学習を使ってジェスチャーを認識するための識別器を作成するツール、Visual Gesture Builder が存在する。開発者が直接閾値を決定する必要がなくなった。

二つのアルゴリズムを動作の種類によって使い分け、学習させることができる。それぞれの違いを表 3 に示した。

表 3　Visual Gesture Builder で用いられる二つのアルゴリズム

アルゴリズム名	Adaboost	Random Forest
アルゴリズム概要	弱い識別機を適用し、誤分類してしまったものの重みを増やし、重みがついたものを優先的にみて分類することを連続で行う。ランダムより高確率で２値に分類する手法。	大量の決定木を作成して、それぞれの決定木が出した答えを多数決し、最も支持の多かったクラスに分類する手法。集団学習アルゴリズム。
動作の種類	Discrete（離散的）	Continuous（連続的）
型	Bool	浮動小数点
特徴	複数のジェスチャーを同時に判断できない	複数ジェスチャーの組み合わせに対応

第 4 章 研究成果

4.1 座標変換

三つの座標系を、Color 座標系もしくは Depth 座標系に統一して出力する方法が一般的である。使用する PC で動作が軽快であったことと、目的とするシステムに高フレームレートは必要ではないと判断したため Depth 座標系を選択した。

Color 座標系および Camera 座標系から Depth 座標系への変換は、次のような流れで実装した。まず Kinect を開き、座標変換のため CoordinateMapper を取得した（26、27 行目）。Color、Depth、Body、BodyIndex のバッファを作成すると共に MultiSourceFrameReader を取得した（28 行目）。大量のデータを扱うシステムを開発するときには、個別のリーダーを開かなくてよくなり、MultiSourceFrameReader の中で必要なストリームを宣言することで記述が簡単になる。

```
26 KinectSensor kinect;
27 CoordinateMapper mapper;
28 MultiSourceFrameReader multiReader;
```

リーダーを用意した後には、MapDepthFrameToColorSpace メソッドを利用した。1 ピクセルあたりのバイト数と解像度の大きさを引数として与えることにより、Depth データのインデックスが Color 座標系のどの座標に対応しているかを取得することができる（図 5）。

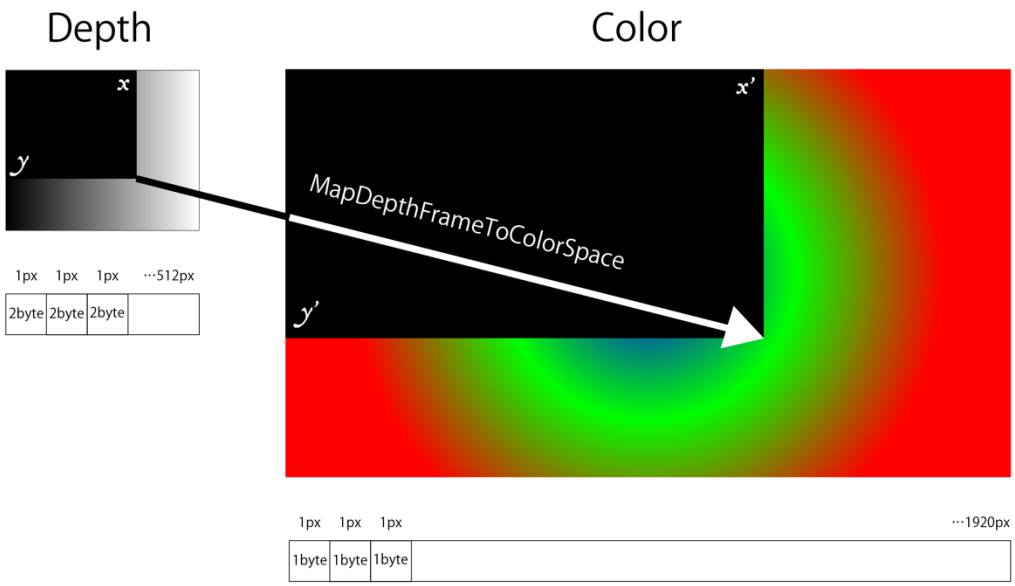

図 5　Depth 座標系と Color 座標系対応関係

```
298  var colorSpace = new
299  ColorSpacePoint[depthFrameDesc.LengthInPixels];
     mapper.MapDepthFrameToColorSpace(depthBuffer, colorSpace);
```

あとは座標位置が統合された各ストリームデータのバッファを、条件に応じて画面表示させる。これにより元から座標系を共有する BodyIndex と Infrared 以外の Color 座標系を Depth 座標系に変換することができた。

4.2 人物検出の判定と表示

Depth Frame が取得できたらデータを取り出し、画像として可視化する。取り出した Depth データは 16bit（0〜8000）で 1 画素を構成している。このままでは画像として表示できないので、これを 8bit（0〜255）の範囲に収めるように変換する。

取り出した Depth データを 8bit の範囲に収めるように変換したのち、データを返す。8bit のためデータは 255 のレベルに分かれている。センサーから対象物までの距離が近いほど白く（255)、遠いほど黒く（0）表示される。ここで、取り出した BodyIndex データより人物の位置を判定しその範囲以外の領域を出力することで、無限遠と人物の範囲以外はカラー画像で出力することに成功した。

```
301    Parallel.For(0, depthFrameDesc.LengthInPixels, i =>{
310    int bodyIndex = bodyIndexBuffer[i];
311    if (bodyIndex != 255)return;
319    });
```

4.3 危険行動種類の判定と表示

監視カメラを見つめる状態や、その場で徘徊する動きを危険行動と考えた。前者は離散的行動と分類し、AdaBoost アルゴリズムを用いて学習させた。後者は連続的行動と分類し、Random Forest アルゴリズムを用いて学習させた。

Kinect Studio を用いて想定される行動を xef ファイルに記録した。VGB を用いて教師信号を作成し、学習ファイルを作成した。

図 6 は「人物が立っているかどうか」を判定する gba ファイルを約 3 分間、約 180 フレームの動画から生成し、VGB 上で動作させた画面である。着席時と起立時でグラフの形が異なり「人物が立っている可能性が高い」と判定できていることがわかる。

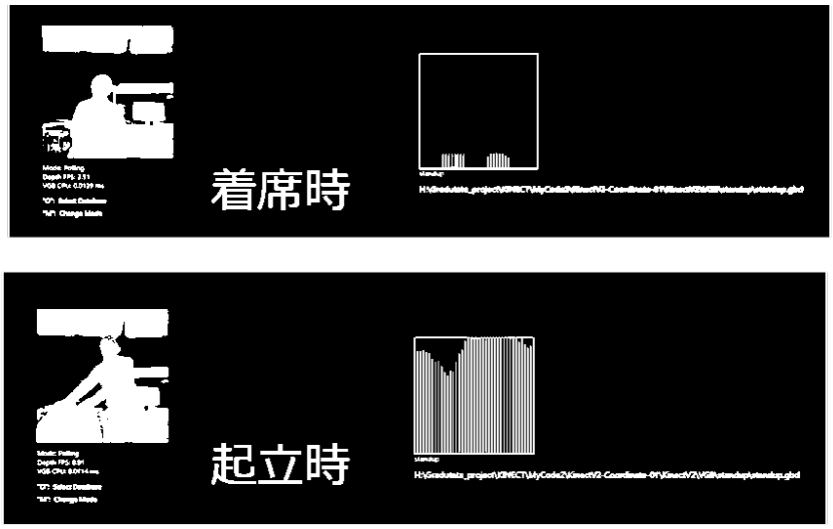

図 6　standup.gba 精度確認

ローカルサーバーに保存した学習したファイルをプログラムから読み取るためには二つの操作が必要である。

一つ目に、dll ファイルをダウンロードするため、ビルド後イベントのコマンドラインに次の命令を加えた。

```
  xcopy "$(KINECTSDK20_DIR)\Redist\VGB\x86" /e /y /i /r & if exist
"$(ProjectDir)\*.gbd" copy "$(ProjectDir)\*.gbd" /y
```

二つ目に、gbd ファイルをソースファイルに追加してコンパイルを行なった。gbd を「すべてのファイル」から追加し、出力ディレクトリを「常にコピーする」設定にする必要がある。

プログラムは一つの学習ファイルを読み込むだけで複数のジェスチャーを判定できる。gba ファイルには一つのジェスチャーの学習データのみ保存することができ、gbd ファイルには複数のジェスチャーの学習データを保存することができる。VGB によって作成された gba ファイルは、まとめて gbd ファイルを作成することができる。行動名と、二種の行動分類に対応したプロパティの値から危険行動を検出し判定した。

```
174    if (gestureFrame != null &&
       gestureFrame.DiscreteGestureResults != null&&
175    gestureFrame.DiscreteGestureResults[stare] != null)
176    {
177    var stareresult = gestureFrame.DiscreteGestureResults[stare];
178    if (stareresult.Detected ==true)
181    {
       TextBlock1.Text = "危険行動検出｜stare";
```

4.4　人数判定機能の実装

現在観測中の人数を判定する追加機能を実装した。クラス BodyFrameSource には、一度に取得できる人数の最大値を返す BodyCount プロパティが存在する。さらに、クラス Body には 6 人中何人目を追跡しているかを返す TrackingID プロパティと、追跡しているかどうかをブール値で返す IsTracked プロパティが存在する。

この三つのプロパティを組み合わせ、Kinect が取得できる人物の認識状態を足し合わせることで人数判定を可能にした。

```
71    BODY_COUNT = kinect.BodyFrameSource.BodyCount;
72    bodies = new Body[BODY_COUNT];{
277   for (int count = 0; count < BODY_COUNT; count++)
278   {
279   Body body = bodies[count];
280   bool tracked = body.IsTracked;
281   if (!tracked)
282       {
283     continue;
284     }
285   ulong trackingId = body.TrackingId;
288   BODY_CAPTURE++;
289    }
```

第 5 章 動作確認

実際にシステムを起動させると図 7 の表示となる。人物を検出していない、通常状態ではこの表示になる。左上に「人物非検出」というポップアップがあるのみだ。

図 7　動作画面（人物非検出時）

開発したシステムには大きく二つの機能が備わっていると述べた。

一つ目の機能、プライバシー保護機能の確認のため実験者が Kinect の視野に立ったところ、実験者はシルエットで表示された（図 8）。顔はもちろん、身体的特徴が判別できない状態となっていることがわかる。左上のポップアップは「1 人検出」と示され、人数判定機能が正しく動作していることがわかる。

図 8　動作画面（シルエット表示）

次に、防犯機能確認のため、学習させた動作を実験者が行った。

離散的行動である監視カメラを直視する動作を行った。結果、実験者はカラー画像で表示された（**図 9**）。「1 人検出」の下に、「危険行動検出 | stare」と表示されていることがわかる。危険行動を複数学習させた場合を想定して、判別時にどの行動であるかわかるようにした。

さらに、Xaml のコントロール Visibility を操作することで、危険行動検出時のみ表示を行い、シンプルなインターフェイスを構築した。

13(.xaml)	<TextBlock Name="TextBlock1" Foreground="White" FontSize="15" Text="初期化" Visibility="Hidden"/>

180(.css)	TextBlock1.Visibility = System.Windows.Visibility.Visible;

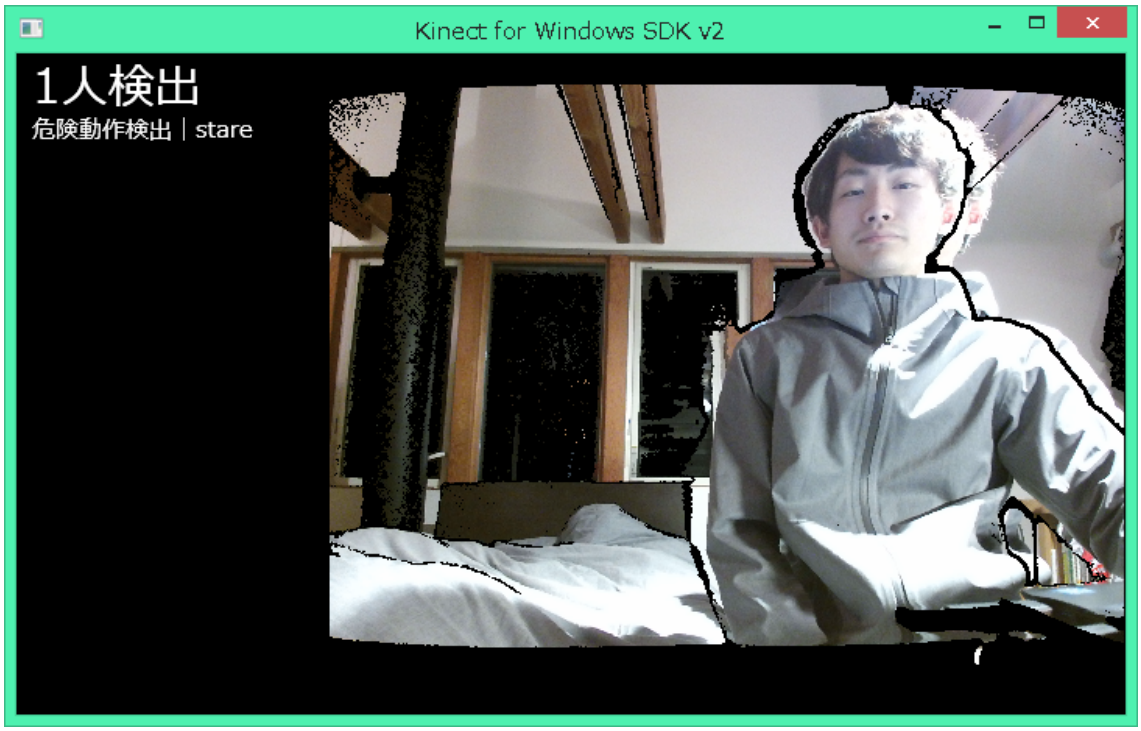

図 9　動作画面（離散的危険行動検出時）

連続的行動である、その場を徘徊する行動をとった。結果、実験者はカラー画像で表示された（**図 9**）。

離散的危険行動を検出した時には Xaml のコントロールの一つである ProgressBar の表示を行う。このブロックは、危険行動の確からしさに応じて動的に変化するため、直感的にわかりやすいという利点がある。

```
215     var progressResult =
216 gestureFrame.ContinuousGestureResults[wander];
        ProgressBar1.Value = progressResult.Progress;
```

図 10　動作画面（連続的危険行動検出時）

第 6 章 まとめ

本研究では、通常時には人物をシルエットとして表示し、危険行動を検知した時のみカラー画像で人物を映すことに成功した。さらに、危険行動を目的の動作を行うプログラムの作成ができた。

機械学習による不審者検出と Kinect を用いた不特定人物保護によって、利用者が負担も少なく安心して使える防犯システムが実現するのではないだろうか。

今後は危険行動検出時の録画機能や、外出の際でも見られるようなライブ配信機能など付加機能によって、さらなる実用性の向上を目指す。Kinect を用いた防犯システムが実際に社会で使われることが期待できる。

第 7 章 謝辞

本研究の遂行及び論文や概要、発表資料の作成にあたり、有益なご助言、ご指導を賜りました豊田工業高等専門学校電気･電子システム工学科大野亙准教授に深く感謝いたします。

第 8 章 参考文献

[1] 個人情報保護委員会(2017),個人情報の保護に関する法律についてのガイドライン

[2] 中村薫,杉浦司,高田智浩,上田智章(2015),KINECT for Windows SDK プログラミング, 秀和システム

[3] 「Summary?Blog Computer Vision, Image Processing, and Sensors」 http://unanancyowen.com/kinect-v2-coordinate-system-mapping/

[4] 「Kinect for Windows SDK でカメラ映像／深度情報／骨格情報の取得」 http://www.atmarkit.co.jp/ait/articles/1109/09/news140_2.html

[5] 「Kinect v2 × Visual Gesture Builder でファイトだよっ！」 http://sv.buu0528.com/d/c87vgb.pd