飼い主の視線に反応して行動する
ARペットの研究

村瀬　竜則
44201043-6

修士（工学）

指導教員　田中二郎
早稲田大学大学院　情報生産システム研究科

2022.2

Abstract

ペットは、人の心を癒したり楽しませてくれたりするといった理由で飼われる動物である。自律して行動する動物を鑑賞することで得られる癒し、触れたりすることで得られる癒し、そして飼い主の命令に従って行動する伴侶動物としての機能に私たちは喜びを感じてきた。これまで、自律して行動するペットや触感による癒しの研究がなされているが、飼い主の命令によって行動するペットという観点ではあまり研究がなされていない。 近年では、飼い主の命令によって行動するロボット型ペットや端末型バーチャルペットが発売されているが、ロボットは機械的であり、端末型ペットは行動できる空間に制限がある。 また、「おすわり」のような音声によって飼い主の意図をペットに伝達できるロボット型ペットも存在するが、飼い主と伴侶動物であるペット間のアイコンタクトなど飼い主の視点を中心とした研究はなされていない。

本研究では、飼い主の視点を中心としたARペットとの相互的なやりとりを拡張し、命令によって現実空間に存在するARペットを制御できる体験を提供する。

アプローチとしては、ヘッドマウントディスプレイ（Head Mounted Display，HMD）の拡張現実（Augmented Reality，AR）技術を使用し、現実空間にARペットを実現させ、アイトラックによる移動制御、飛び移り制御、アイコミュニケーション、餌やりをできるシステムを提供する。 実装では、HoloLens2による空間理解および物理オブジェクトの認識を行い、実空間に３Dペットモデルを配置することで、あたかも実空間にARペットが存在するかのように投影する。アイトラッキング技術を使用し視線追跡を行い、視線に依存したARペットのアニメーションによる行動を実装する。

予備的なユーザ評価では、システムの使いやすさと効果をテストし、有用性を確認でき、また改善点についても得ることができた。

Keywords: Augmented Reality; eye tracking; AR pets

謝辞
本研究を進めていく中で、指導教員である田中二郎教授には、研究者としての心構えなど多くのことを学びました。異分野出身である私を受け入れ、ご指導いただいたこと心より感謝申し上げます。 またIPLABのメンバーには、情報技術からプライベートに至るまで技術的、精神的に支えてくれたこと感謝申し上げます。本研究を通じて得たことは、これから社会人になる私にとって、業務や企画、人間関係など多岐にわたり必ず大きな糧になると確信しております。. 最後に、遠方への修士の進学を理解してくれた家族に心より感謝申し上げます。

Contents

List of figures

Chapter 1

はじめに

1.1　はじめに

ペットは、人の心を癒したり楽しませてくれたりするといった理由で飼われる動物である。自律して行動する動物を鑑賞することで得られる癒し、触れたりすることで得られる癒し、そして飼い主の命令に従って行動する伴侶動物としての機能に私たちは喜びを感じてきた。これまで、自律して行動するペットや触感による癒しの研究がなされているが、飼い主の命令によって行動するペットという観点では研究がなされていない。近年では、飼い主の命令によって行動するロボット型ペットや端末型バーチャルペットが発売されているが、ロボットは機械的であり、端末型ペットは行動できる空間に制限がある。 また、「おすわり」のような音声によって飼い主の意図をペットに伝達できるロボット型ペットも存在するが、飼い主と伴侶動物であるペット間のアイコンタクトなど飼い主の視点を中心とした研究はなされていない。

本研究では、AR技術を使用することで現実空間にペットを実現させ、アイトラッキング技術を使用した飼い主の視点を中心とした命令に従い行動するペットを提供する。

1.2 本論文の構成

第 2 章では本研究の目的およびアプローチについて述べ、第 3 章では関連研究について述べる。第 4 章では本システムの概要およびデザインについて述べ、第 5 章では本システムの実装について詳述する。第 6 章では本システムの予備的な評価を述べ、第 7 章にて本研究のまとめを述べる.

Chapter 2

研究目的とアプローチ

2.1　研究目的

現実空間において、飼い主ユーザの命令に従い行動するペットを実現することが本研究の目的である。

ロボット型ペットや端末型ペットでは困難な行動や動きをAugmented Reality技術を使用し可能にする。また、命令方法として、ユーザからの音声と視線によるペットに対して自然な命令の方法で制御できるシステムを提供する。

2.2　アプローチ

HoloLens2に搭載されているAR表示技術とアイトラッキング技術を使用する。ペットアバタを実空間内に投影しARペットを配置する。視線をトラッキングし、音声と組み合わせることで、ARペットに語りかける形で命令できる体験を提供する。具体的なアプローチ内容を下記に示す。

(1) 状態制御

飼い主ユーザは、ARペットアバタを見つめることで、ARペットアバタのアクティブ状態を制御できる。また、逆に見つめない状態が続くと、ARペットは非アクティブ状態へと変化する。

(2) アイコミュニケーション

飼い主ユーザの視線とARペットアバタの瞳が合うと、ARペットアバタは飼い主に近づき、声をあげて喜ぶ。飼い主ユーザとARペットアバタのアイコンタクト、飼い主ユーザの位置までの帰還制御が可能である。

(3) 移動制御

飼い主ユーザは、意図した場所へARペットを移動させることができる。視線を使用し、意図した場所を注視し、音声命令「いけ」をトリガーとしてARペットの移動を命令することができる。

(4) 飛び移り行動制御

ARペットへの移動制御中において、飼い主ユーザは視線を実空間の障害物に向け、ジャンプ命令をすることで、ジャンプを制御をすることが可能である。

(5) 餌やり行動

飼い主の視線がペットアバタの口から飼い主の手への視線移動によってペットアバタは食べる行動をとり、飼い主からの自然な餌やり行動が可能である。

(6) 餌やりに伴う体型変化

飼い主ユーザの餌やりの回数により、ARペットの体型が変化していく。飼い主ユーザは、動物的な自然な餌やりのフィードバックを視覚的に理解できる。

Chapter 3

関連研究

近年のペットに関する関連研究では、ロボット型ペット [1][2][3][4][5]、デバイス内で活動するバーチャルペット[6][7][8]、実空間内で行動するARペット[9][10]に分類できる。

3.1　ロボット型ペット

近年のロボット型ペットは鑑賞することで得られる癒し、触れることで得られる癒しという観点で研究が多くなされている。 ロボット型ペットは機械であるため、触感に制限がある。そのため、周りを布等で覆うことや、熱媒体を実装することで癒し効果を向上させる研究[5]やそれに伴うペット自身の感情を表現する研究[1][2][3]が行われている。 また、自ら障害物を避ける、飼い主を認識するなど「自律的に行動するペット」が飼い主に癒しを提供する立場で研究がなされている。[4]

飼い主による命令に反応する相互的な行動においては、音声を使用した「お座り」「ふせ」、タッチの種類によって行動を変えるロボットなどが実装されている。[3]

これらの研究には、以下のようなの制限が存在する。

- 音声による命令のみでは、飼い主ユーザからのペット制御できる行動に限界がある。

- ロボットは機械であるため、飛び移りや体型変化などの実現可能な動きに限界がある。

3.1.1 アザラシ型ロボット「パロ」

産業技術総合研究所の柴田ら[1]によって開発されたアザラシ型ロボット「パロ」。 ロボット本体は、ユーザが触った場所、なでた方向、たたかれたこと、抱っこされたことなどを認識することができる。このセンサーは軟らかく、パロの体を軟らかく感じられるように作られているため触感による癒し効果がある。。

音の方向と、多言語の音声を認識し、自分の名前を学習する。人間の呼び掛けに反応するようになったり、なでられたり抱きかかえられると喜ぶよう設計されている。ユーザとロボットとの相互的な作用として、なでる、抱きしめる、叩かれるなどのことに反応して行動するよう設計されている。

3.1.2 ぬいぐるみ型ロボット“コモコモ”

株式会社東芝の鈴木ら[2]は、ペットロボット“コモコモ”を開発した。 このロボットは、感情機能を持ち、人物識別能力と対人関係能力を自動的に学習発展できる。ユーザの働きかけに対してロボットに発生する情緒と利用者の顔パターンとが関連付けられて記憶され、ユーザに対する好き嫌いを決めるのに使われる。好感情を与える利用者にロボットは懐くようになり、ペットロボットが特定の人物と親密になるメカニズムを設計している。

コモコモは、ユーザの接近、利用者による撫でや叩きの接触刺激、利用者の音声語彙、顔の有無と人物の別、ハンドジェスチャーを認識しつつ、動作と音声で応答する。

3.1.3 犬型ペットロボットAIBO

1999年、SONY[3]は、家庭用ペットロボットAIBOERS-110を開発した。ERS-110では学習機能が導入され、ユーザからのインタラクション(なでるなど)を通じ

て、AIBOは人の接し方、あるいは周りの環境によりそれぞれ異なるロボットに変化していく。

このAIBO-ERS110に続いて、ERS-210、ERS-7など進化を続け、2021年現在では、SONYは、「AIBO」の新型後継機「aibo」を発売し、しなやかな動きを表現するなど改良し、加えて、音声認識によるユーザとペットロボットの音声インタラクション（おすわりや伏せ）、また鼻先のカメラによる画像認識を使用したオーナーの識別、障害物の検知が可能になっている。

3.1.4 自律移動型LOVOT

LOVOT[4]の特徴として、部屋全体の様子を正確に把握して部屋の設置物、人の顔、声のする方向まで正確に察知することが可能である。

LOVOTの瞳は、6層のディスプレイ構造で、10億通り以上の組み合わせにより、視線の動きや瞳孔の開き、眼の揺れなどの表現を可能としている。加えて、声帯をシュミレーションした独自のデジタルシンセサイザが搭載されており、動的に音声を生成することができ、LOVOTの状態に合わせて発声させる際に毎回異なる声を出すことのできる鳴き声を表現している。

3.1.5 Nene: an interactive pet device

PaFanら[5]は、熱や触覚などのペットの特徴をリアルタイムで再現するインタラクティブな猫型のデバイス、視覚・音響機能のモバイルアプリなどの一連のデバイス「ねね」を開発した。この研究では、孤独とストレスの側面に焦点を当てて、ネネが人間に与える心理的影響を調査する。評価は、ユーザーに心を落ち着かせリラックスさせる効果をもたらし、ストレスや孤独感を軽減する可能性があるため、肯定的な結果を示している。

3.2 バーチャルペット

デバイス内で活動するバーチャルペットも同様に、自律して行動するペットを世話をすることで得られる癒し[6]や、触れることで得られる癒し効果を提供する研究がなされてる。[8] また、ペットとしての機能に加え、天気情報や目覚まし機能などを付属させ利便性を高める研究が行われている。[7]

飼い主との相互的な行動としては、スマートフォンやコントローラを使用した餌やり、デバイスのボタンを使用した映像内での移動命令などが可能である。[6][8]

しかしながら、バーチャルペットは端末の映像上で行動し、コントローラ操作することから以下の ような制限が存在する。

- バーチャルペットの餌やり等の操作はコントローラのボタンやスマートフォンで行われるため、実際の餌やりする時の行動とは異なり機械的である。
- バーチャルペットはデバイスの外（実空間）上で行動することができない。

3.2.1 たまごっち

1996年株式会社バンダイの横井ら[6]によって開発された卵の形をした携帯型デバイスの中に登場するバーチャルペット「たまごっち」。白黒液晶画面の下部に3つのボタンを備える。ボタンは左からコマンドの選択・決定・キャンセルに割り当てられている。
機能として主に、餌やり、トイレ掃除、病気の治療、画面内での電気を消すなどが可能である。これらを定期的に行うことで「たまごっち」の体力状態が良くなり、逆に、餌や、掃除を怠ると状態が悪くなり最悪の場合には死ぬこともある。世話をしながら時間が経てば「おやじっち」等のキャラクターに成長する。キャラクターは体重や機嫌に左右されるよう設定されている。また、画面を切り替えることでたまごっちは部屋を移動することができる。

3.2.2 バーチャルペット「Juno」

2021年株式会社ユピテル[7]によって開発された箱型デバイスの中に登場するバーチャル猫型ペット「Juno」。
飼い主ユーザがスマートフォンを使用した餌やりや音声認識を使用した「おはよう」等の言葉かけを行うことで、懐き度レベルが一定間隔で上がる。スマートフォンと連携していることで遠隔からの餌やりが可能。レベルが成長するとバーチャルペット「Juno」を介して、天気情報の提示や、タイマー機能、目覚まし機能、IoT家電の操作などの機能が扱えるようになる。

3.2.3 LITTLE FRIENDS –DOGS CATS-

イマジニア株式会社[8]によって開発されたNintendo Switch™ソフト「LITTLE FRIENDS –DOGS CATS-」。
Swich付属のJoy-Conというコントローラを使用し、仮想的なペットに触る、撫でる、頬を引っ張る、餌をあげる等の相互的な行動が可能である。これらの操作した時にコントローラに振動が伝わり触感フィードバックを実現している。 また、着せ替え機能、SNS投稿機能などを有している。

3.3 ARペット

最後に、実空間内で行動するARペットである。ARペットは実空間上に映像を映し出し行動するペットである。近年では、Weiら[9]は、実空間の環境に対して自律的に行動するARペットを提案している。また、Nahalら[10]は環境内の他の実在の人々に関して、実空間の人との衝突を検知し、ARドックが倒れるなどAR犬を見ることができるまたは見ることができない他の人々との社会的な相互作用が変化することを示している。 また、Wuら[11]は、ペットにナビゲーション機能など便利な機能を持たせる観点から研究を行なっている。

しかしながら、ARペットは、飼い主の命令に従って行動するペットという観点から研究はされていない。

以上のことから、本研究では、現実空間において、ARペットの行動を飼い主の命令によって制御できる体験を提供する。

3.3.1 Scene-Aware Behavior Synthesis for Virtual Pets in Mixed Reality

Weiら[9]は、HoloLens2を使用し、実空間におけるシーン理解を行い、仮想ペットが複合現実で独立して自然に行動できるようにするARペットの行動を環境に対して組み込む新しいアプローチを提案した。キャプチャされたシーンを前提として、著者のアプローチは一連のペットの食べる、寝るなどの行動を合成した。次に、食べた後に休むなど時間経過に伴う各動作を実際のシーン内の場所に割り当てた。

3.3.2 Walking Your Virtual Dog: Analysis of Awareness and Proxemics with Simulated Support Animals in Augmented Reality

Nahalら[10]はARドックが環境内の他の実在の人々に関して、空間把握に関する移動を支援した。著者は、ARドッグを使用し、ARドックとそれを見ることのできない実空間の人との衝突を検知し、ARドックが倒れるなど、参加者の移動行動、およびAR犬を見ることができるまたは見ることができない他の人々との社会的な相互作用が変化することを示した。

3.3.3 Real-world AR Pet: An Emotional Supporter and Home Guide

Wuら[11]は、孤独感やアルツハイマーの問題を解決するために、特に高齢者に同行してナビゲートするためのRealworldARペットシステムを提案している。この研究は、スマートフォンを通して、ユーザはARペットを視覚的に理解し、家の中をユーザに同行してナビゲートする体験を提供している。

3.4 アイトラック

近年、ユーザの視線を検出するアイトラッキング技術を使用したデバイスが開発されている。視線追跡デバイスは、頭部固定型デバイス[12]、遠隔アイトラッキングデバイス[13][14][15]、ウェアラブル型デバイス[16]、そして統合型デバイス（HoloLens）に分類される。

これらのデバイスは、マウスやキーボードといった様々な入力機器の代替となりうるヒューマンインターフェイスであり、文字入力など手足が不自由な障がい者のコミュニケーション装置やユーザの視線を解析するデータ分析などに活用されている。 仮想映像表示デバイスを投影するデバイスHololensにアイトラックセンサが搭載された統合型デバイスが開発されたことで、仮想空間上における視線ベースで行われるインタラクションの研究が行われている。

具体的には、AR空間内においてユーザの視線によって任意の情報を取得する研究[17]や、視線に基づく仮想空間内での観測方法の研究。[18]AR空間内での視線を使用した対象オブジェクトを選択する自然なインタラクションの研究[19]などがなされている。また、視線情報に加えて、脳波を組み合わせることでVR空間でのテキスト入力をサポートする研究[20]も行われている。

視線を使用した制御の観点からは、視線追跡技術を使用し、ロボットを視線で制御する研究などが行われている。具体的には、電動車いす移動を視線ベースで制御する。[21]などがあげられる。

アイトラッキングの基礎的な研究では、VR空間においてユーザのハンド操作時どこを見ているかを記録し解析する[22]。などのデータ解析の観点から研究。また非装着型アイトラッキングトラックデバイスについての研究。[13][14]アイトラッキング技術の精度を向上する。[23][24]などの研究がなされている。

本研究では、仮想空間のARアバタを視線を使用し制御するという観点から研究する。飼い主からの視点を中心としたARペットとの相互作用を実現する。具体的には、注視することで起床するフィードバック、ユーザの見ている実空間上の場所の伝達、ARペットとのアイコミュニケーション、ユーザ視点からの視線移動による餌やりの体験が可能となる。

3.4.1 頭部固定型デバイス EyeLink1000 Plus

EyeLink1000 Plus [12]は、据え置き型でかつ頭部を固定する形で視線追跡を行うアイトラックデバイスである。頭部が固定されているので視線データが正確に判断できる。視線データを精密に分析する医療や心理学の分野で使用されている。

3.4.2 遠隔視線追跡デバイス Tobii pro フュージョン

Tobii Proフュージョン[15]は、据え置き型のアイトラックデバイスである。デバイスから遠隔でユーザの視線を追跡することが可能で、装着する必要性がないため、実験参加者の都合やシナリオに合わせてデータが収集できる。

3.4.3 メガネ型デバイス Tobii pro グラス

Tobii Proグラス[16]は、メガネ型のアイトラックデバイスである。このタイプは、装着型なので野外でのユーザからの視点データの記録、解析することや、スマートフォンの操作や運動など頭部を頻繁に動かす必要のある分野で活用されている。

3.4.4 StARe: Gaze-Assisted Face-to-Face Communication in Augmented Reality

Radiahら[17]は、拡張現実（AR）でサポートされている会話中の視線追跡の使用について調査した。このシナリオでは、ユーザーは実際の会話の邪魔をすることなく、会話をサポートする情報を取得できる。ユーザーが視線を使用し、徐々に情報を明らかにできることを提案した。情報はユーザーの頭の周りに表示され、視線がその領域に明示的に当たると完全に見えるようになる。ユーザーは文脈情報と視線の双方向性で会話を強化することに前向きであったことを示している。

3.4.5 Anyorbit: orbital navigation in virtual environments with eye-tracking

Benjaminら[18]は、非常に有利な軌道ナビゲーション技術であるAnyOrbitを紹介している。これは、視線追跡を使用して軌道の動きの中心を制御する、仮想環境での直感的でハンズフリーの観測方法を提供する。3D天文データの観測や観戦スポーツなど、仮想/拡張現実のヘッドマウントディスプレイおよびデスクトップセットアップでのいくつか実証されている。

3.4.6 Combining Brain-Computer Interface and Eye Tracking for High-Speed Text Entry in Virtual Reality

視線によるテキスト入力システムでは、視線追跡とブレインコンピューターインターフェイス（BCI）が一般的に使用される。Xinyaoら[20]は、定常状態視覚誘発電位（SSVEP）と視線追跡を組み合わせることにより、VRにテキストを入力するためのハイブリッドBCIシステムを紹介した。

3.4.7 EyeHead: Synergetic Eye and Head Movement for Gaze Pointing and Selection

Ludwigら[19]は目と頭の相乗的な動きを活用し、視線とポインターの動的な結合、空中に停止するホバーインタラクション、選択の周囲の視覚的探索、およびターゲットの反復的な確認を可能にする3つの新しい手法を提案した。

3.4.8 Gaze behaviour on interacted objects during hand interaction in virtual reality for eye tracking calibration

Ludwigら[22]は、仮想現実で手のオブジェクトの相互作用中に注視固定を達成する確率とタイミングを調査した。15人の参加者を対象に、仮想オブジェクトとの対話中に視線を記録した評価を実施した。データを分析して、さまざまなオブ

ジェクトタイプの相互作用のさまざまなフェーズでの凝視の確率に影響を与える要因を見つけた。

3.4.9 Exploring Eye-Gaze Wheelchair Control

視線は、車椅子やロボットの操縦に使用される可能性があり、それによって、実空間を移動する場所を選択するサポートした。この論文は[21]、視線制御インターフェースの実現可能性を調査し、VR実験では、3つの制御インターフェースが18人の健常者によってテストされ、（1）視線による進路方向ボタン選択、（2）ドライバーの前の地面の継続的な注視点までの移動、（3）地面に配置されたターゲットへのナビゲーション。結果は、ナビゲーション方式が優れたパフォーマンスを示し、ユーザーにも最も好まれた。

3.4.10 Eye Gaze Controlled Robotic Arm for Persons with Severe Speech and Motor Impairment

Sharmaら[13]は、重度の発話および運動障害のあるユーザーがロボットアームを操作するための視線制御インターフェースの設計と開発をし、重度の発話および運動障害のあるユーザーを対象としたユーザー研究を報告した。

3.4.11 Environmental Control System for Locked-in Syndrome Patients Using Eye Tracker

Analynら[14]の主な目的は、アイトラッカーによって制御される支援技術を使用し、一部の電気/電子デバイスの制御においてLISに苦しむ患者の生活を改善することである。コントローラーと、ユーザーが視線を入力として4つのコントローラのオンとオフを切り替えることができるソフトウェアを設計し、環境制御システムを開発した。

3.4.12 Long range eye tracking: bringing eye tracking into the living room

Craigら[23]は、アイトラッカーがスマートTVなどのデバイスと対話する際には自宅の居間などのより困難な環境で動作する必要があり、居間環境での視聴者の動きの自由を可能にするため、非装着型の視線追跡システムを提示している。

3.4.13 Eye Movement Synthesis

視線データの合成のために畳み込みフィルタリング技術が導入されている。これは生の視線位置座標とアイトラッキングで得られたデータを合成するためである。Andrewらは[24]、アイトラッキングで得られたデータと実際の視線位置のデータを比較し、評価を行った。

Chapter 4

システムデザイン

4.1　システム概要

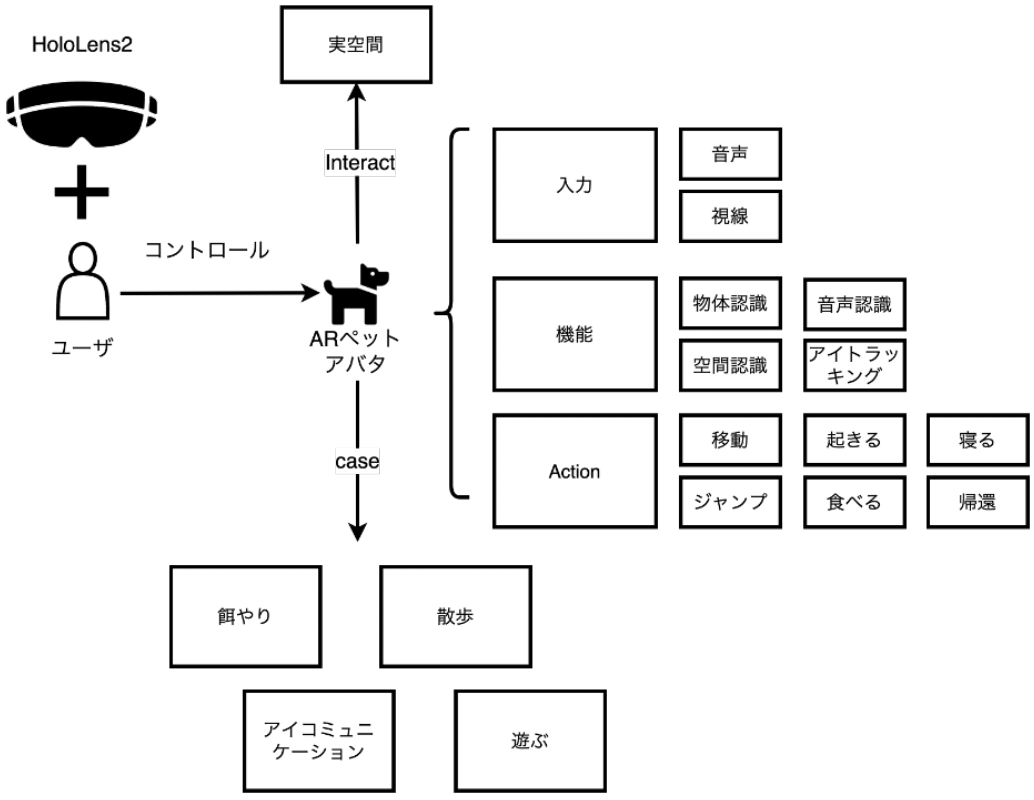

Fig. 4.1 システム概要

図4.1のように、飼い主ユーザは、実空間上で仮想的なペットアバタを自由に制御できる。 飼い主ユーザが実空間内で自由にARペットの行動を制御するために、本システムは、実空間上に仮想的なペットアバタを投影、飼い主のユーザの視線をトラッキングするために、以下のハード・ソフトウェアを使用する。

- Microsoft HoloLens2（Hardware）
- Unity 3D（Software）
- Mixed reality tool kit（Software）
- Azure Custom Vision Service(Software)

Microsoft HoloLens[25]は，ARの技術を提供するシースルータイプの ヘッドマウントディスプレイであり、3D オブジェクトとして仮想的なペットアバタを 実空間上に投影する。またHoloLens2には、アイトラックセンサーが搭載され、これを使用し、飼い主ユーザの見ている座標取得、視線を使用した自然なペットアバタの行動制御に使用する。

4.2 行動可能領域の決定

4.2.1 空間理解

飼い主ユーザが自由に実空間内でARペットアバタの様々な行動を制御するには、周辺空間が制御可能な空間か否かについて判断をする必要がある。本システムではARペットアバタの行動可能領域を決定するための最初のプロセスとして、空間上の床面、壁面、障害物を識別する。

Fig. 4.2 空間認識設定図

- Update Interval...0.1
  Spatial Mapping のアップデート間隔
- Observer Shape...Axis Aligned Cube
  球体や箱などの領域を定義する図形
- Triangles/Cubic Meter...500
  立方メートル毎の空間に適用する三角ポリゴンの数
- Display Option...オクルージョン
  Spatial Mapping のメッシュの表示/非表示の設定

空間認識の設定を図4.2に示す。Microsoft HoloLens を介して視界上の実空間に対し、深度センサおよび Mixed RealityToolkit（MRTK）[26]の Spatial Mapping / Understanding ライブラリ[27]の組み合わせを使用して、周辺空間形状の特徴点を抽出し、蓄積された空間上の特徴点群を基にメッシュ化し、空間認識を行う。この処理を0.1秒間に1回繰り返すことで、常に空間の物体を認識している状態を維持する。 また、メッシュ化したものをオクルージョンに設定することで、ユーザがHoloLens2を介して空間を眺めてもメッシュ化が非表示になるようにした。

4.2.2 物理オブジェクト理解

ユーザの視界内に映る物理的なオブジェクトの認識には，Microsoft 社よりクラウドサー ビスとして提供されている Azure Custom Vision Service –

Object Detection[28]を使用する。Microsoft Azure 製品の 1 つである Azure Custom VisionService は，あらかじめ物理対象オブジェクトの学習モデルを作成し、そのモデル に対し新たに与えられた画像との特徴量を比較して、物理対象オブジェクトが画面内に存在するか否かを判定するものである。 今回は、物理オブジェクトとして日常的な空間に存在する椅子を代表として扱い、ARペットアバタの飛び移り行動がより滑らかに行えるようにした。

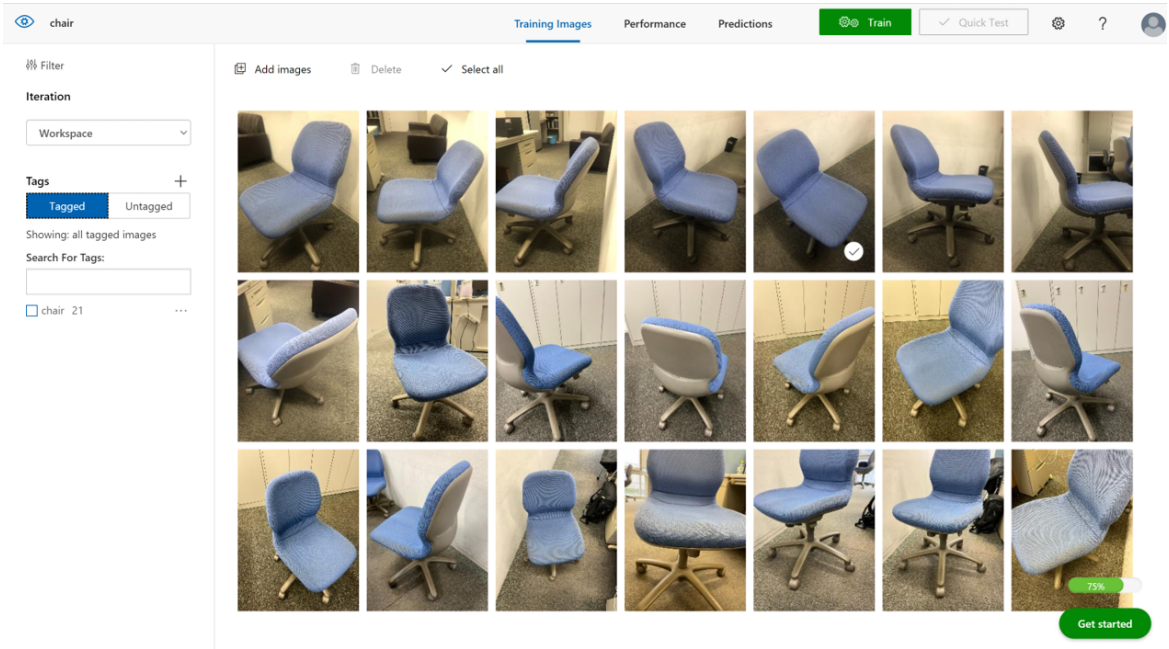

Fig. 4.3 学習のための画像登録及びタグ付け

Microsoft Azure 製品には同様の機能を持つAzure Cognitive Service に使用されるモデルは Microsoft 社が事前に定義したもののみであるため、本システムではユーザ独自のカスタマイズが可能なAzure Custom Vision Service を使用している。図4.3に学習のための画像登録およびタグ付けの設定画面を示す。ユーザ独自の学習モデルを作成するために、例として物理対象オブジェクトである椅子の画像を新規に登録しているものである。また、これらの画像を一枚ずつ椅子の範囲を指定し、タグ名”chair”としてタグ付けを行った。 高い認識精度である学習モデルとして成立させるためには、図 4,3の登録処理を 繰り返す必要があり異る椅子の画像21枚を準備し、それらの画像を全て登録した後、Azure Custom Vision Serviceの

学習モデルとしてトレーニングを行い、空間上での椅子の位置検出を可能とした。

4.3 ARペットアバタの行動制御

4.3.1 アクティブ制御

本研究の目的は、ARペットを空間内で制御することであるが、ペットとしての機能を不随するため、飼い主ユーザは、起きている(アクティブ)寝ている（非アクティブ）の状態を視線を使用することで、制御できるようにした。これは、あたかも飼い主の視線をARペットが感じ取り起きる行動をとる。視線がないつまりアバタに興味がないと寝る行動をとるものを模倣した制御の形である。また、寝ている状態では、例え飼い主がペットアバタに命令をしたとしても行動をしない。視線を使用したアクティブ化を行うことでARペットは飼い主の命令に従うようになる。

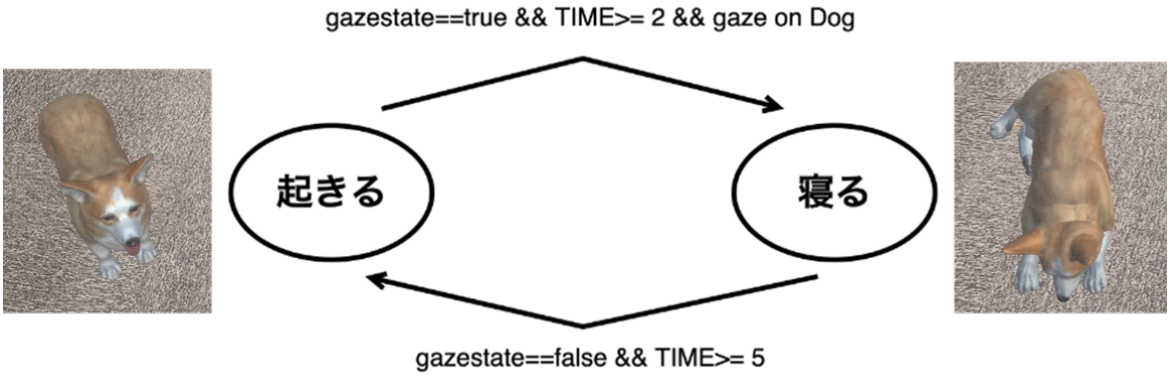

Fig. 4.4 アクティブ制御状態遷移図

図4.4に起きる寝る制御のための状態遷移図を示す。ARペットアバタが寝ている場合においては、視線入力がある、アバタの上に視線がある、その状態が２秒間以上続く、これら三つの条件が満たされると、アバタは起きる行動をとる。逆

に、ARペットアバタが起きている場合においては、視線入力がない、その状態が５秒間続く、この２つの条件が満たされると寝る行動をとる。

4.3.2 アイコミュニケーション

ユーザの視線に着目したARペットとのアイコミュニケーションを構築する。また、これを使用し、飼い主ユーザのもとに帰還する行動を制御することも可能である。自律的な帰還自体は、ロボット型ペットでも可能であるが、本研究は、目と目による自然な帰還制御および、アイコミュニケーションを可能としている。飼い主ユーザの目とARペットアバタの目が合った時、ARペットは飼い主ユーザの現在地まで移動した後、ユーザの元で声を上げながら喜ぶ。 システムは、ユーザの視線ベクトルを常に取得しており、ユーザの視線がペットアバタの瞳に位置するコライダとの衝突を検知することで、視線と視線が合ったことを認識している。衝突を認識した際、ユーザの現在地を取得し、その座標から前方0.5mまでペットアバタが移動し、音声とともに喜ぶアクション行動をとる。これにより視線と視線の相互的な行動を可能とする。

4.3.3 移動制御

本研究の移動命令は、あたかも飼い主の意図を読み取って移動する信頼的な移動ができるペットをデザインする。そのため、ARペットアバタの移動制御は、飼い主ユーザの視線と音声を使用し行う。これは、視線で目的地をとらえ、音声がトリガーとなる自然な移動命令を可能とするためである。

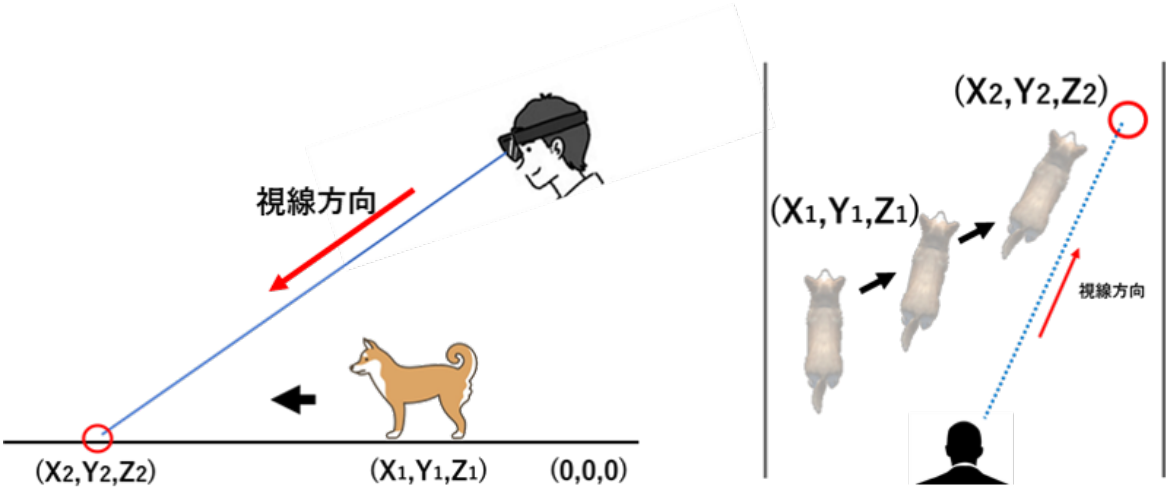

Fig. 4.5 アバタ移動制御

図4.5にアバタの移動制御図を示す。システムによりユーザ(原点)からペットアバタ(x1,y1,z1)までの座標を常に記録している。飼い主ユーザの音声命令「いけ」を認識した時点で、飼い主ユーザの視線ベクトルと実空間の床との交点座標を取得し、意図する目的地(x2,y2,z3)をシステムにより記録を行う。記録された座標データをもとに、ペットアバタは、現在地から、目的地まで角度を変えながら移動する。音声認識には、Mixed Reality tool kitのInputDataProvider[29]から特定の文字(この場合は"行け")に対するスクリプトを指定し、音声認識による行動遷移を行っている。

4.3.4 飛び移り制御

ARペットアバタの飛び移り行動は、飼い主ユーザの視線と音声を使用し行う。これは、移動制御に伴って、行うことのできる制御であるからである。移動制御と同様に、視線で目的地をとらえ、音声がトリガーとなる自然な移動命令を可能とする。

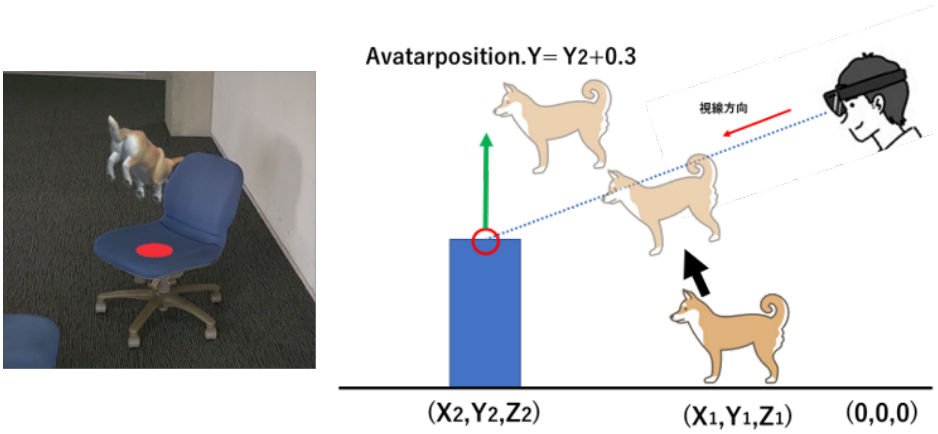

Fig. 4.6 アバタ飛び移り制御

図4.6にアバタの飛び移り制御図を示す。システムは、音声命令"ジャンプ"を認識するとペットアバタは障害物の上まで物理オブジェクトを認識し、座標情報及び高さ情報を取得する。空間認識のみで障害物を検知することは可能だが、ここでは、高精度で飛び移り行動が可能となるよう、Azure Custom Vision Serviceの物体認識によるユーザから障害物までの座標特定、Mixed RealityToolkit（MRTK）の Spatial Mapping / Understanding ライブラリによる障害物の高さ認識を併用して物理オブジェクトに対し飛び移り行動ができるようにする。これらの座標と高さのデータをもとに、アバタのジャンプした目的地を決定するとともに、ジャンプ高さの最高到達点を検出した目的地の高さの30cm高く設定することで上から目的地に到達することを可能としている。

4.3.5 餌やり制御と体型変化

図4.7に視線スワイプ図と実際のARペットの行動を示す。 ARペットアバタの餌やりは、飼い主ユーザの視線を使用し行う。ユーザが視線をアバタの口からユーザの手に視線をスワイプする動作を認識する。これは、餌をあげる際に飼い主ユーザペットの口から手に自然に視線を落とす行動に着目したものである。音声命令お座りの後に、ARペットアバタの口から飼い主ユーザの手に視線がスワイ

プすると、ARペットは餌を食べる行動をとる。これによりユーザは直感的な餌やりが可能になる。

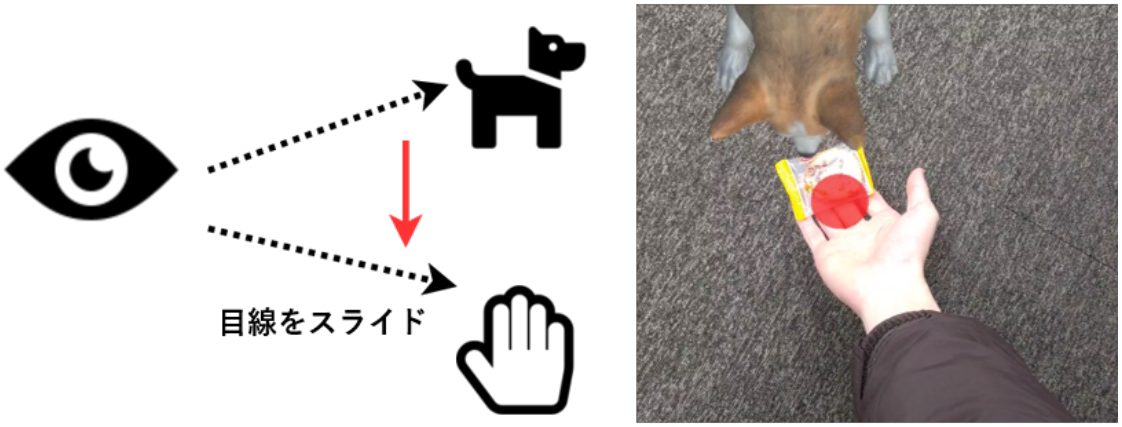

Fig. 4.7 視線スワイプによる餌やり

また、従来のロボット型ペットでは、体型変化は容易ではないため実装されていない。そのため本研究では、餌を食べたARペットの自然な相互作用として、一定のバイアスで体型が増加方向に変形していくようにデザインする。

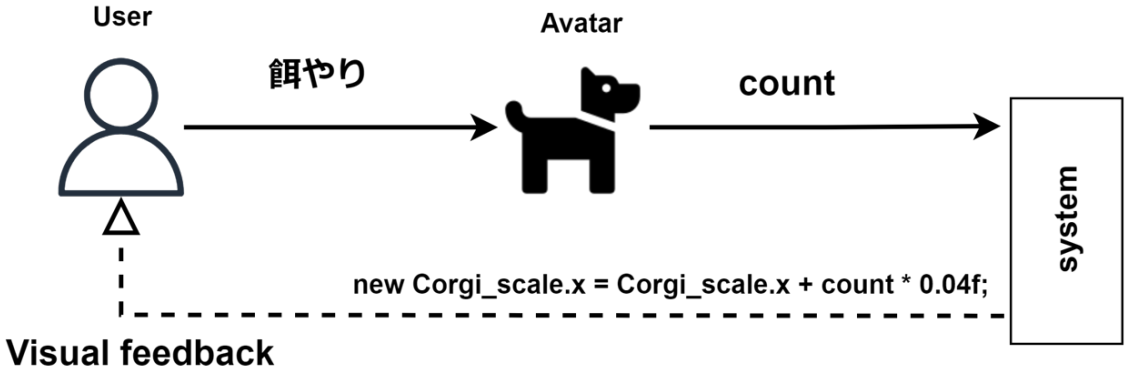

Fig. 4.8 餌やり行動に対するフィードバック

図4.8に、餌やり行動に対する視覚的なフィードバックの図を示す。ユーザがペットアバタに対し餌やり行動を行うと、行った回数をシステムが記録する。記録した回数に0.04倍する。これは、自然な体型変化にするためこの値に設定した。この値にアバタのX軸つまり横軸方向に、増加させ、視覚的に増加方向に体

型が変化するようにユーザに対し、フィードバックを行った。 図4.8に餌やり回数による体型変化を示す。標準状態と10回、20回餌をあげた状態のように、ユーザが餌やりの回数を増やすことで体型は徐々に変化していく。

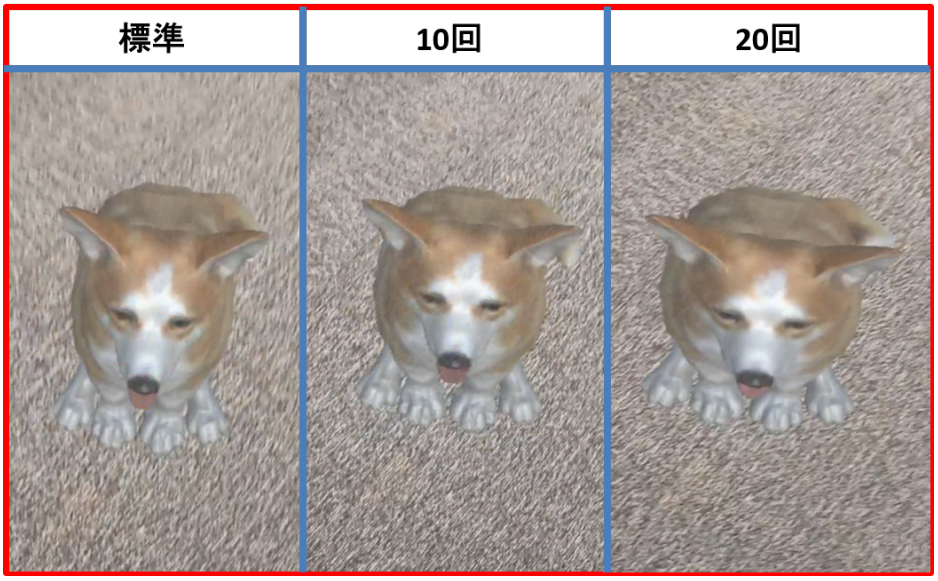

Fig. 4.9 餌やり回数と体型変化

Chapter 5

システム実装

5.1　ハードウェア

　このシステムのハードウェアはHoloLens2（図5.1）というAR表示デバイスを使用して構築した。仮想ペットアバタをHoloLens2を用いて実空間の床に投影することでARペットを実現した。ユーザの視線はこの HoloLens2に搭載されたアイトラッキング技術[30]を使用することで、ユーザの視線の向き、入力、動きを検出した。

Fig. 5.1 HoloLens2

5.2 開発環境

開発に使用したソフトウェアを以下に示す。

5.2.1 Mixed Reality Tool Kit

Mixed Reality Toolkit（MRTK）は，Unity 上でクロスプラットフォームのMRアプリケーションを開発するためのライブラリであり、Microsoft 社が提供している。本システムでは Microsoft HoloLens[25]を使用した空間認識および理解を行うために、Spatial Mapping/Understanding ライブラリ[27]を使用している。また、speech Input Handler[29]による音声認識、およびIMixedRealityEyeGazeProviderによる視線追跡を使用すし、本研究のシステム構築を行う。

5.2.2 Unity3D

Unity 3D[31]は，OS に依存しない形での 3D開発プラットフォームであり，ゲーム制作のみならず、AR/MRアプリケーション開発が可能である。本システムではUnity 内に5.2.1のMixed Reality Tool Kit をライブラリとして読み込み、UWP（Universal Windows Platform）アプリケーションとしてコンパイルした後、Visual Studio を介してMicrosoft HoloLens2 にアプリケーションをコンパイルしている。

5.2.3 Azure Custom vision Service

独自の画像データをAI処理し、「画像分類」と「物体検出」を行うサービスである。ユーザの視界内に映る物理的なオブジェクトの認識にDetection を使用する。Microsoft Azure 製品の1 つであるAzure Custom Vision Serviceは、あらかじめ物理対象オブジェクトの学習モデルを作成し、そのモデルに対し新たに与えられた画像との特徴量を比較して、物理対象オブジェクトが画面内に 存在するか否かを判定するものである。今回は、物理オブジェクトとして日常的 な空間に存在する

椅子を代表として扱い、ARペットアバタの飛び移り行動がより 滑らかに行えるようにした。

5.3 空間認識

Mixed-RealityToolkit のSpatialMapping/Understandingライブラリを使用し、HoLolens2のヘッド部からRay（光線）を飛ばし、空間の特徴点を検出することで、実空間に床、物体を認識、識別する。またこの処理を0.1秒間に 1 回行うことで、ユーザの移動に伴った空間認識を可能としている。

5.4 アバタと空間配置

ARペットアバタの3Dモデルは、unity asset store[32]から骨格を有するfbxファイルをダウンロードした。ダウンロードしたものをunityにインポートしゲームエンジン上で3Dモデルを扱うことのできるようにした。図5.2にrigid bodyの設定を示す。質量を持たせるために 1 に設定し、また重力を付加できるrigid body を3Dモデルに付加し、また、回転軸をX,Y,Zそれぞれにおいて固定した。これにより、重力があり、各座標へ傾きのない自然な3Dアバタを構築した。

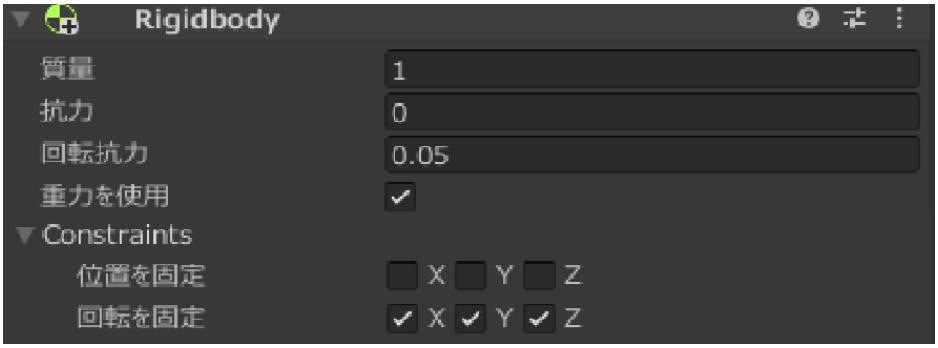

Fig. 5.2 重力付加と軸回転の固定

図5.3にメッシュした床と配置したアバタを示す。配置した床に対し、重力との衝突を検知し床に3Dモデルが配置できるよう衝突を検知および、重力に対し抗力が働き床に対して静止状態をとることのできるコライダを3Dモデルの胴体範囲

内に設定し床に配置するようにした。また、床をメッシュ化することで、環境に左右されずかつユーザには床オブジェクトが見えず実空間上に3Dモデルを配置できるようにした。

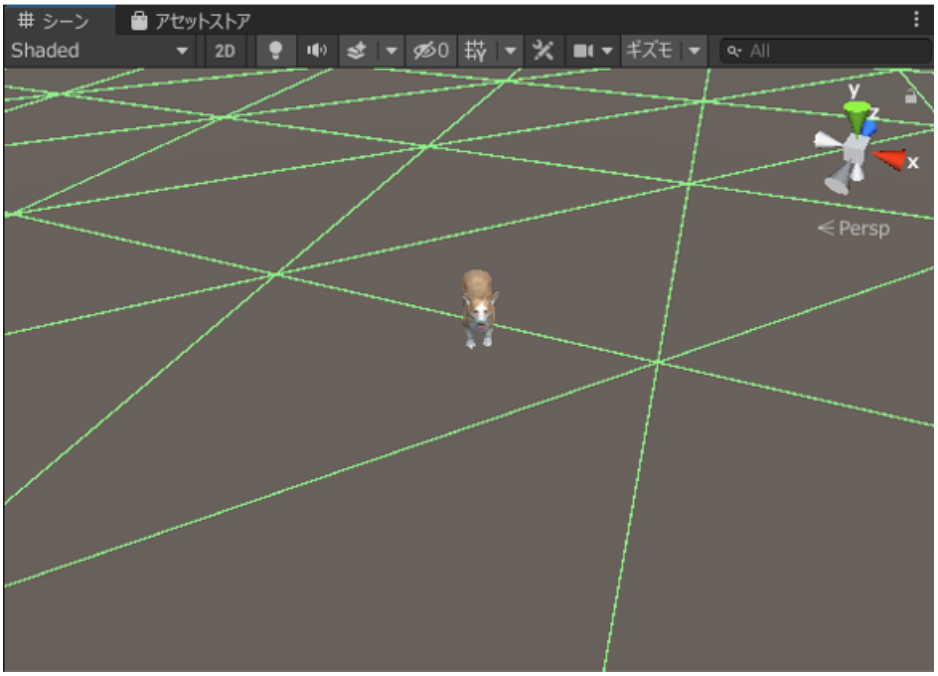

Fig. 5.3 床のメッシュ化と3Dモデルの配置

5.5 アニメーションと命令による遷移

3Dモデルの骨格を制御し、ペットの行動を制御する。付属してあるアニメーションに加え、骨格単位で座標と回転を時間経過とともに制御し、歩くジャンプなどのアニメーションの位置や回転具合を調節し、高いジャンプや走るなどの行動できるようにした。 関数の遷移を視覚的に制御できるアニメータを使用し、bool値のture/falseにより行動が遷移するように設定した。また、命令による行動遷移を可能にするため、HoloLens2の音声認識、視線追跡を使用し、ユーザの音声や視線による入力により、bool値がture/falseに切り替わるようにし、ARペットのアニメーションが遷移できるようにした。

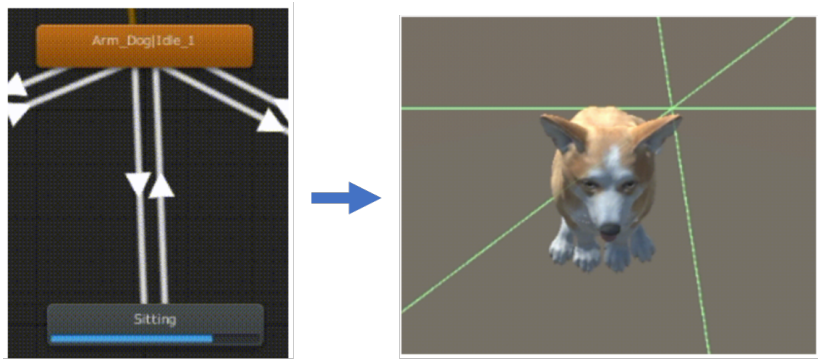

Fig. 5.4 アニメータによる行動遷移

5.6 ARペットアクティブ判定

Mixed Reality tool kitのIMixedRealityEyeGazeProvider[30]による視線追跡機能を使用し、ユーザの視線入力の有無を判定する。図5.5にソースコードを示す。bool値gazeStatusに視線入力の有無情報を格納し、もし視線入力がない場合はbool値eyedownにtureで返す。反対に、視線入力がある場合は、falseで返す。これにより、eyedownで有無判定を可能とする。

```
using Microsoft.MixedReality.Toolkit;
            ⋮
public Boolean eyedown()
{
    bool? gazeStatus = CoreServices.InputSystem?.EyeGazeProvider?.
                       IsEyeTrackingEnabledAndValid;
    if (gazeStatus == false)
    {


        return true;
    }

    else
    {
        return false;
    }
```

Fig. 5.5 視線入力の有無判定

次に、２秒間以上の視線入力があり、視線がARアバタの上にある状態と、５秒間以上視線入力がない状態を判定する。まず、ARアバタの上に視線があるかないかの判定には、章5.4で使用したアバタ全体へに配置した衝突を検知するコライダを使用する。また複数のコライダと区別をするため、３Dモデル全体に付加したコライダを"Dog all"のタグ付けをしグループ分けを行う。図5.6にソースコードを示す。コライダの衝突を検知すると、wakeをtureで返すように設定した。

```
void OnTriggerStay(Collider Collider)
{

    if (Collider.gameObject.tag == "Dog all" )
    {
        wake = true;

    }
}
```

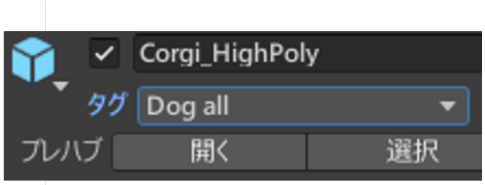

Fig. 5.6 視線と犬アバタとの衝突判定

図5.7にカウントのソースコードを示す。DataTimeofssetを使用し時間カウントを行う。条件分岐で３Dモデルのアニメーションが起きる。そして、寝るを切り替えられるようにした。視線入力も衝突もない場合が5秒以上経過すると、アニ

メータ遷移関数Sleep();により、アニメータのbool値が変化し、ARペットの行動アニメーションが遷移し、寝る行動に移り変わる。 逆に、視線入力があり、コライダとの衝突を検知するとStartCountDown2が読み込まれ、カウントが開始される。開始されてから２秒間経過すると、sleepend();関数が読み込まれる。つまり起きるアニメーションに遷移する。この判定によりARアバタのアクティブ制御が可能となるよう構築した。

```
    if (startCountDown < 0 && eyedown() && wake == false)
    {//first time enter
        startCountDown = now;

    }
    else if(startCountDown > 0 && now - startCountDown > 3
                                    !eyedown() && wake == false)
    {
        if (now - startCountDown > 2 && eyedown() == false)
            sleepst();
        startCountDown = -1;


    }


    if (now - startCountDown > 5 && wake == true)
    {
        if (startCountDown2 < 0)
        {//first time enter
            startCountDown2 = now;

        }
        else
        {
            if (now - startCountDown2 > 3)
            {

                sleepend();
                startCountDown2 = -1;
                wake = false;
            }
        }
    }

}
```

Fig. 5.7 アクティブ化判定

5.7　アイコミュニケーションのための視線交差判定

図5.8に3Dモデルの目に衝突を検知するコライダの設置位置を示す。モデルの目にコライダを設置する。同様に、視線情報にもコライダを設置し、２つのコライダ同士の衝突を検知する。章5.6のコライダによるグループ分けと同様に、タグ"Dog"でグループ分けし、他のコライダとの衝突検知を防ぐようにする。

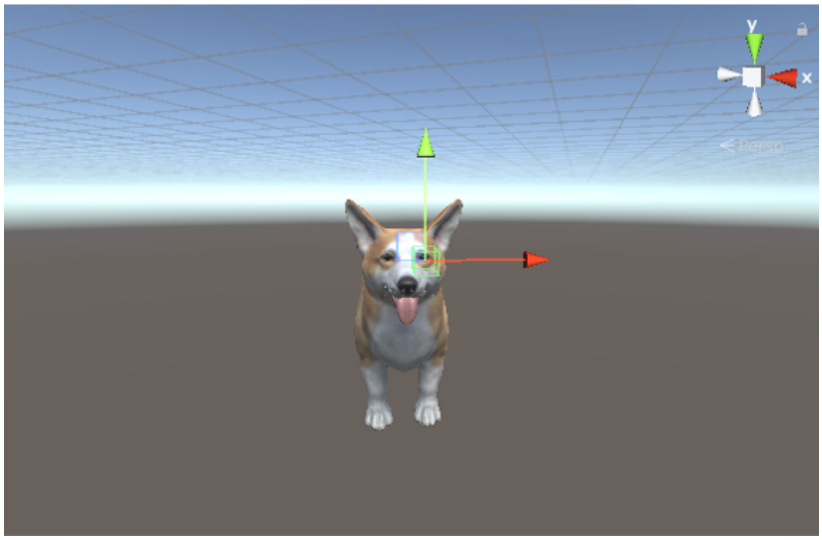

Fig. 5.8 3Dモデルの目へのコライダ設置

図5.9にアバタのユーザの正面までの移動に関するソースコードを示す。２つのコライダの衝突を検知した場合、ユーザの現在地座標から正面から0.5mをの座標を取得し、その座標まで移動し、喜ぶアニメーションに遷移するようにした。

```
void OnTriggerStay(Collider Collider)
{

    if (Collider.gameObject.tag == "Dog")
    {


        target.x = Camera.main.transform.position.x +
                                Camera.main.transform.forward.x * 0.5f;
        target.z = Camera.main.transform.position.z +
                                Camera.main.transform.forward.z * 0.5f;
        target.y = DogTr.transform.position.y;
        anim.SetBool("Walkbl", true);
    }
}
```

Fig. 5.9 ユーザの正面座標の取得

5.8 移動制御における視線の判定

図5.10に視線入力を判定するソースコードを示す。３Dモデルの座標移動、目的座標までの角度調節のため、目的地をtargetに定義した。unityシステムのmethodである、MoveTowarsを使用し、現在地から目的地の二点間の移動を可能とする。また、同様にmethodのRotateTowardsを使用し、二点間移動における角度調節を行う。 条件文で、目的地と現在地の二点間距離が0.01f未満すなわち限りなく近づいたとき3Dアバタの歩くアニメーションを終了するようにした。

```
//2点間を移動
transform.position = Vector3.MoveTowards(
            transform.position, target, speed * Time.deltaTime);
//2点間の角度を調節
transform.rotation = Quaternion.RotateTowards(
             transform.rotation, Quaternion.LookRotation(
                     target - transform.position), 120.0f * Time.deltaTime);

//目的地に着いたら歩くのをやめる
if (Vector3.Distance(transform.position, target) < 0.001f && comeback == true)
{
    moveornot = false;
    comeback = false;
    Gladstart();
```

Fig. 5.10 3Dアバタの移動

次に、ユーザが見ている座標を取得する。MRTKのEyeGazeprovidderを使用し、ユーザの視線と床との交点の目的地座標を取得する。図5.11に視線と床の交点の位置情報を格納するソースコードを示す。音声"行け"を認識した時点での視線と床との交点hitpositionをtargetに格納する。また図5.12に音声認識の設定を示す。音声認識には、MRTKのSpeechInputHandlerを使用し、特定のkeyword”いけ”の認識と音声によってトリガーとなるスクリプトをアタッチする。このtarget座標を図5.11のtargetに座標を返し、ユーザの任意の座標まで移動ができるようにする。

```
target = CoreServices.InputSystem.EyeGazeProvider.HitPosition;
```

Fig. 5.11 視線による目的地指定

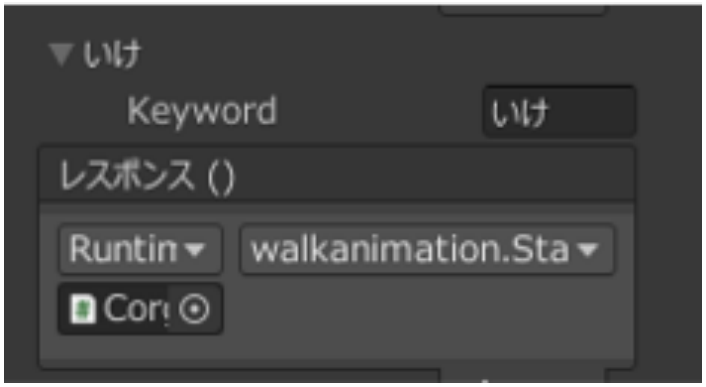

Fig. 5.12 SpeechInputHandlerによる音声認識

5.9 飛び移りシステムの構築

飛び移りに使用する障害物をここでは、一般的な椅子を選択する。Azure Custom vision Serviceを使用し、障害物の有無の判定を行う。4章の4.22で記述したように画像を、21枚用意し、学習をするため"chair"でタグ付けを行った。(図5.13)

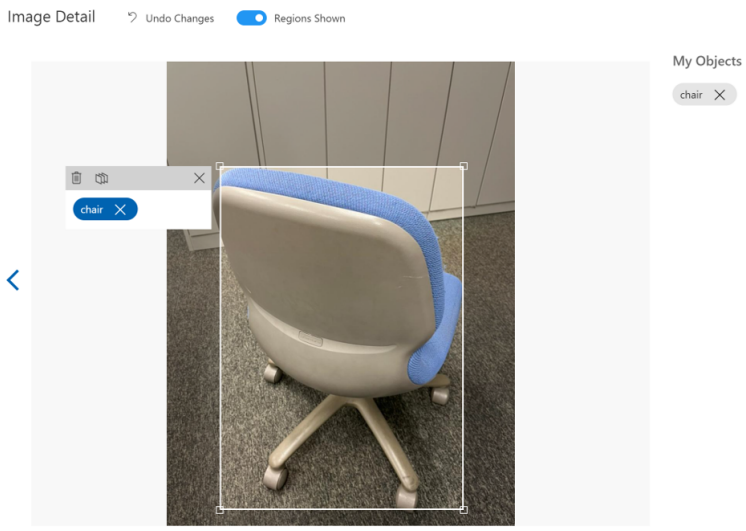

Fig. 5.13 椅子のタグ付け

タグ付けを行った画像をクラウド上で学習する。高い認識精度を伴う学習モデルとして成立させるためには，タグ付けを行なった画像の登録処理を繰り返す必要がある。図5.14は，異る椅子の画像 40 枚を準備し，それらの画像を全て登録

した後，Azure Custom Vision Service の学習モデルとしてトレーニングした結果である。

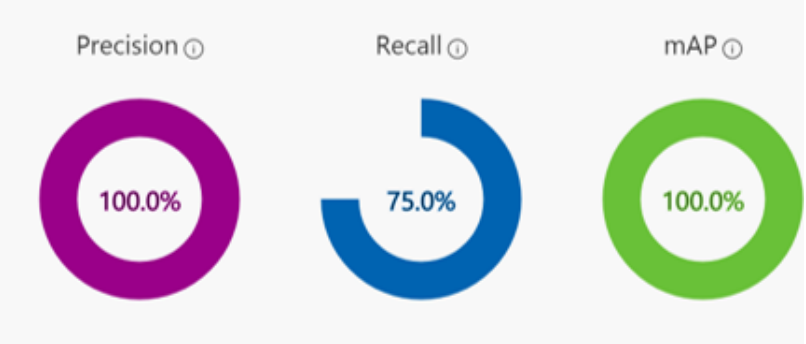

Fig. 5.14 画像検出学習結果

(1) Precision（適合率）
正解が含まれていると予測したデータの中で、実際に正解が含まれていた確率

(2) Recall（再現率）
正解が含まれる複数のデータの中から、実際に正解のデータを取り出す確率

(3) mAP（mean Average Precision, 平均適合率の平均）
モデル内に存在する複数のオブジェクトラベルを総合した Precision 平均値

Precision と Recall はトレードオフの関係にあり、適合率が高い場合は再現率が 低く、再現率が高い場合は適合率が低くなる傾向がある。Azure Custom Vision Serviceの閾値設定を使用し本システムでは適合率を優先している。また、Azure Custom Vision Service の WebAPIを使用するには，前述のトレーニング後に、モデルをパブリッシュ（発行）する必要がある。 Prediction URL および Prediction-Key を用いることで，WebAPI としてモデルへのアクセスが可能になる。Prediction URL および Prediction-Keyの設定ソースコードを図5.15に示す。

```
if (string.IsNullOrEmpty(trackedObject.CustomVisionTagId)
    || string.IsNullOrEmpty(sceneController.CurrentProject.CustomVisionIterationId))
{
    trackedObject.CustomVisionTagId =
        "[redacted]";
    sceneController.CurrentProject.CustomVisionIterationId =
        "[redacted]";
}
```

Fig. 5.15 WEB APIとunityの結合

本システムでは作成した学習モデルに対し，Microsoft HoloLensのビデオカメラ機能を使用し、3 秒に 1 回キャプチャしたユーザの視界画像を WebAPI を介してポストしている。判定結果が障害物と認識された場合、その座標を取得、また飛び移り命令が可能となる。図5.16にWEB APIへの送信と結果主取得のソースコードを示す。

```
yield return unityWebRequest.SendWebRequest();

string jsonResponse = unityWebRequest.downloadHandler.text
```

Fig. 5.16 WEB APIへの送信と結果取得

HoloLens2からキャプチャした画像を送信する。 WWWForm インスタンスを生成し、 PredictionKey と PredictionEndpoint を指定する。API への送信は、unitynetworkクラスのmethodであるSendWebRequestを使用し、また、結果の取得には、downloadHandlerを使用し、 JSON（JavaScript Object Notation）形式の戻り値を取得している。

次に、3Dモデルジャンプの高さを変化させるために、ジャンプアニメーションの最高点を調節できるようにする。アニメーションを時間単位で骨の座標を指定するソースコードを図5.17に示す。y座標を現在の座標に0.3mを加算することで、最高到達点を着地点の0.3m上げ飛び移り行動を可能とする。

```
[HeaderAttribute("match target")]
public Vector3 matchPosition;       // 指定パーツが到達して欲しい座標
public Quaternion matchRotation;    // 到達して欲しい回転


[HeaderAttribute("Weights")]
public Vector3 positionWeight = Vector3.one;
public float rotationWeight = 0;            // 回転に与えるウェイト。

private MatchTargetWeightMask weightMask;



Unity メッセージ|0 個の参照
override public void OnStateEnter(
            Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
{
    matchPosition.y = +0.3f;
    weightMask = new MatchTargetWeightMask(positionWeight, rotationWeight);
}

Unity メッセージ|0 個の参照
override public void OnStateUpdate(
            Animator animator, AnimatorStateInfo stateInfo, int layerIndex)
{
    animator.MatchTarget(
        matchPosition, matchRotation, targetBodyPart, weightMask, start, end);
}
```

Fig. 5.17 最高点の座標指定

target Body part をRootに設定し、Y座標における位置を加算するアニメーションの時間区間を選択する。図5.18にアニメーションの時間単位設定を示す。3Dモデルのジャンプアニメーションの最高到達点の開始時間0.012から終了時間0.37までの間Y座標のみ(0,1,0)の重みをつけて、高さを加算するよう設定する。また、この作業を0.37からアニメーション最終時間において加算しないように同様に設定することで着地点は元の座標にジャンプができるようにした。

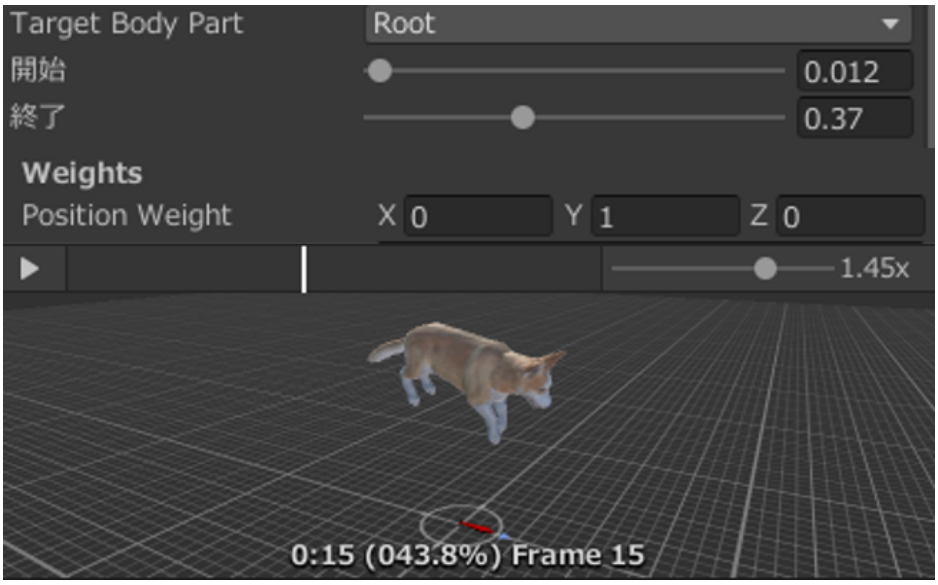

Fig. 5.18 アニメーションの時間区間と座標

MRTKのEyeGazeprovidder[30]を使用し、ユーザの視線と床との交点の目的地高さを取得する。音声"ジャンプ"を認識した時点での視線と障害物との交点hitpositionをtargetに格納する。音声認識”ジャンプ”をトリガーとし、3Dモデルは飛び移り行動を始める。

5.10 餌やり制御の判定

3Dモデルの口と手に衝突を検知するコライダを付加して視線との衝突を検知する。 口からユーザの手への視線のスライドを認識するため、各コライダにtag付け行い、tagをコライダの衝突により切り替わるようにした。 まず、3Dモデルの口付近にボックスコライダeat collを設定する。図5.19にコライダの設定位置を示す。

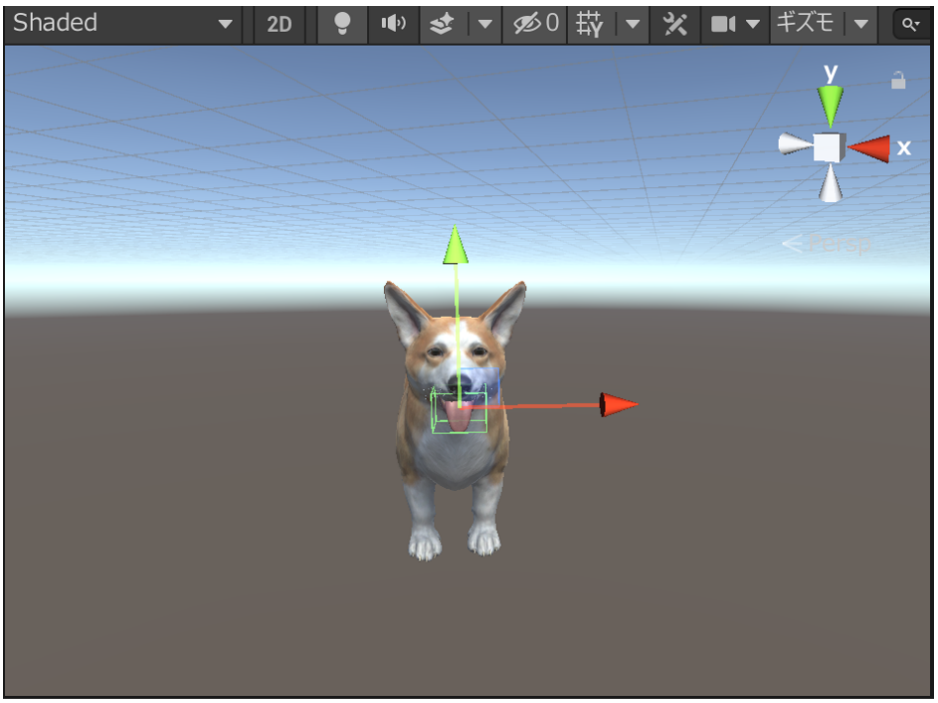

Fig. 5.19 3Dモデルの口へコライダ設置

次に、MRTkのJoint Hand TrackのRequestJointTranceformを使用し、右手の手のひらの座標を取得する。取得した座標データをオブジェクトの三次座標データに格納する。

```
var handJointService = CoreServices.GetInputSystemDataProvider<
                            IMixedRealityHandJointService>();

if (handJointService != null)

{

    Transform jointTransform = handJointService.RequestJointTransform(
                TrackedHandJoint.MiddleKnuckle, Handedness.Right);
    this.transform.position = jointTransform.position;

    print(jointTransform.position);


}
```

Fig. 5.20 ハンドトラックによる手の座標追跡

図5.21にフレームごとの位置データを示す。手を動かすと、ハンドトラックによって毎フレームごとの座標を三次元の座標データで取得する。 このような値を取得するスクリプトを右の手に常に追従する手のコライダ”handcoll”オブジェクトにアタッチし、手をコライダが常に追従するようにする。

Fig. 5.21 ハンド追跡の座標データ

口に追従するコライダと手に追従するコライダを視線のスワイプによって制御するため、コライダと視線をtag付けしグループ分けする。 図5.22にタグの設定を示す。eat collのtagを"mouth"に、hand collのtagを"hand"に設定する。 "tag"分けを行うと、同じtag同士のみ衝突を検知するので視線との衝突を管理できる。

Fig. 5.22 3Dモデルの口へコライダ設置

同様に、視線オブジェクトにもtag付けを行う。tag名は、"お座り"の音声を認識し"mouth"に固定される。 お座りをした状態で口元のeatcollに視線が衝突すな

わちコライダへの衝突を検知し、視線のtagを"hand"を切り替える。図5.23は、コライダによる視線"tag"の切り替えである。コライダの衝突による条件分岐により、eyecolorオブジェクトのtagが切り替えている。 この視線によるスライドを検知し、3Dモデルは食べるアニメーションへと切り替わる。

```
if (Collider.gameObject.tag == "mouth" && swipe == false)
{

    swipe = true;


    Debug.Log("しょうとつ1");
    eyecolor.gameObject.tag = "hand";

}


if (Collider.gameObject.tag == "mouth" && swipe == true)
{




    Eatstart();
    Debug.Log("しょうとつ2");
    eyecolor.gameObject.tag = "hand";

}
```

Fig. 5.23 視線によるコライダのtag変更

5.11 餌やりに伴う体型変化

アニメーションのイベントは、特定の時間に特定のスクリプトを読み込むことが可能である。3Dモデルの食べるアニメーションの食べ終わり付近にアニメーションのイベントを追加する。図5.24にアニメショーンのイベント設定を示す。0.95あたりにイベント青線を追加し、イベントの関数名をEndEatEventに設定する。EndEatEventを定義してあるスクリプトeatを対象オブジェクトにアタッチする。

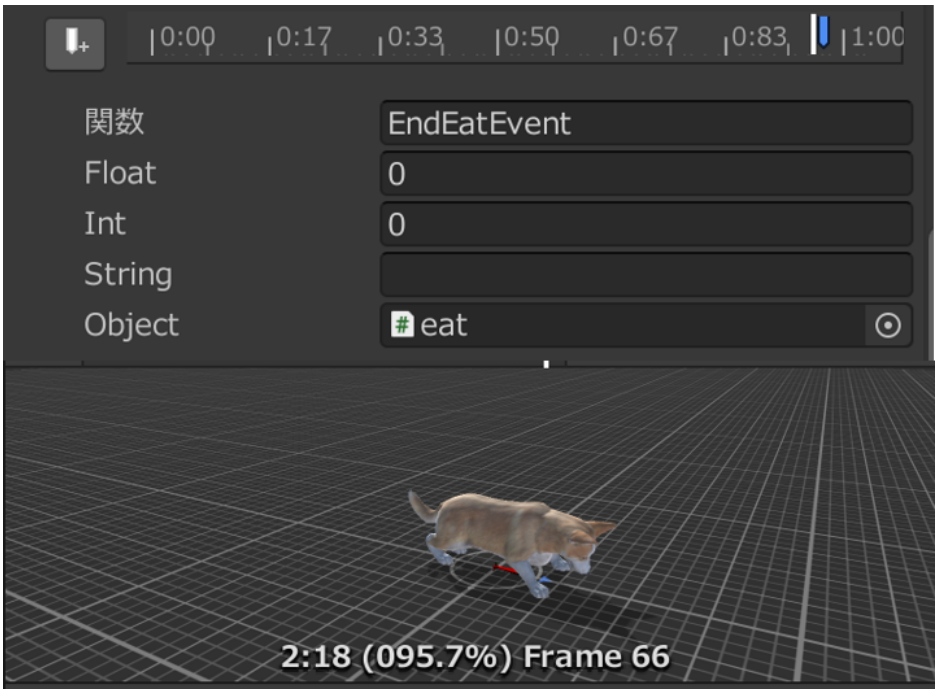

Fig. 5.24 アニメーション内のイベント追加

図5.25に3Dモデルのカウントアップソースコードに示す。アニメーションが読み込まれイベントが発生するごとに1ずつカウントアップされていく。3Dモデルのx軸すなわち3Dモデルを正面からみて横方向にカウント数に依存する増加を付与する。

```
public void EndEatEvent()
{
    count += 1;
    Debug.Log(count);

    Corgi_scale.x = Corgi_scale.x + count * 0.04f;
    Corgi.transform.localScale = Corgi_scale;
}
```

Fig. 5.25 食事によるカウントと体型増加

Chapter 6

予備的評価

6.1　参加者

システムの使いやすさと有用性を評価するために、20 代の男性9人の参加者によって評価実験とアンケートを行った。

6.2　評価方法

すべての参加者に、システムの簡単な紹介および Microsoft HoloLens の基本操作を説明した。参加者がデバイスに慣れた後、ARドックを制御してもらった。環境の要因でARペットの映像が乱れたり、視線制御が滑らかにできない等の問題も生じたため、実験後、デモビデオでアンケートを実施した。

アンケートには以下の 5 つの質問があり，1 から 5 までのグレーディング（1 = 非常にネガティブ，5 =非常にポジティブ）によって評価を行った。

(a) このシステムを使いやすいと思いますか？

(b) ARペットの動きは滑らかに感じますか？

(c) ペットに対して命令しているように見えますか？

(d) 視線を使用したARペットの制御は自然だと思いますか？

(e) 知人にこのシステムを紹介したいと思いますか？

6.3 結果

９人それぞれのデータとその平均値をとり、折線グラフにした。そのアンケート結果のグラフを図6.1に示す。

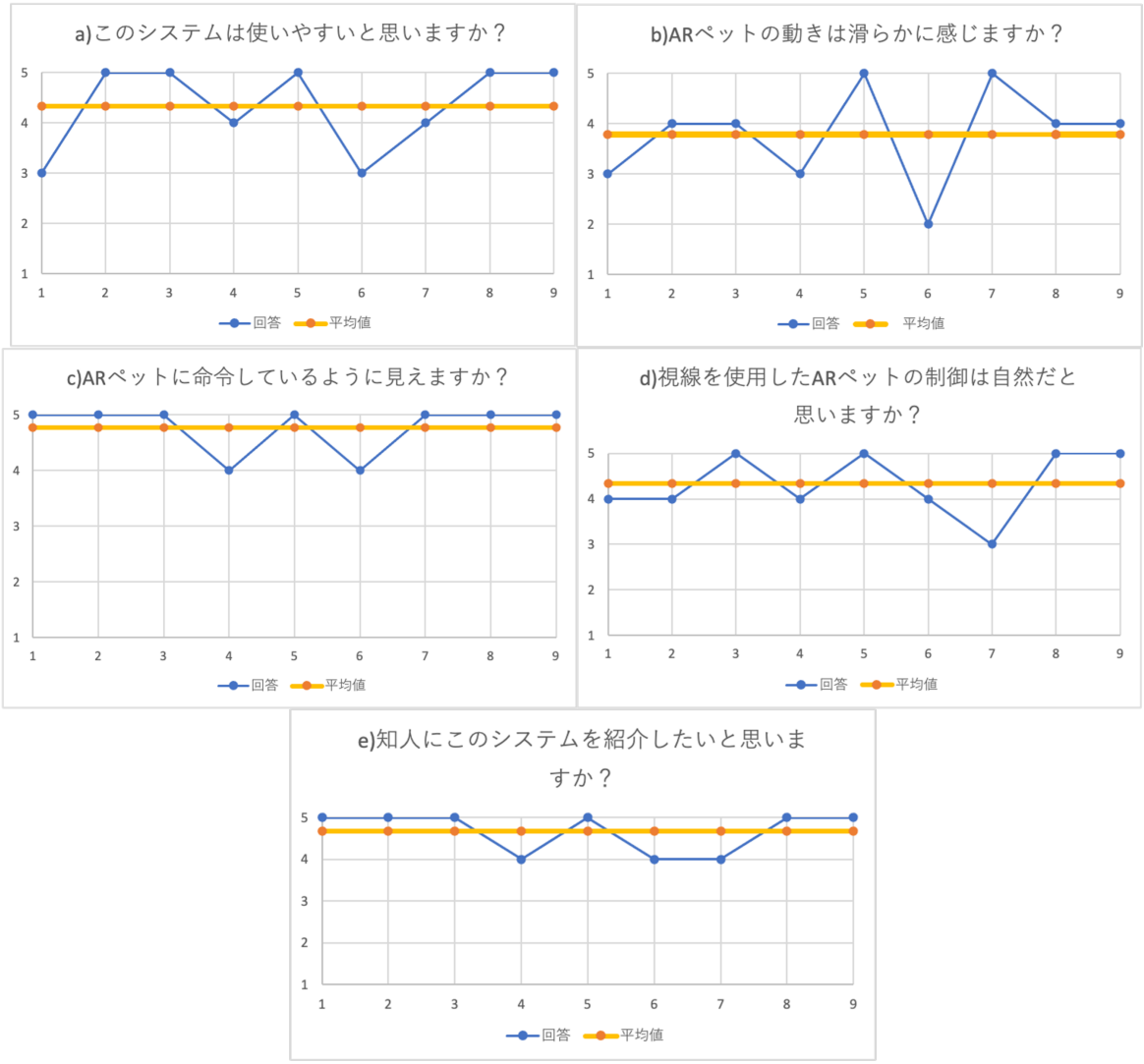

Fig. 6.1 アンケート結果グラフ

また、9人のアンケート結果をもとにそれぞれの設問の回答データの平均値と最大値、最小値、標準偏差を計算した。その結果を図６.２に示す。

	a	b	c	d	e
平均	4.333333333	3.777777778	4.777777778	4.333333333	4.666666667
最大値	5	5	5	5	5
最小値	3	2	4	3	4
標準偏差	0.866025404	0.971825316	0.440958552	0.707106781	0.5

Fig. 6.2 アンケート結果分析データ

図６.１を見ると、「ペットに対して命令している感覚はありますか？」という質問と、「知人にこのシステムを紹介したいと思いますか？」という質問に関しては、平均値4.5以上になり多くの参加者が高評価を示した。一方で、一方で、「ARペットの動きは滑らかに感じますか？」という質問の平均値は3.6で懸念を示す参加者が多くみられた。 図6.2の標準偏差を見ると「ARペットの動きは滑らかに感じますか？」という質問に対して平均からの標準偏差が離れているため、ばらつきがあるデータと言える。一方で「ペットに対して命令している感覚はありますか？」という質問と、「知人にこのシステムを紹介したいと思いますか？」に対しては、標準偏差が低く多くの参加者が同じような回答をしたと言える。

本研究では、飼い主がARペットに対して命令をする観点から研究をしているため、命令している感覚があるという点には一定の有用性があるといえる。 一方で、「ARペットの動きは滑らかに感じますか？」という質問には参加者でばらつっきがあり、懸念を示す参加者が多くみられた。特に、アニメーションの切り替えがぎこちない等の意見があり、詳細なアニメーション遷移を改善するなど、ARペットの動きが自然に見えるための必要性あることが示された。

Chapter 7

まとめ

本研究では、近年の疑似的なペットに着目し、飼い主ユーザの視点を中心としたペットとのコミュニケーションを拡張し、飼い主が命令することでARペットを実空間上で制御できる体験を提供した。ユーザの視線を追跡するため、HoloLensに搭載されている視線追跡技術を活用し、視線入力の判定とそれに対するARペットに行動を構築した。また、ARペットならではの従来のロボット型ペットや端末型ペットでは困難な行動を実装するため、ＡＲ技術を使用した実空間上で行動するARペットを構築した。 予備的評価では、ARペットに対して命令している感覚があるという質問に対して肯定的な評価が多く、本研究の飼い主の視線を使用したARペットの制御方法に一定の有用性があると言える。 今後の課題として、ARペットを行動アニメーションをスムーズにすることや、ロボット型ペットや端末型ペットでは困難な動きが可能なARペットを構築し、ARペットの有用性、新規性の拡張を目指す。

References

[1] 柴田崇徳, “メンタルコミットロボット「パロ」の開発と普及. 情報管理,” vol. 60, no. 4, pp. 217–228, 2017.

[2] 鈴木薫,金澤博史, “感情動因学習モデルを用いたペットロボット. 東芝レビュー,” vol. 56, no. 9, pp. 37–40, 2001.

[3] M. Fujita, “On activating human communications with pet-type robot aibo. IEEE,” vol. 92, no. 11, pp. 1804–1813, 2001.

[4] GROOVE X(株), “家族型ロボット"lovot"に迫る. 電学誌,” vol. 140, no. 7, pp. 442–1445, 2020.

[5] P. Julsaksrisakul, G. Chernyshov, M. Nakatani, B. Tag, and K. Kunze, “Nene: an interactive pet device.ubicomp ’17: Proceedings of the 2017 acm international joint conference on pervasive and ubiquitous computing and proceedings of the 2017 acm international symposium on wearable computers,” no. 7, pp. 89–92, 2017.

[6] “横井昭裕「変なことからメジャーが生まれる」https://fika.cinra.net/article/201901-yokoikarakawa,” 2019.

[7] “株式会社ユピテル.育てる猫型バーチャルペットjuno発売.https://prtimes.jp/main/html/rd/p/000000002.000075895.html,” 2021.

[8] “ネイロ株式会社.nintendo switch™ソフト「little friends –dogs cats-」発売.https://prtimes.jp/main/html/rd/p/000000008.000011837.html,” 2018.

[9] W. Liang, X. Yu, R. Alghofaili, Y. Lang, and L.-F. Yu, “Scene-aware behavior synthesis for virtual pets in mixed reality,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2021.

[10] N. Norouzi, K. Kim, M. Lee, R. Schubert, A. Erickson, J. Bailenson, G. Bruder, and G. Welch, “Walking your virtual dog: Analysis of awareness and proxemics with simulated support animals in augmented reality,” in *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 157–168, IEEE, 2019.

[11] M. WU, “Real-world ar pet: An emotional supporter and home guide.,” Master’s thesis, Waseda University. 2020.

[12] “Eyelink1000plushttp://sr-research.jp/products/eyelink_cl_series/,”

[13] V. Krishna Sharma, K. Saluja, V. Mollyn, and P. Biswas, "Eye gaze controlled robotic arm for persons with severe speech and motor impairment," in *ACM Symposium on Eye Tracking Research and Applications*, ETRA '20 Full Papers, 2020.

[14] A. N. Yumang, J. F. Villaverde, D. A. Padilla, and M. M. V. Gatdula, "Environmental control system for locked-in syndrome patients using eye tracker," in *Proceedings of the 2020 10th International Conference on Biomedical Engineering and Technology*, pp. 234–239, 2020.

[15] "Tobii pro fusionhttps://www.tobiipro.com/ja/product-listing/tobii-pro-fusion/,"

[16] "Tobii pro grasshttps://www.tobiipro.com/ja/product-listing/tobii-pro-glasses3/,"

[17] R. Rivu, Y. Abdrabou, K. Pfeuffer, A. Esteves, S. Meitner, and F. Alt, "Stare: Gaze-assisted face-to-face communication in augmented reality," in *ACM Symposium on Eye Tracking Research and Applications*, pp. 1–5, 2020.

[18] B. I. Outram, Y. S. Pai, T. Person, K. Minamizawa, and K. Kunze, "Anyorbit: Orbital navigation in virtual environments with eye-tracking," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, pp. 1–5, 2018.

[19] L. Sidenmark and H. Gellersen, "Eye&head: Synergetic eye and head movement for gaze pointing and selection," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pp. 1161–1174, 2019.

[20] X. Ma, Z. Yao, Y. Wang, W. Pei, and H. Chen, "Combining brain-computer interface and eye tracking for high-speed text entry in virtual reality," in *23rd International Conference on Intelligent User Interfaces*, pp. 263–267, 2018.

[21] J. M. Araujo, G. Zhang, J. P. P. Hansen, and S. Puthusserypady, "Exploring eye-gaze wheelchair control," in *ACM Symposium on Eye Tracking Research and Applications*, pp. 1–8, 2020.

[22] L. Sidenmark and A. Lundström, "Gaze behaviour on interacted objects during hand interaction in virtual reality for eye tracking calibration," in *Proceedings of the 11th ACM Symposium on Eye Tracking Research amp; Applications*, ETRA '19, 2019.

[23] C. Hennessey and J. Fiset, "Long range eye tracking: Bringing eye tracking into the living room," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pp. 249–252, 2012.

[24] A. T. Duchowski, S. Jörg, T. N. Allen, I. Giannopoulos, and K. Krejtz, "Eye movement synthesis," in *Proceedings of the ninth biennial ACM symposium on eye tracking research & applications*, pp. 147–154, 2016.

[25] "Hololens2/https://docs.microsoft.com/ja-jp/hololens/,"

[26] "mrtk開発者ドキュメントhttps://docs.microsoft.com/ja-jp/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05,"

[27] “specialmapping/understandinghttps://docs.microsoft.
com/ja-jp/windows/mixed-reality/mrtk-unity/features/
spatial-awarenessspatial-awareness-getting-started?view=mrtkunity-2021-05,”

[28] “microsoft azure custom visionドキュメントhttps://docs.microsoft.com/ja-jp/azure/
cognitive-services/custom-vision-service/,”

[29] “mrtkのInputDataProviderhttps://docs.microsoft.com/ja-jp/windows/mixed-reality/
mrtk-unity/features/input/speech?view=mrtkunity-2021-05,”

[30] “mrtkアイトラッキングドキュメントhttps://docs.microsoft.com/ja-jp/windows/
mixed-reality/mrtk-unity/features/input/eye-tracking/eye-tracking-main?view=
mrtkunity-2021-05,”

[31] “unity3dhttps://docs.unity3d.com/Manual/index.html,”

[32] “unity asset storehttps://assetstore.unity.com/?locale=ja-JP,”