MR空間内における猫じゃらしを用いたアミューズメント方式の研究

Ota, Riku / 太田, 琳久

(出版者 / Publisher)
法政大学大学院理工学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 理工学研究科編
(巻 / Volume)
64
(開始ページ / Start Page)
1
(終了ページ / End Page)
8
(発行年 / Year)
2023-03-24
(URL)
https://doi.org/10.15002/00026388

MR 空間内における猫じゃらしを用いたアミューズメント方式の研究

Research on amusement method using catnip in MR space

太田 琳久
Riku Ota
指導教員　金井 敦

法政大学大学院理工学研究科応用情報工学専攻修士課程

In recent years, technologies that utilize the interaction between real and virtual space, such as AR and MR, have become a hot topic. This paper proposes a method to calculate the depth distance of a real object from a user for the purpose of communicating with a virtual character using a real object to reduce the technical distance between the virtual space and the real space.It has been difficult to obtain depth coordinates of small objects from MRHMD. In this paper, we propose a method to calculate the depth coordinates of small objects in MR space, such as the tip of a catnip, and verify the method with an amusement system. The proposed method was compared with the actual measurements and resulted in an error of about 0.001 m. The proposed method succeeded in calculating almost accurate depth coordinates.
***Key Words**:MR,Hololens2,objectdetecting*

1. 序論

近年、AR、MR 技術といった仮想空間を利用した技術がさまざまな用途で利用され始めている。エンターテインメント業界においては、ユーザーと仮想キャラクターとの対話を可能にするようなスマートフォン向けの AR ゲームアプリケーションが話題になっている。内容としては実際の地図上にキャラクターが出現し、その場所に向かうことでキャラクターとアプリ上で遭遇することができ、キャラクターを捕まえたり、道具を使って遊ぶことができる。仮想的なキャラクターとの対話を AR 技術を用いることで可能にすることで現実空間と仮想空間との技術的距離を縮めることが可能となる。そこで実際の物体を用いた仮想キャラクターとの対話が可能であれば、より現実空間と仮想空間との技術的距離を縮めることが可能なのではないかと考える。本稿では MR 技術を利用し、実際の猫じゃらしを用いて仮想の猫を戯れさせるようなアミューズメントシステムを実装する。

仮想の猫が実際の猫じゃらしに飛びつくように描写するためには実際の猫じゃらしの MR 空間内における三次元位置座標の算出が必要不可欠である。現行研究として、物体検出モデルを作成することで、対象の物体の映像上の座標を容易に抽出可能であるが、MRHMD のみを用いた物体の奥行き座標の算出は困難である。本稿では実際の猫じゃらしに対して少ない計算量で正確に奥行き座標を算出する手法を提案する。また、提案手法を元に実装したアプリケーションの性能評価を行う。

2. 関連研究

本章では本稿の関連研究として映像から現実空間内の実物体を認識する手法に関する現行研究を複数紹介する。

2.1 モデルベースの手法

対象物体の 3D モデルを構築し、それをもとに 3 次元位置推定を行う手法も存在する。

Marchand らはカメラ映像から識別対象の物体の位置姿勢をロバストに追跡する手法を提案している [2]。識別対象物体から生成した粗い 3D モデルを用いることで、物体の不可視の部分の認識が可能となり、物体の姿勢推定をロバストに行えるようになるため、複雑な環境下におけるトラッキングが可能となる。しかし、植物などの個体差のあるような物体を対象としてトラッキングを行った場合、各個体ごとに 3D モデルを構築する必要があるため、非効率的である。

Torre らは仮想人型モデルを AR のように現実空間上に配置し、仮想人型モデルとのチェスの対戦が可能となる

手法を提案している [4]。この手法では、カメラの視点移動に対してロバストに実際のチェスボードの位置姿勢のトラッキングを行う。まず、Marchand らの手法 [2] を用いることで、幾何学的制約に従う 3 次元の頂点の集合からなるチェスボードの 3D モデルを生成する。さらに、ロバストなトラッキングを行うため線形運動予測を行う一次関数により次のフレームにおける位置座標を予測する。そして 3D チェスボードモデルに対するカメラの位置姿勢のパラメータに対する非線形エネルギー関数を反復処理によって最小化することにより、最適な位置姿勢を計算する。しかしこの手法の場合、1 フレームごとの位置座標を計算するための計算量が多くなってしまう。

2.2 マーカーを用いた手法

マーカを用いて物体の位置座標を算出する手法については多数提案されている。

今本らは ARToolKit マーカといったような ID 付きマーカと ID が付いていない点マーカを併用することで高い精度で実際の物体の位置座標を算出する手法を提案した [5]。汐崎らはテンプレートマッチングによって追跡するテクスチャ特徴と MRHMD カメラ映像に配置された物体情報の ID 付きのマーカにより対象物体を追跡し、位置座標を算出を行う手法を提案した [6]。また、ID 付きマーカを用いない手法も存在する。Neumann らは色と形の異なるマーカを物体上に 3 点配置することでマーカの配置パターンから様々な物体の検出を容易にする手法を提案した [7]。マーカのバリエーションを増やすことに着目し、Thomas らはバーコードを同心円上に配置することで多数の物体の判別を可能にした [8]。マーカを用いる手法の場合、物体の付近でのマーカの設置が必要不可欠である。そのため、識別対象物体を増やせば増やすほどマーカの数が増え、現実空間上の景観を損ねてしまう可能性が考えられる。また、識別対象物体それぞれにマーカーを設置する必要があるため、状況によっては実用化が困難であるとも考えられる。例えば、道路標識を認識するようなシステムにこの手法を適用する場合に、マーカによる道路標識の識別は可能であったとしても、道路標識すべてにマーカを設置することは困難であると考えられる。

2.3 その他の手法

Microsoft 社は自社で開発している Hololens2[9] と呼ばれるシースルー型の MRHMD を用いて物体の三次元位置座標を算出する手法として、Hololens2 から出される Ray と呼ばれる光線を用いた手法を提案している [10]。Hololens2 では Ray の照射先の対象の奥行き座標の算出が可能ある。Microsoft 社は画像認識システムを用いて Hololens2 のカメラ画像から二次元座標を算出し、Ray を物体に照射させることで奥行き座標を算出する。しかしこの手法は検出対象となる物体が小さい場合や、動いている場合に Ray が対象物体から僅かにずれてしまい、正確な奥行き座標の算出が困難になる可能性がある。

また、Wikitude 社は Wikitude 社独自の物体追跡、イメージトラッキング技術を Apple 社開発の AR フレームワークである ARKit や Hololens2 の位置追跡機能と統合させることで、ラジコンカーの位置、速度に合わせてラジコンカーからエフェクトを表示させるシステムの開発を行う [11]。対象物体の形状のデータやトラッキング情報を含む wto ファイルを生成し、映像から検出された物体の形状や大きさから wto ファイルを基に物体のおおよその位置姿勢を計算することで追跡を行う。しかし、形状に個体差があるような物体を対象物体とした場合、個体ごとに wto ファイルを作成する必要があるために、非効率的である。また、動物や軟体など環境によって形状を変化させるような物体を対象とした場合には有効的な手法ではないと考えられる。

3. 提案手法

3.1 CustomVision を用いた物体検出モデルの作成

本手法では Azure Custom Vision という Microsoft 社開発の Web 画像認識システム [12] を用いて物体検出を行う。CustomVision では学習データとする画像を Web 上にアップロードし、図 1 で示すように視覚的特性に従った独自の画像識別子に従い画像上に含まれる形状をもつ物体に対しバウンディングボックスが複数設けられる。そして図 2 で示すようにユーザーは与えられたバウンディングボックスの中から識別対象を囲うものにタグ付けを行い、独自のアルゴリズムでトレーニングを行うことで物体検出モデルを生成する。本稿では検出モデルの構築にあたって種類の異なる猫じゃらしの画像をあらかじめ複数枚用意し、各画像において CustomVision 上で設けられたバウンディングボックスの中から識別対象物体である猫じゃらしの先端部分を含むものを指定しタグ付けを行い、学習データとして猫じゃらし検出モデルを生成する。

アプリケーション実行後、Hololens2 に搭載されているカメラによってユーザー視点の画像を撮影する。そして撮影された画像をバイト配列に変換し、スクリプト上から Custom Vision の Web サーバーへ送信することで Web サーバ上で生成した物体検出モデルを用いた物体検出が行われる。その後、Web サーバーから物体の識別結果として先述した図 1 で示したようなバウンディングボックスの二次元位置座標とその予測値を含む応答を受け取る。そして予測値の値が最も大きい要素を抽出し、予測値があらかじめ閾値として定めた数値よりも高い値であれば識別対象物体であると判定する。

3.2 手の座標を用いた奥行き座標算出手法

本章以降では MR 空間上における空間位置座標系を x 軸を横軸、y 軸を縦軸、z 軸を奥行き座標軸とする。本手

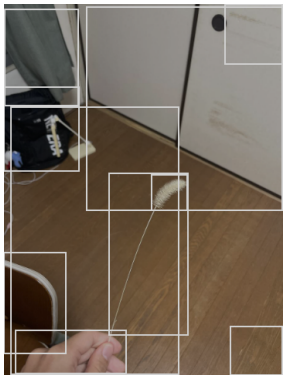

図 1: 設置されたバウンディングボックス

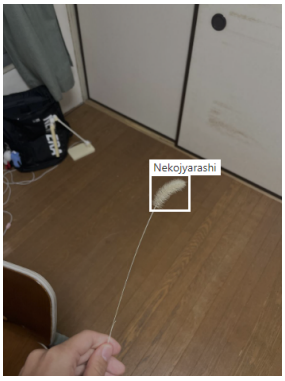

図 2: 識別対象を囲うものにタグ付け

法では図 3 で示すような手順で猫じゃらしの奥行き座標の算出を行う。

図 4 で示される奥行き距離の算出に必要な変数の算出方法について述べる。まず、

L : 猫じゃらしの長さ

W: 猫じゃらしの先端と猫じゃらしを持っている手の x 座標値の差分

H: 猫じゃらしの先端と猫じゃらしを持っている手の y 座標値の差分

の 3 つの変数からユーザーの手の空間位置座標から猫じゃらしの先端の空間位置座標までの奥行き距離 $Depth$ を、直方体の対角線の導出式を変形することにより得られた (1) 式により算出する。

$$Depth = \sqrt{W^2 + H^2 - L^2} \tag{1}$$

Hololens2 では生成される MR 空間における座標系の原点が Hololens2 自体である。そのため、猫じゃらしの空間位置座標における x 座標、y 座標は CustomVision より得られたバウンディングボックスの位置座標を用いるが、z 座標については猫じゃらしの持ち手の親指の先端の位置座標における z 座標の値に $Depth$ を加えたものを用いる。

3.2.1 第一変数の算出

まず、第一変数の L の算出方法について述べる。Hololens2 ではハンドトラッキング機能によりユーザーの手の位置姿勢や手の関節位置の検出が可能であり、ユーザー自身の手で直感的な操作が可能であり [9]、タップジェスチャーと呼ばれるタッチするような動作の検出が可能である。猫じゃらしを地面に対して水平になるように手で持ち、もう一方の手で猫じゃらしの先端に対してタップジェスチャーを行う。その際、タップジェスチャーを行なった手の人差し指の先端の空間位置座標と猫じゃらしを持っている方の手の親指の先端の空間位置座標を算出

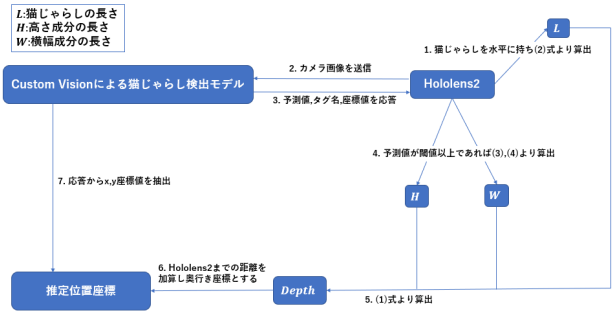

図 3: 奥行き座標算出手法

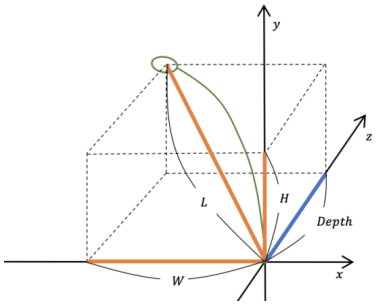

図 4: 奥行き座標の算出に用いられる長さ

し、2 本の指の空間位置座標の二点間距離を (2) 式より算出する。

$$L = \sqrt{(Index\_x - Thumb\_x)^2 + (Index\_y - Thumb\_y)^2} \tag{2}$$

ここで $Index\_x$、$Index\_y$ はそれぞれタップジェスチャを行なった手の人差し指の x 座標、y 座標であり、$Thumb\_x$、$Thumb\_y$ はそれぞれ猫じゃらしを持っている手の親指の x 座標、y 座標である。

3.2.2 第二変数・第三変数の算出

次に、第二、第三変数の W、H の算出方法について述べる。まず猫じゃらしの長さの算出後、ユーザーの持ち手を元の持ち手に戻す。その後、Hololens2 のカメラ画像を撮影し、撮影した画像のバイト配列を CustomVision の Web サーバに送信する。応答として得られた要素内で予測値が最も大きい要素において、予測値が閾値よりも大きかった場合、その要素が示すバウンディングボックスが猫じゃらしの先端部分を囲っていると識別し、対象のバウンディングボックスの位置座標を取得する。その後、Hololens2 のハンドトラッキング機能によりユーザーが猫じゃらしを持っている手の親指の先端部分の二次元位置座標を取得する。最後に、持ち手の親指の先端の位置座標と CustomVision から得られた猫じゃらしのバウンディングボックスの位置座標の二点間距離を (3)、(4) 式より算出する。

$$W = Box\_x - Thumb\_x \tag{3}$$

$$H = Box\_y - Thumb\_y \tag{4}$$

ここで *Box_x*、*Box_y* はそれぞれバウンディングボックスの *x* 座標、*y* 座標である。もし、応答として得られた予測値の最大値が閾値よりも低かった場合、画像上に猫じゃらしが存在しないと認識し、座標計算は行われない。

3.3 猫じゃらしの停止状態識別手法

本稿では、猫じゃらしが動きを止めた瞬間に VirtualCat が猫じゃらしに飛びつくような動作を実装する。そこで本章では猫じゃらしが動いているか止まっているかを識別する手法について述べる。

本稿では第二、第三変数の算出を反復して行うことにより猫じゃらしの三次元位置座標を繰り返し取得する。まず、猫じゃらしの画像を取得し三次元位置座標の算出後、その座標値を記録する。記録後、もう一度同じ手順を繰り返し、そのフレームにおける三次元座標値を算出する。そこで、算出された座標値と前フレームにおける座標値の二点間の距離を算出し、2 つのフレーム間での移動距離を導出する。そこでの移動距離があらかじめ指定された距離閾値よりも短くなれば、2 つのフレーム内で動きがなかったと考え、猫じゃらしは” 停止” 状態であると認識する。反対に、移動距離が距離閾値を上回った場合、猫じゃらしは” 活動” 状態であると認識する。

3.4 3D モデルのアニメーション動作

VirtualCat を実際の猫のように動作させるために、本手法では 3 種類のアニメーションを用意する。一つ目は Stay モーションと呼ばれる待機姿勢である。このアニメーションは主にユーザーが猫じゃらしが” 活動” 状態である場合と猫じゃらしが認識できない場合に実行される。二つ目は Walk モーションと呼ばれる歩行姿勢である。猫じゃらしが” 停止” 状態になったとき、VirtualCat は猫じゃらしの推定位置座標に向けて移動する。その際、猫じゃらしに向かって移動する間に Stay モーションから Walk モーションに推移させることで、歩いているかのように見せる。最後に三つめは Catch モーションと呼ばれる捕獲姿勢である。猫じゃらしが” 停止” 状態となったとき、VirtualCat は Walk モーションを実行して移動し、目的となる猫じゃらしの推定位置座標に到達した際に Catch モーションへ推移させる。

4. 実装

本章では先述した手法の実装方法について述べる。

今回の実装において、Hololens2 が認識するユーザーの指の位置座標を MixedRealityToolKit(MRTK)[13] というコンポーネントを用いて取得する。MRTK は MR アプリケーションの開発に役立つように Microsoft 社によって設計されたプラグイン、サンプル、ドキュメントの形式のコンポーネントであり、25 個のユーザーの手の関節座標の取得が可能である。

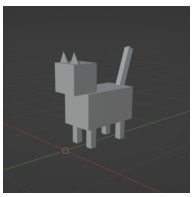

図 5: Stay

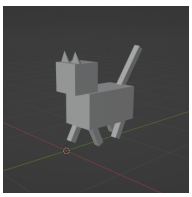

図 6: Walk

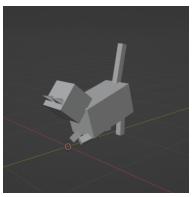

図 7: Catch

開発プラットフォームには Unity を用いる。Unity にて 3D モデルや UI の設置、作成したスクリプトと 3D モデルの紐づけ、VirtualCat のアニメーションの実行条件などの設計を行う。そして作成したアプリケーションのソリューションファイルのビルドも Unity にて行う。また、スクリプトの作成やソリューションファイルの実行は VisualStudio にて行う。

VirtualCat の 3D モデルとそのアニメーションは Blender を用いて作成する。構成されたモデルにボーンを設定し、各部位モデルを回転させることで VirtualCat のアニメーションを作成する。作成した Stay モーションを図 5 に、Walk モーションを図 6 に、Catch モーションを図 7 に示す。Stay モーションではしっぽの部位のみを左右に動かすことでユーザーの指示を待つ姿勢を表現している。Walk モーションでは足を前後に動かし、しっぽを左右に動かすことで歩く姿勢を表現している。Catch モーションは足元にある猫じゃらしをつかむような姿勢を表現している。

5. 検証

5.1 猫じゃらし検出モデルの作成と認識精度の検証

5.1.1 猫じゃらし検出モデルの作成

本手法では先述したように CustomVision を用いて猫じゃらしの検出モデルを生成する。猫じゃらしの個体差に対応させるため、20 本の異なる個体の猫じゃらしの画像を 830 枚用意し、タグ付けを行い、学習データとして学習を行う。また、CustomVision では物体検出モデルを生成する際に学習に用いる時間が指定可能であり、本稿では 15 時間の学習により検出モデルを生成する。CustomVision によって生成されたモデルは適合率、再現率、mAP の 3 つの評価指標によって認識精度が評価される。

5.1.2 認識精度の検証

物体検出モデルの誤認識を防ぐためには適切な予測値の閾値を設定する必要がある。まず、一般家庭での使用を考えたリビングでの環境、暗い環境、野外環境の 3 種類の異なる背景でモデルを用いた物体検出を行い、推定に用いられた画像における各バウンディングボックスの予測値とその位置を CustomVision の Web サーバ上から得ることで、適切な閾値の値を吟味する。

アプリケーションの利便性を高めるためには、ユーザーが用いる猫じゃらしの形状に関わらず位置座標の推定が

図 8: 形状的特徴が失われる場合

図 9: 毛が抜けている個体

図 10: 先端の長さが短い個体

図 11: 先端の長さが長い個体

可能である必要がある。猫じゃらしの先端の形状はその個体差によって様々存在する。また、猫じゃらしの持ち方などによりその位置姿勢の変化で認識が困難になる可能性も考えられる。例えば、図 8 で示すよう猫じゃらしの先端部分を下から見るような姿勢の場合、猫じゃらしの形状の特徴が失われてしまう。そこで、本稿では猫じゃらしの個体差とその位置姿勢の変化に伴う位置座標推定への影響についても検証を行う。

まず、図 9、10、11 で示すような猫じゃらしの先端部分の毛が抜けている個体と先端部分の長さが短い個体と長い個体の三種類の猫じゃらしを用意する。そして検出対象の猫じゃらしを 45 度ずつ回転させ、1 周期分の予測値を記録することで、姿勢変化に伴う予測値の変化を記録する。

5.2 推定位置座標の算出

推測された猫じゃらしの位置座標と実際の猫じゃらしの位置との誤差を算出し、その結果から本手法の有効性を吟味する。奥行き座標の算出結果を 100 回分抽出できるまで繰り返し、Hololens2 から猫じゃらしの持ち手までと持ち手から猫じゃらしまでの水平成分の距離と奥行き距離のそれぞれの数値について平均値をとる。最後に算出した平均値と実測値のそれぞれの値を比較する。ここで、算出した奥行き距離の値は持ち手から猫じゃらしの先端までの水平距離に Hololens2 からを持ち手の親指の先端までの距離を加えた値と比較するものとする。

5.3 アプリケーションの動作と検証

最後に、実際の猫じゃらしに VirtualCat が飛びつくようなアプリケーションの実装を行う。算出された三次元位置座標とひとつ前に記録した位置座標の二点間距離を距離閾値と比較することにより猫じゃらしが” 活動” 状態であるか、” 停止” 状態であるかを判断する。本稿では距離閾値を 0.1m として実験を行う。また、猫じゃらしが” 停止” 状態であるとき、VirtualCat は推定位置座標に向けて移動し、到達後に Catch モーションを行う。そのため推定位置座標が実際の猫じゃらしの位置から大きく外れてしまえば、VirtualCat と実際の猫じゃらしの間に距離ができてしまう。そして、VirtualCat と猫じゃらしの位置関係などによっては、VirtualCat が Catch モーションを行った際に猫じゃらしを捕らえていないかのように見えてしまう。そのため、Catch モーションを行った際の VirtualCat 位置姿勢と実際の猫じゃらしの位置姿勢を撮影し、実際の猫じゃらしをつかんでいるように見えるかを検証する。

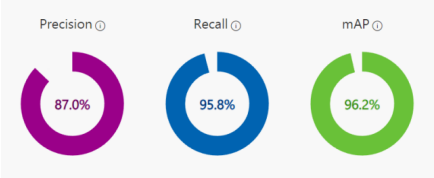

図 12: 生成した猫じゃらし検出モデルの分類精度

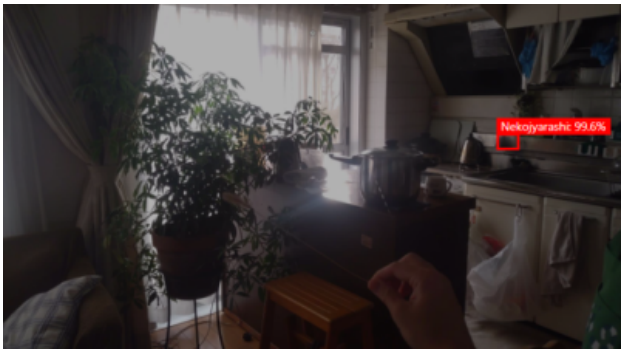

図 13: リビングの画像における予測値算出結果

6. 結果

6.1 検出モデルの学習結果

CustomVision による猫じゃらし検出モデルを生成した結果を図 12 に示す。適合率は 9 割を下回るような結果となってしまったが、すべての評価指標は 8 割を超え、再起率、mAP は 9.5 割を上回るような結果となった。

6.2 検出モデルの認識精度の検証結果

6.2.1 背景変化に伴う認識精度

5.1.2 節で述べた 4 種類の異なる背景において猫じゃらしを対象とした物体検出を行った結果を示す。まず、リビングにて物体検出を行った結果をを図 13 に示し、暗い環境において物体検出を行った結果を図 14 に示し、野外環境にて物体検出を行った結果を図 15 に示す。ただし、ここでは予測値が 8 割を超えたバウンディングボックスのみ検出結果として表示させている。

リビングにおいて物体検出を行なった結果、図 13 で示すように猫じゃらし自体にはバウンディングボックスが配置されず、キッチンの一部分に配置されたバウンディングボックスの予測値が 99.6 と高い値で算出されてしまう結果となってしまった。そして予測値が 8 割を超えるようなバウンディングボックスはこのバウンディングボッ

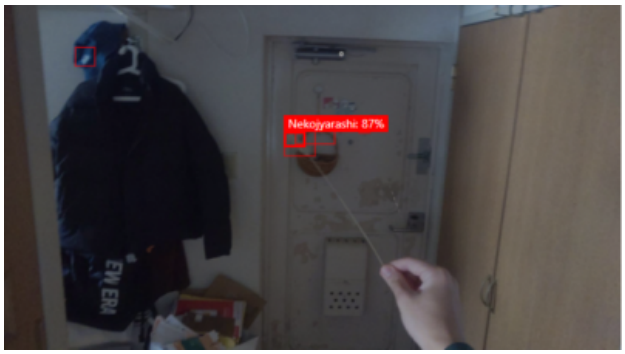

図 14: 暗い環境の画像における予測値算出結果

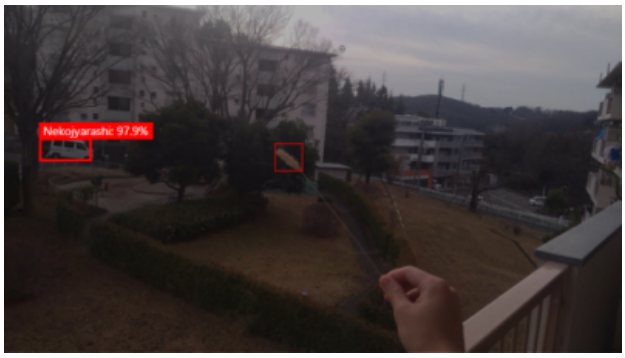

図 15: 野外環境の画像における予測値算出結果

クスのみであった。また、暗い環境においては、画像の左側に位置している帽子のロゴ付近にバウンディングボックスが配置されてしまっているが、猫じゃらしの先端部分にバウンディングボックスが設置され、予測値は 87.0 と算出された。そして、野外環境においては図 15 で示すように猫じゃらしの先端部分にバウンディングボックスが配置されたが、奥の建物の前に駐車されている白の車両を囲うバウンディングボックスの予測値は 97.9 と算出されてしまった。

6.2.2　個体差に伴う認識精度

5.1.2 節で述べた方法で猫じゃらしの 3 種類の個体に対して予測値を算出した。各事象において猫じゃらしの先端部分を囲うバウンディングボックスの予測値を記録した結果を表 1 に示す。

表 1 で示すように、全体を通して 95%を超える高い予測値となった。また、各個体の中でも猫じゃらしの先端部分が長い個体は、どの角度においても 99.9%と高い識別結果が算出された。それに対し、毛が抜けた個体を回転させずに認識を行った結果は 96.3%と少々低い値になってしまった。

表 1: 各個体の角度とその予測値

回転角	0°	45°	90°	135°	180°	225°	270°	315°
毛が抜けた個体 (%)	96.3	99.7	99.5	99.5	99.7	99.5	99.9	99.5
短い個体 (%)	99.4	99.8	99.9	99.9	99.9	99.7	99.8	99.7
長い個体 (%)	99.9	99.9	99.9	99.9	99.9	99.9	99.9	99.9

表 2: 計測値の平均値と実測値

	計測値	実測値	誤差
Hololens2-持ち手 (m)	0.597	0.608	-0.011
持ち手-猫じゃらし (m)	0.129	0.130	-0.001
奥行き距離 (m)	0.727	0.738	-0.011

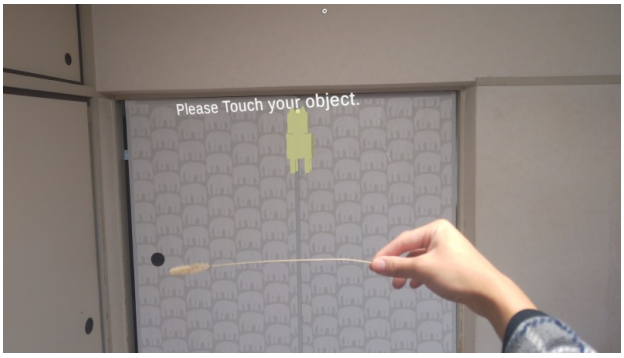

図 16: タップジェスチャー待機姿勢

6.3　位置座標の算出と誤差

5.2 節で述べた手法で 100 回分の奥行き距離の算出し、各値の平均値と実測値を表 2 に示す。ただし、表 2 における”Hololens2-持ち手”は Hololens2 からユーザーの持ち手までの水平距離であり、”持ち手-猫じゃらし”は持ち手から猫じゃらしの先端部分までの水平距離であり、奥行き距離はそれらの値を合計することにより算出したものである。ただし、予測値における奥行き距離の値は計測した値の平均値であり、誤差は予測値から実測値を差し引いた値である。この実験において物体検出をするにあたり各バウンディングボックスの予測値の閾値を 96 とする。そして画像の撮影から三次元位置座標の算出までの一回の処理にかかる時間は 4.37 秒であった。

Hololens2 から持ち手までの距離が実測値より約 1cm 程度短い結果となってしまったが、今回算出した持ち手から猫じゃらしの先端までの距離の誤差は約 1mm 程度となり、高い精度で距離算出ができている。

6.4　アプリケーションの動作と検証

アプリケーション実行後、猫じゃらしの長さの入力を待機している画面を図 16 に示す。猫じゃらしの先端部分にタップジェスチャーを行い、3.2.1 節で示した手法により猫じゃらしの長さを算出する。算出後、3.3 節で述べたようなキャプチャループにより猫じゃらしの停止状態を取得する。猫じゃらしが停止状態である時、算出した猫じゃらしの予測位置座標に VirtualCat が飛びついた結果を正面から見た結果を図 17 に示し、真横から見た結果を図 18 に示す。

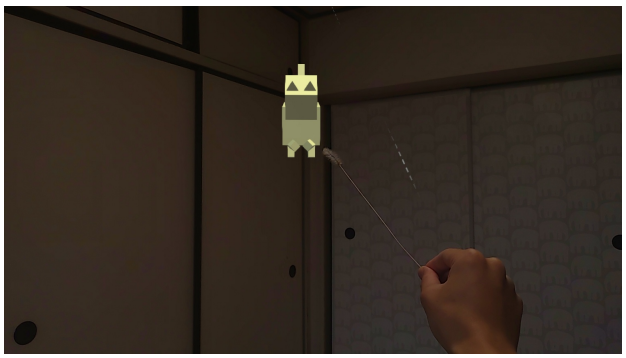

図 17: 猫じゃらしに飛びついている様子 (正面)

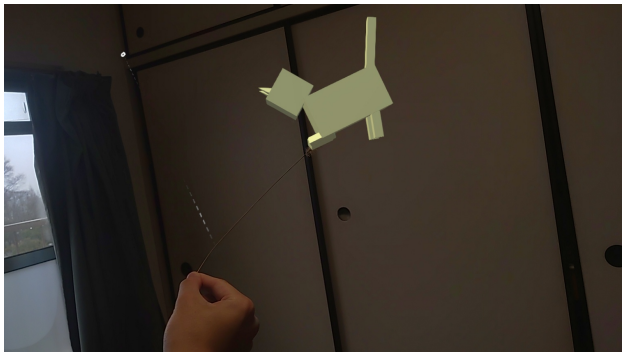

図 18: 猫じゃらしに飛びついている様子 (真横)

7. 考察

7.1 学習モデルの認識精度

図 12 で示したように適合率が 87.5%と他の評価指標よりもより低い値となってしまっている。そのため、野外環境において図 15 で示したように白い車両や、リビングの画像における猫じゃらしではない部分を高い予測値で猫じゃらしとしてしまったのだと考えられる。

また、図 13 については猫じゃらしにバウンディングボックスも配置されていなかった。これはおそらく、背景が複雑であったために猫じゃらしの先端部分を物体として区別できていなかったのだと考えられる。本手法では CustomVision においてバウンディングボックスが設定できない場合には位置座標の算出が不可能となってしまう。そのため、実行する環境としては床など複雑でない背景上に猫じゃらしを映す必要があると考えられる。

また、個体差に伴う認識精度について毛が抜けた個体を回転させずに物体検出を行なった結果については、猫じゃらしの毛が少なかったため猫じゃらしの形が変形してしまっているためであると考えられる。その他の個体は全てのパラメータにおいて予測値が 99%以上であったため、正しく予測できている。本手法では猫じゃらしと認識できない場合、再度認識処理を繰り返す必要がある。そのため、猫が猫じゃらしに飛びつくようなアプリケーションを想定した際には現在の猫じゃらしの位置情報が得られないため、猫が飛びつくまでに時間がかかってしまう可能性がある。

7.2 奥行き座標算出精度

猫じゃらしの奥行き座標の算出精度について、Hololens2 から持ち手までの距離が誤差が $1cm$ となってしまった。本手法において Hololens2 から持ち手までの距離は Hololens2 のハンドトラッキング機能より得られた持ち手の親指までの値である。そのため、ハンドトラッキング機能の誤差かユーザーが実測時よりも腕を縮めてしまっていることが原因であると考えられる。また、本手法での要点としている持ち手から猫じゃらしのまでの奥行き距離 $Depth$ の誤差は $1mm$ であったため、かなり正確に予測できているといえる。

7.3 アプリケーション実装の観点での問題点

背景の複雑さや猫じゃらしの個体差によって VirtualCat が猫じゃらしを捕らえるまでに時間がかかってしまう可能性がある。先述したように背景が複雑である場合、また毛が抜けているような個体の場合、物体の認識が難しくなるようなケースが生まれる。予測値が閾値よりも低くなってしまった場合は猫じゃらしとして認識されないため、三次元位置座標の算出が行えない。図 13 のように全く別の場所を猫じゃらしとして認識してしまった場合は値が算出不可能になってしまったり、誤った三次元位置座標を算出してしまう可能性も生まれる。アプリケーション内において止まっている猫じゃらしを認識するためには最低でも二回の正確な位置座標の算出が必要不可欠である。今回の実験で 1 度の奥行き座標の算出にかかる時間は 4.37 秒であることがわかっている。そのため、一度でも算出が不可能となってしまった場合にはユーザーは 9.14 秒以上もの間猫じゃらしをその姿勢を保たなければならないため、ユーザーの身体的負担が生まれてしまう。

7.4 本手法の利点

本節では本手法の有効性について述べる。Merchand らの手法の場合、目的関数の最小化に加え、次フレームの予測に必要な一次関数の計算が必要であった。本手法では単純な距離計算とルート計算のみであるため、1 フレームにおける計算が少なく、計算コストの削減も可能である。

またモデルベースの手法における 3D モデルや物体の正面の画像を用いる手法 [2, **?**, **?**] についてはあらかじめ識別対象物体のモデルや画像を用意する必要がある。猫じゃらしなどの形状に個体差のある物体を対象とした場合においては、その都度 3D モデルや画像を用意する必要があるため、非効率的である。本手法については事前に必要とする情報は猫じゃらしのの長さのみであり、手の位置座標を用いて容易に抽出可能であるため、事前準備が不要となる。

マーカーを用いた物体の位置座標を算出する手法 [5, 6, 7, 8] についてはマーカーを現実空間上に配置する必要があるため、景観を損ねてしまう可能性がある。しかし本

手法の場合はHololens2のみで実行可能であり、景観を損ねず、事前にマーカーを配置する手間も省ける。

Hololens2 の Ray を用いて奥行き座標を算出することで検出対象の物体の奥行き座標の算出を行う手法 [10] では、検出対象物体が猫じゃらしの先端部分のように小さい場合に Ray が検出対象物体からずれてしまうことにより正確な奥行き座標の検出が困難であった。そこで本手法を用いることで猫じゃらしの先端部分に対して正確な奥行き座標の算出に成功した。本手法は大きさの小さい猫じゃらしのような物体において有効的であるといえる。

7.5 問題点

本手法の問題点として位置座標の算出に時間がかかりすぎる点が挙げられる。本稿では猫じゃらしの停止を検出し、停止したところに VirtualCat が飛びつくというテーマのアプリケーションを実装を行った。しかし、本手法の場合は CustomVision サーバーへの送受信や画像解析によって、VirtualCat が猫じゃらしに飛びつくまでにかなりの時間を要する。この問題は、今後 VirtualCat に猫じゃらしを目や顔で追跡させるなどのリアルタイムの位置座標を知る必要のある動作を実装する際に VirtualCat が猫じゃらしの動きに追いつけない可能性があるため、実装が困難になってしまうと考えられる。しかし、実際の猫は猫じゃらしを目で追う際に、猫じゃらしの動きが速かった場合には正確に目で追えずに固まってしまうケースを考える。この場合、本稿のように座標値の取得が遅くなった場合であっても、『猫じゃらしを目で追い切れていない猫』を再現できる可能性があるとも考えられるため、実装し検証を行う必要がある。

7.6 展望

本稿では実際に猫が猫じゃらしに飛びつくようなモーションの実装を行った。今後他のモーションを実装することで、より実際の猫の動きに近づけると考えられる。その際、先述したように座標値の取得が遅れる場合においても実際の猫の動きに近づかせることができる可能性があるため、各モーションについて実装し検証を行う必要がある。また本稿で実装したアプリケーションでは VirtualCat が空中に配置されてしまっているため、ユーザーは違和感を感じてしまう。Hololens2 の標準機能である空間認識機能により、地面の認識が可能であるため、Unity 上で VirtualCat に重力を与えるなどして、認識された地面上に VirtualCat を配置する必要があると考えられる。また、アプリケーションの展望として、音声認識によりユーザーの声を認識させることでより現実に近い形での実装が可能であると考えられる。例えばユーザーの声から猫の名前を認識することで、名前を呼ぶことで近くに寄ってくるモーションなどの実装が可能となる。

8. 結論

今回の結果で示したように、本手法は形状に個体差のある猫じゃらしにおいて少ない計算量で正確な奥行き座標の算出が可能である。よって本手法は現行研究に対して有効的であるといえる。

参考文献

[1] G. Klinker,K. Ahlers, D. Breen, etc: Confluence of Computer Vision and Interactive Graphics for Augmented Reality, Teleoperations and Virtual Environments,1997.

[2] E. Marchand, P. Bouthemy, F. Chaumette, and V: Moreau:Robust real-time Visual Tracking Using a 2D-3D Model-Based Approach, In International Conference on Computer Vision,1999.

[3] P. Bouthemy: A maximum likelihood framework for determining moving edges, IEEE Trans. on Pattern Analysis and Machine intelligence,1989.

[4] R. Torre, P .Fua, S. Balcisoy: Thalmann:Interation between real and virtual humans:playing checkers, Proc.Eurographic Workshop on Virtual Enviroments 2000(EGVE 2000),2000.

[5] 今本, 加藤, 橘: テーブルトップ拡張現実感システムにおける仮想物体操作インターフェース, ヒューマンインターフェースシンポジウム 2000 論文集,pp.275-278,2000.

[6] 汐崎, 加藤, 橘: テンプレートマッチングによる拡張現実感のための位置合わせ手法, 信学技報,PRMU 2001-230,pp63-70,2002.

[7] U.Newmann and Y. Cho: A self-tracking augumented reality system,Proc. VRST '96,pp.109-115,1996.

[8] G A.Thomas,J.Jin,T.Niblett,and C.Urquhart:A versatile camera position measurement system for virtual reality TV production,Proc.International Broadcasting Convention (IBC'97),pp.284-289,1997.

[9] Hololens2-概要、機能、仕様 — Microsoft Hololens , 入手先〈https://www.microsoft.com/ja-jp/hololens/hardware〉

[10] HoloLens (第 1 世代) と Azure 310 - 物体検出 - Mixed Reality, 入手先〈https://learn.microsoft.com/ja-jp/windows/mixed-reality/develop/unity/tutorials/mr-azure-310〉

[11] Wikitude SDK Android 8.1.0 ドキュメント, 入手先〈https://wikitude.ar.cybernet.ne.jp/lib/templates/help/AndroidJS/objecttracking.html〉

[12] CustomVision のドキュメント- Azure Cognitive Services, 入手先〈https://learn.microsoft.com/ja-jp/azure/cognitive-services/custom-vision-service/〉

[13] MRTK へようこそ — Mixed Reality Tool Kit Documentation, 入手先〈https://learn.microsoft.com/ja-jp/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05〉