J-013

Azure Kinect DK を用いた CPR 訓練システムの開発検討
A Development of CPR Training System using Azure Kinect DK

栗谷川 知紀† 石黒 銀河† 皆月 昭則†
Tomoki KURIYAGAWA Ginga ISHIGURO Akinori MINADUKI

1. はじめに

毎年，7 万人以上が日常生活で心停止の危機に見舞われている．危機を目撃した時，「救命の連鎖」の開始点は，第一に心停止の早期認識と通報する市民の意思決定と行動であるが，市民が心停止者に対して実施する心肺蘇生法の正確な実技教育が確立していない．この課題へのアプローチとして，拙研究室は 2009 年から心肺蘇生法の実技を中核にした「救命の連鎖」教育支援システムを医療者・学会関係者・市民に公開しながら，機能の改良を積み重ねてきた．

本システムは，社会に普及し市民にとって身近に実技ができるようにするため，2020 年 4 月 2 日に日本で発売されたマイクロソフト社の 最新デバイスの Azure Kinect DK に換装するためのシステム要件を検討した．開発では，ホスト PC 要件の実験と CPR 姿勢変化を捉える実装をした．

2. CPR 訓練の NUI アプリケーション開発技術

CPR 訓練時の身体の姿勢変化の入力検知センサーデバイスとして先行研究の開発では 2 種を使用していた．先行研究のシステム構成の中核は，図 1 の写真に示すように，Kinect for Windows sensor である．先行研究で開発してきたシステム（以下，旧システム）では，2012 年に発売された Kinect for Windows v1 と 2014 年に発売された Kinect for Windows v2 のセンサーを使用してきた．

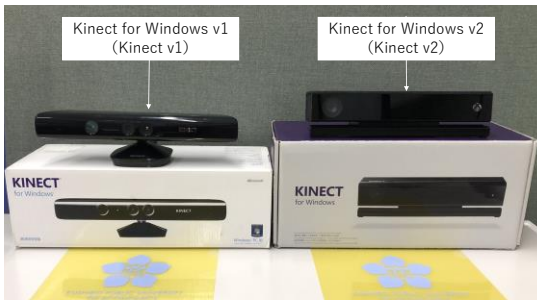


図 1　先行研究（旧システム）で用いた 2 種の KINECT

KINECT は，家庭用ゲーム機「Xbox 360」の入力センサーデバイスとして開発されて，図 1 に示すように Windows PC に接続する NUI（Natural User Interface）用入力センサーデバイスである．Kinect for Windows センサーの SDK が提供されていたため，先行研究では，CPR 訓練システムの NUI アプリケーション開発してきた．CPR の訓練時の姿勢の動作判定評価は，身体全体の入力検知のユーザーインターフェースが必要不可欠であり，キーボードやマウスによる入力デバイスによるシステム構成の訓練では要件を満たしていなかったことから，KINECT を使用して CPR 訓練時の腕などを検知する NUI アプリケーション開発してきた．

図 2 は旧システムによる市民への CPR 訓練の様子である．Kinect v1 から v2 の換装では，深度センサーの方式，カラー解像度／深度解像度，深度センサーの視野角，骨格検出数など仕様変更毎に開発実装の見直しを繰り返してきた．

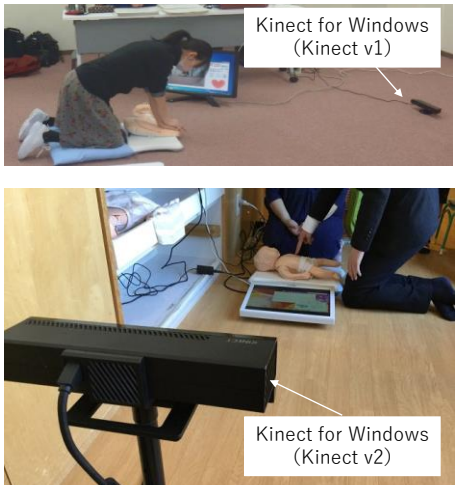


図 2 市民への CPR 訓練システムを用いた様子
（上段：Kinect v1・下段：Kinect v2）

2.1 最新デバイス Azure Kinect DK の特徴

家庭用ゲーム機から脱却した Azure Kinect DK（以下，Kinect v4）は，図 3 が示すように，従来の Kinect（Kinect v2: 外形寸法 249 x 66 x 67 mm）に比較して，外観形状はダウンサイジング（外形寸法 103 x 39 x 126 mm）されており，CPR 訓練システムのセットにおいて良好な要件になった．Kinect v4 は，小型化・低消費電力の仕様になっており，耐衝撃対策として外観形状に金属を用いているが，本体重量は 440g で，Kinect v2（970g）よりも大幅に軽量化されており，Kinect v1（599 g）よりも，さらに軽量である．

Kinect v4 は 2 種類のカメラが組み込まれている．深度（デプス；Depth）カメラは，マイクロソフト社が設計した 1 メガピクセルの ToF（Time-of-Flight）方式で，奥行き（距離）センサーを備えている．例えば Depth モードの視野角は，WFOV (広視野深度モード)で投射角 120 度のパルス光を照射し，画素ごとに反射時間を計測してフレーム全体の奥行きを集約して認識する仕組みである．

RGB（カラー）カメラは，OV12A10 12MP CMOS センサー のローリングシャッターセンサーが内蔵されており，カメラ解像度は 4K まで対応している．アスペクト比 16:9/4:3, MJPEG フォーマットの他，6 種類の解像度に対応している．

図 3 Kinect v4（手前）と Kinect for Windows v1 と v2（奥）

† 釧路公立大学, Kushiro Public University

2.2 Kinect v4 の SDK による開発環境と接続 PC

Kinect v4 の SDK（ソフトウェア開発キット）を使用する開発においては，PC（以下ホスト PC）のシステム要件が定められている．OS バージョンと PC の最小ハードウエア要件は次のように定められており，OS は Windows 10 (Version 1803, OS Build 17134) release (x64)以降で，ホスト PC は，Seventh Gen（第 7 世代） Intel® CoreTM i3 Processor (Dual Core 2.4 GHz with HD620 GPU or faster)，4 GB Memory，Dedicated USB3 port，Graphics driver support for OpenGL 4.4 or DirectX 11.0 が定めており，以上の要件を満たせないホスト PC では Kinect v4 の SDK による開発がマイクロソフト社のサイトに記載されていた．本研究の開発では，Kinect v4 の最初の接続ホスト PC に Dell 製 G3 15 3590（第 9 世代, CoreTM Core i7 Processor, 16 GB Memory , 6 Core 4.5GHz）を使用した．また，Kinect v4 の SDK の Body tracking のハードウエア要件の GPU（Graphics Processing Unit）も，NVIDIA GEFORCE GTX 1070 以上と定められており，Dell 製 G3 15 3590 の GPU は NVIDIA GeForce GTX 1650 である．

Kinect v4 の SDK を使用した開発では，Azure Kinect SDK (K4A)として，Depth camera access，RGB camera access and control や Motion sensor などが含まれる SDK を GitHub の NuGet サイトでパッケージをインストールする必要がある．本開発において NuGet サイトからダウンロードしたパッケージのバージョンを以下に示すと，

・Microsoft.Azure.Kinect.Sensor(v1.4.0)
・Microsoft.Azure.Kinect.BodyTracking(v1.0.1)
・OpenGL.Net(v0.8.4)
・OpenGL.Net.Core.UI(v0.8.4)
・System.Drawing.Common(v4.7.0)

である．

Kinect v4 の SDK を使用した開発では，Visual Studio 2019 バージョン 16.5 を使用した．開発の準備としてビルドメニューから構成マネージャーで 64 ビットのプラットフォーム（Any CPU で x64 新規作成を選択）に設定した．

Kinect v4 とホスト PC の接続については，旧システムケーブルの煩雑さを解消するために Kinect v4 付属の 2 本のケーブル（電源は電源供給ケーブル， データは USB Type-C to Type-A ケーブル）を使用せずに，1 本のパッシブケーブル（USB3.1 gen 1 Type-C）に集線した．マイクロソフト社の技術書では，最大 5.9 W，1.5 A 以上，長さ 1.5m をサポートするケーブルで接続可能という表記であったが，データ伝送が著しく不安定になる現象が発生した．結果，長さ 1mパッシブケーブルに短縮し，接続性問題は解消された．

図 4 接続性問題になった Type-C to Type-C ケーブル

2.3 CPR 動作を検知するための 2 種のカメラ特性

2.1 で述べた Kinect v4 に組み込まれている 2 種のカメラである RGB カメラと深度カメラが捉えた画像を図 5 に示す．

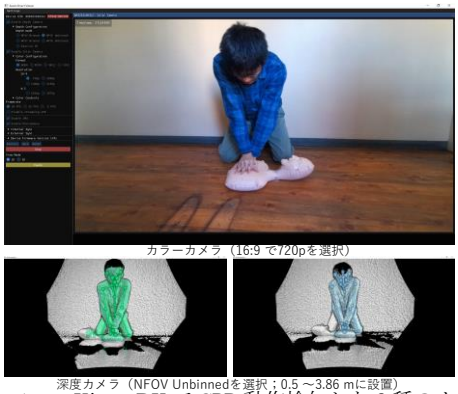


図 5　Azure Kinect DK で CPR 動作検知した 2 種のカメラ出力

Kinect v4 の 2 種のカメラの機能のおもな特性は，Color（RGB）と Depth（深度）の検知処理である．Color 処理では，カラー映像を撮影するカメラで，Depth 処理では，画素単位に深度（奥行き距離）を検知するカメラで，mm 単位整数まで深度取得が可能であり，ホスト PC に 3D Visualization 表示が可能である．

Kinect v4 の SDK によるホスト PC での Azure Kinect Viewer 機能の設定では，以下の 2 つの大別が可能である．

第 1 設定の Depth Configuration では， Depth mode の 5 種類 （NFOV Binned， NFOV Unbinned， WFOV Binned，WFOV Unbinned，Passsive IR）のモードを選択することが可能で，図 5 の深度モードは NFOV Unbinned，解像度 640 x 576，対象のフィールド 75° x 65°,動作範囲 0.5～3.86 m で CPR 動作検知の深度を出力している．

第 2 設定の Color Configuration では，4 種類（BGRA，MJPG，NV12，YUY2）のフォーマットが選択可能で，図 5 の設定は，カラーフォーマットは BGRA ピクセル形式を選択しており，他に MJPG（ MJPEG：Motion-JPEG のような画像の圧縮方式で JPEG を応用した動画データを生成する圧縮・記録方式）や NV12（データの容量を削減するための圧縮方式）の選択が可能である．Color Configuration 項目内の Resolution では，解像度が 16:9 で 4 種類（720p，1080p, 1440p，2160p），4:3 で 2 種類（1536p ， 3072p）が選択可能であり，図 5 の設定は 16:9 で 720p の出力である．また，Color Controls の Framerate では，単位時間あたりに処理させるフレーム（静止画像数）を 3 種類 （30FPS，15FPS，5FPS）フレームレートから選択することが可能で，図 5 の設定は 30FPS である．

他にも，Disable streaming LED，Enable IMU，Enable Microphone や View Mode で 2D や 3D 表示の選択が可能であり，使用しなければ適時オフにすることで，ホスト PC の処理の負荷の軽減が期待される．

2.4 胸骨圧迫時の姿勢推定技術の実装

最初の開発では，Visual Studio の C#2019 と Open.GL の Body Tracking 処理を実装した．

Body Tracking は，胸骨圧迫時の身体動作を追跡して，それぞれの身体の部位の変位を特定するセグメンテーションであり，固有の ID が決められており，24 箇所（正確には 25 箇所）の骨格情報を 3 次元座標で検知・取得している．

図 6 に示すように，最初の開発環境では，24 箇所の骨格情報のなかで，8 箇所の骨格変化を抽出しようと試みた．

2.2 節で述べたように Sensor SDK の開発では，ローレベルセンサーとデバイスアクセスのみであれば，ホスト PC のシステム要件は Core i3 7000 以上，Dual Core 2.4 GHz/HD620 GPU 以上，4GB メモリ , USB3 であり，ホスト PC のスペックが低くても開発が可能である．しかし，3D で身体を追跡する Body Tracking SDK の開発では，システム要件が高くなっており，Core i5 以上，NVIDIA GPU· GTX 1070 以上，CUDA10.1， cuDNN v7.5 のようなホスト PC のスペックが要求される．

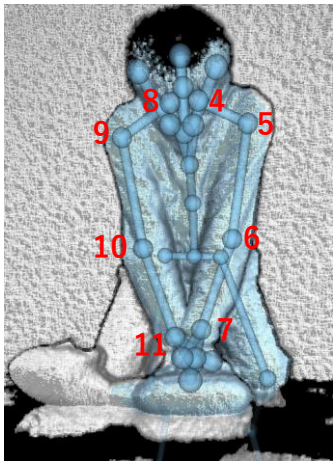


図 6　Body Tracking 時の CPR 姿勢の深度カメラの関節

図 7 に示すように 8 箇所の骨格の固有 ID（全 24 箇所）には，定義と親関節情報が付与されている．それぞれ，以下のように右腕（_RIGHT）と左腕（_LEFT）を示す．

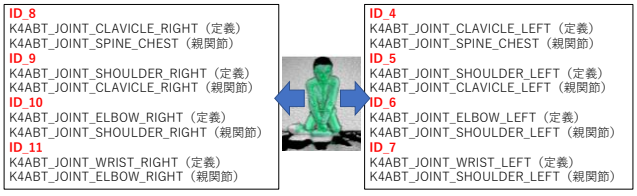


図 7　Body Tracking SDK における骨格の固有 ID 情報

深度カメラとプロジェクションに関する実験時には，強度，反射，回折，干渉が起きることを注意するようにした．マイクロソフト社の技術資料によれば，深度カメラでサポートされる動作モードでは，動作範囲は 850 nm で 15%～95% の反射率，2.2 μW/cm2/nm，確率的誤差標準偏差 ≤ 17 mm，標準的な系統誤差 < 11 mm + マルチパス干渉のない距離の 0.1%で動作範囲外の深度が提供されると記載されている．CPR 訓練の旧システム（Kinect v1 と Kinect v2）使用時は，衣服の反射率が影響したため，衣服の色を指定するなどしたが，Kinect v4 の実験時では問題が発生しなかった．

2.5 実験の結果

最初の開発の Body Tracking は，Kinect v4 の Azure Kinect SDK の C#ラッパーで実装したところ，Azure Kinect Viewer 機能の Color Controls 設定した Framerate 30FPS での表示が得られなかった．姿勢の変化を捉えて，なめらかな動きを決める要素がフレームレートであるが，カメラによって生成されるデータ量が大きいために，あらかじめ設定したフレームレート 30FPS に，ホスト PC の処理が追従できなった可能性がある．目視で，自然な動きに見えるフレームレート設定は 30FPS と言われており，CPR 訓練時の姿勢変化のリアルな動作確認のためには，映画～テレビの機器映像のように 24～30FPS 程度が満足されている必要がある．Kinect v4 は，旧システムの v2 よりも大容量のカラーデータ，深度データを検知しており，C#ラッパーのみの実装プログラムで解決する試みは失敗した．深度カメラで検知・生成されるデータは，BodyIndex（人検出において人物のいる領域を取得）と Body（骨格トラッキングして人体の姿勢を関節で取得）において，mm 単位整数とマイクロソフト社の技術書で公表されている．これらの膨大なデータをホスト PC 側の処理するための負荷が増大し，あらかじめ設定したフレームレートに影響した可能性がある．そこで，処理の負荷が増大する問題に対しては，ホスト PC のスペックを最上位機種にして，同じ Body Tracking 時の CPR 姿勢の実装プログラムで実験をしてみた． 最上位機種の PC の CPU と GPU スペックは，第 10 世代 CPU 搭載の MSI 製ゲーミングノートのステルス（GS66 Win10Pro, i9, RTX2070Super Max-Q, 15.6FHD 300Hz 16GB 1TB GS66-10SFS-022JP）である．結果は，フレームレートの影響は改善されておらず，Body Tracking 時の CPR 姿勢の表示は，なめらかな動きにはならなかった．

3. Unity を用いた胸骨圧迫時の姿勢推定技術の実装

Unity は，米国の Unity Technologies が提供するゲームエンジンであり，ゲーム開発では世界シェアナンバー1 で，最も使われているゲームエンジンである．本研究では Unity に，2.5 節で述べたホスト PC の処理負荷の軽減やフレームレートの改善につなげる活路を見いだした．

Unity は，IDE をともなったゲームエンジンであり，ゲームエンジン本体は C 言語/C++で書かれており，C#ラッパーでのプログラミング実装が可能である．今回，本研究で使用した Unity（Unity3D）2019 は，Unity2018 から提供された機能の拡張として C# Job System を備えており，マルチコアプロセッサーによる並列処理機能用にスクリプティングすることが可能である．結果，C# Job System によって，並列計算による競合やデッドロック負荷を回避しながら高い演算性能を利用する実装が可能になった．

ゲームエンジンだけあって，Unity2019 のデフォルト設定では，実行環境によってフレームレートが自動で調整されるようになっており，デフォルトの状態の Unity プロジェクトは，最速で実行しようという特性を備えており，フレームはディスプレイデバイスのリフレッシュレートの制約のもとであっても，可能な限り速くレンダリングされる．Unity2019 の機能強化における 3D テクスチャ対応，メモリとパフォーマンスの節約のためにジオメトリデータの改善，メッシュを効率的に非三角形ジオメトリのレンダーポイントとラインで構築する機能など生かした点を評価すると，Kinect v4 の Body Tracking 処理に向いていると期待した．

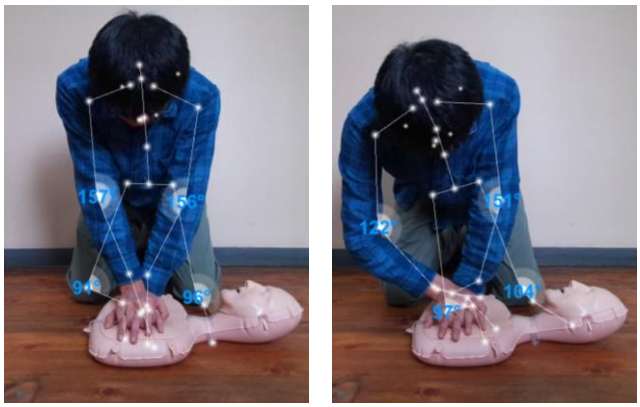
Unity2019 のゲームエンジンの機能によって，フレームレートの向上を改善すべく，Kinect v4 の Body Tracking する骨格に関するデータを単位時間あたりに処理させるフレーム数（frame/sec）を少なくして， Unity 側でフレーム補完するなどして，映画～テレビの機器映像のように 24～30FPS 程度の表示を満足させることを実装の目標にした．

3.1 Unity を用いた実験の結果

Kinect v4 の Azure Kinect Sensor SDK における C#Wrapper を Unity2019 で使用した．今回，開発実装に使用した unity のバージョンは unity2019.3.15f1 である．

Kinect v4 が想定している人体モデル（定義と親関節）が，Unity が想定している Humanoid モデルとの互換性がないため，Kinect v4 の Kinect BodyTracking API の Joint ID と Unity の HumanBodyBones を対応させてシリアライズして，戻り値を Unity の対応する関節名に変換することで，Kinect v4 の骨格と Unity が想定している骨格の定義を対応させた．これらを含めた UnityAPI の実装も行い実験の結果，フレームレートの問題は違和感なく改善され，肘，膝の関節角度の変化表示が，CPR 訓練で可視化できるようになった．

結果，旧システムでは，両腕が伸展位・屈曲位かの判定のみのアラート表示であったが，新システムでは，CPR 訓練時の 1 分タイマー機能を Unity で実装し，1 分間の CPR 動作について，左右両「肘」の角度の変化を検知し，胸骨圧迫時の姿勢が**左右にブレている**ことを**【非垂直圧迫】**として判定しており，また，左右両「膝」の角度の検知で，胸骨圧迫時の姿勢が**前傾**ぎみなのか，**後傾**ぎみなのかを，**【圧迫姿勢不安定】**として判定することを可能にした．



垂直圧迫　→→→→→→→ 【非**垂直圧迫**】
圧迫姿勢安定　→→→→→→→ **【圧迫姿勢不安定】**
図 8　左右両「肘」「膝」の角度の変化の検知の例

3.2 PC の処理遅延とアプリケーション配布の対処

Unity の導入によって，ホスト PC のスペックをフルに引き出すため，消費する電力が顕著であり，低スペック PC をホスト PC にする場合，電源ケーブルの常時接続を推奨する．電源ケーブルの非接続の実験では，ビルドしたアプリケーションは全画面表示になるので，顕著に遅延が発生した．こうした遅延対処は，電源ケーブルの常時接続で解決できる．対処の補足的な事項では，Kinect センサーで処理するアプリケーションを実装すると，ホスト PC 側でも消費電力を抑えようとすると遅延が発生するため，ホスト PC を電源に常時接続するか，非接続の場合であれば，ホスト PC 側の省電力モードをオフにしておくべきである．

アプリケーション化の準備については，unity からの起動ではなく，スタンドアローンのアプリケーション化をする場合，64bit で出力にするために x86-64 にしておく必要がある（デフォルトでは x86 の 32bit に設定されている）．アプリケーション化の手順は，unity でアプリケーション化したいシーンを開く→File→Build Settings→必要なシーンにチェックを入れる（シーンが何も表示されてないときは Add Open Scenes をクリック）→Architecture を x86-64 にする→Build→保存するフォルダを選択し，圧縮して配布する．

4. おわりに

4 月に発売された Azure Kinect DK を用いた CPR 訓練システムの開発検討というテーマに取り組んだ．Azure Kinect DK の CPR 訓練システム開発のため，実装に用いた 4 種のホスト PC および 2 種のパッシブケーブルは，実験によって明確な特徴が判明した．開発当初は，Azure Kinect DK に旧システムの機能を換装すればよいという安易な考えで，実装が完了すると想定していたが，結果的に最上位 PC での実装になり，開発は困難を極めた．開発において，メーカー（Microsoft 社）のスペック以上という要件は，Azure Kinect DK で何を処理するのかによって，ホスト PC の検討が必要である．Unity による Body Tracking の CPR 姿勢の可視化用のホスト PC の CPU スペックは，開発当初の PC の第 7 世代からフレームレートやレイテンシ改善に効果が見込めるクロックの高さが保証される第 10 世代インテル Core i9-10980HK の 8 コア 16 スレッドのマルチコアプロセッサーで最大 5.3GHz＆全コア時 4.9GHz（第 9 世代 CoreTM Core i7-9750　Processor 比で最大 150％の性能向上）で，GPU スペックは，ゲーミングノートに搭載されている NVIDIA GeForce RTX 2070 SUPER Max-Q（NVIDIA GeForce GTX 1650 比で最大 145％の性能向上）で，物理演算や動画エンコードなどコア数と CPU クロックが効く項目でスコアが公知されたベンチマーク最上位のプロセッサーである．

2.5 節の実験結果の対処方法として，ホスト PC への処理負荷を軽減するため，Color Configuration 項目内の Resolution で解像度 16:9 で 4 種類が選択できようになっており，最低の標準 HD 規格 720p にして，フレームレートに影響を与えない仮説で実験したが遅延が発生した．SensorSDK 機能として，メーカー（Microsoft）の技術資料には，内部的に同期をして遅延設定が可能とあり，そして，ホスト PC 側の同期制御で遅延オフセット設定も可能とあるが実装で試してない．試していない理由として，unity が有効に処理負荷の軽減やフレーム遅延の対処が可能であったため，そのままシステム実装した結果になった．技術資料を精査して，内部的に同期をして遅延設定が可能であるのであれば，再度，unity に依存しない実装も試みたい．

今後は，姿勢変化の記録（Video）機能を開発実装して，訓練指導用の振り返り機能として提案する．CPR の胸骨圧迫時の深さの検知の技術については，本論文の第 2 著者の石黒銀河氏の論文「最適な圧迫深度・姿勢の組み合わせを評価する CPR 訓練システムの開発」を参照いただきたい．

謝辞

本研究は，日本学術振興会の科学研究費助成事業において 2020 年度 基盤研究(B)に採択されたテーマ「救命の連鎖の開始点を担う市民が自ら考え実行できる CPR 実技の構成要素の抽出研究；研究課題番号 20H04291」の関連研究で，国民のみなさまはじめ関係各位に深謝いたします．

参考文献

[1] Azure Kinect DK の開発に関する技術情報，マイクロソフト https://docs.microsoft.com/ja-jp/azure/Kinect-dk/,2020 年 6 月リンク確認.
[2] Unity Learn Premium, https://unity.com/ja/products/learn-premium, 2020 年 6 月リンク確認.