右手を認識対象としたマーカーレス型WebAR の試作

右手を認識対象としたマーカーレス型 WebAR の試作

濵砂　桜太[a)]・坂本　眞人[b)]

Prototype of Markerless WebAR by Using Right Hand as Object of Recognition

Ota HAMASUNA, Makoto SAKAMOTO

Abstract

The author's laboratory had the opportunity to have people experience AR technology at a corporate event. However, our marker-based AR application had various problems. Therefore, we developed an AR system that solves those problems and conducted evaluations experments. Development was done by using Google's MediaPipe Hands to recognize the right hand and place objects in appropriate positions based on the 3D coordinates obtained. We also developed it as a WebAR that does not require the installation of a dedicated application. The evaluation experiment was conducted on a total of 20 participants, including university students, for the purpose of verifying the objective usability of this system, including object positioning and rotation. The results of the evaluation experiment showed relatively good results, with few subjects responding that they felt uncomfortable with the position and rotation of the objects, although improvements were found in the handling of postures that MediaPipe Hands could not recognize and in the revision of the 3DCG.

Keywords:　3DCG, Augmented reality, Markerless, MediaPipe, Vector, Web

1.　はじめに

著者の所属する研究室では、2022 年 7 月 30 日から 7 月 31 日にかけて開催された『第 12 回米良電機企業グループ統合展示会「宮崎の自然との共存」』（図 1）にて、来場者に AR 技術を体験してもらう機会があった。その展示会で我々は、Unity で開発したマーカー型 AR アプリケーション（図 2）を事前に 3 台のスマートフォンにインストールし、それらを貸し出す方法で体験してもらった。そのため、次のような問題点が発生した。

- 来場者のスマートフォンでは体験できない
- 貸し出すスマートフォンを使いまわすことによる新型コロナウイル感染症感染の危険性
- 貸し出すスマートフォンの充電
- マーカーの欠損、マーカーの設置場所の確保

これらの問題を解決するための方法として、平面や空間を認識対象としたマーカーレス型 WebAR が挙げられる。有名な事例として、Apple の公式サイト[1)]や LOWYA AR[2)]がある。これらは商品を AR 表示することで、現実世界でのサイズ感や色合いを確認することができる。しかし、いずれもスマートフォンの画面操作によるオブジェクトの移動や回転を行うため、展示会やイベントなどの娯楽を目的とした用途では没入感に欠ける。

そこで、本研究では、右手を認識対象としたマーカーレス型 WebAR の開発を行った。右手を認識対象とすることで、マーカー型のような現実のマーカーの動きにオブジェクトが追従する良さを残すことができる。

図 1.　第 12 回米良電機企業グループ統合展示会

図 2.　開発したマーカー型 AR アプリケーション

a)　工学専攻機械・情報系コース大学院生
b)　工学基礎教育センター教授

2. 開発手法・開発

2.1 本システムの概要

本研究では、Google 社の提供する MediaPipe Hands[3]を使用して右手の認識を行い、得られた 3 次元座標から表示するオブジェクトの適切な位置と向きを計算する。また、ユーザーが専用のアプリケーションのインストールを必要としない WebAR として開発を行う。

2.2 手を認識対象とした AR の先行研究

既存する、手と仮想物体とのインタラクション操作を実現するための研究の多くが、Leap Motion Controller などの外部デバイスを使用している。しかし、その一方で、加藤晴久（2012）の「てのひら AR」[4]では、てのひらとカメラの相対的姿勢及び位置を、カメラから得られた画像の指の線分を利用することで推定する手法を提案していた。さらに応用例として、てのひらの姿勢を変化させるだけで、直感的な操作で制御できるアプリランチャーの UI を開発していた。我々の研究においても、展示会やイベントを想定しており、スマートフォン 1 台だけで体験できる AR システムを実現したいため、非常に参考になる。

2.3 開発環境

本研究における開発環境及び、使用したライブラリなどを以下の表 1 に示す。また、ファイル構成を図 3 に示す。

表 1. 開発環境及び使用したライブラリなど

PC の OS	Windows 10 Pro
ソフトウェア	Blender 3.1.2
	Visual Stadio Code 1.74.3
テスト用端末	Redmi Note 9T（Android 11）
	iPad mini（第 5 世代）（iPadOS 16.3）
ライブラリなど	A-Frame
	AR.js
	MediaPipe Hands

```
├─index.html
├─script.js
└─assets
      └─sbutterfly.glb
```

図 3. ファイル構成

2.4 開発手順

本研究は、以下の手順に沿って開発を行った。

① Blender で 3DCG モデルを glTF 形式で出力
② HTML ファイルの作成及びコード編集
③ JavaScript ファイルの作成及びコード編集
④ GitHub Pages で公開、テスト用端末で動作確認

2.4.1 手順①

Blender で 3DCG モデルを glTF 形式（sbutterfly.glb）で出力。3DCG モデルは展示会で使用した自作の蝶（図 2）のうち 1 匹をそのまま使用する。以下の図 4 が Blender での蝶、図 5 が glTF 形式に変換した蝶である。

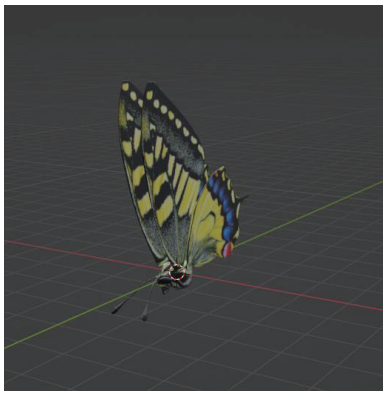

図 4. Blender での蝶

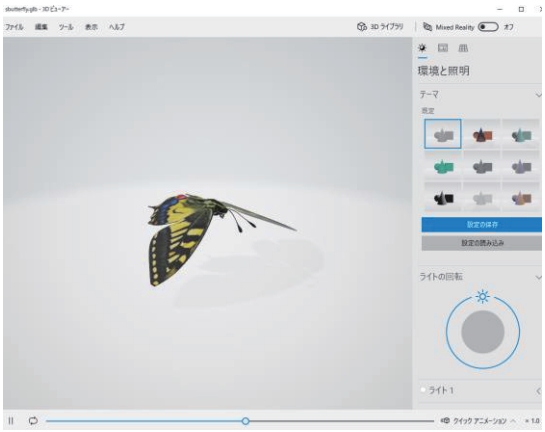

図 5. glTF 形式

この蝶の 3DCG モデルは、ポリゴン数が 17,468 で構成されており、10 フレームで羽を広げ、10 フレームで羽を閉じる、計 20 フレームのアニメーション設定を行っている。

2.4.2 手順②

Visual Studio Code で、HTML ファイル（index.html）の作成及びコード編集を行う。この HTML ファイルでは、表 1 で示した 3 つのライブラリなどの読み込み及び、A-Frame と AR.js における 3DCG モデルとカメラの設定を行う。

A-Frame[5]とは、2015 年に MozVR チームが開発したもので、3D/AR/VR 体験を構築するための、オープンソースの Web フレームワークであり、AR.js と連携させることで簡単に WebAR を実装できる。座標系は、スクリーンから自分へ向かう方向を正の Z とした右手座標であり、HTML と Entity-Component を用いて 3D ワールドを作成する。

まず、各ライブラリの読み込みは、head タグ内で CDN から読み込む。A-Frame を読み込む際は、Blender で設定したアニメーションを再生するために aframe-extras も追加で読み込む。

次に、A-Frame と AR.js における 3DCG モデル（蝶）とカメラの設定は、以下の図 6 に示すような親子関係で設定を行う。

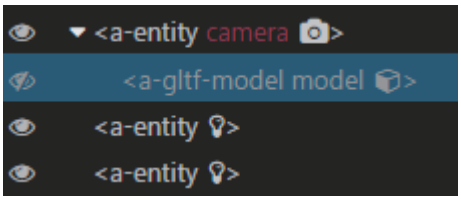

図 6.　3DCG モデル（蝶）とカメラの親子設定

3DCG モデル（蝶）は、MediaPipe Hands で取得した 3 次元座標を用いて表示するため、カメラの子として設定を行い、カメラを原点としたローカル座標で扱う必要がある。また、3DCG モデル（蝶）は、visible = “false”としておき、手を認識していない間は表示しないように設定を行う。

2.4.3　手順③

Visual Studio Code で、JavaScript ファイル（script.js）の作成及びコード編集を行う。この JavaScript ファイルでは、MediaPipe Hands から手指ランドマークの 3 次元座標を取得後、3DCG モデル（蝶）の適切な位置と向きを計算し、HTML の属性値の書き換えを行う。本研究では、3DCG モデル(蝶)が右手の人差し指に追従するように計算を行う。

MediaPipe Hands とは、Google 社が提供するオープンソースの機械学習ソリューションフレームワークである。Leap Motion Controller との性能比較は、先行研究の生野優輝、外村佳伸（2020）「手指ジェスチャー認識に向けた Leap Motion と MediaPipe の比較検討」[6]で検証されており、その研究結果を踏まえ、本研究では、MediaPipe Hands を使用する。MediaPipe Hands は、以下の図 7 に示す通り指先及び関節、計 21 個のランドマークの 3 次元座標を results.multiHandLandmarks という配列に格納し出力する。

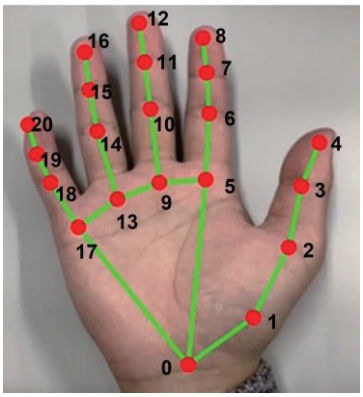

図 7.　MediaPipe Hands

MediaPipe Hands で得られる 3 次元座標における、*X* 座標と *Y* 座標は、画面左上を(0 , 0)、右下を(1 , 1)とする正規化されたスクリーン座標であり、*Z* 座標は手首（0 番）をほぼ 0 とした相対座標であり、*X,Y* 座標と同じような尺度で出力される。

本研究では、3DCG モデル（蝶）の位置を以下の計算で算出し、HTML の属性値を書き換える。

- *Z* 座標

前述した通り、MediaPipe Hands で出力される *Z* 座標は、手首を 0 とした相対座標で、カメラからの距離でないため、ワールド座標における正確な *Z* 座標を得ることができない。従って、本研究では、透視投影の距離が遠いほど物体が小さく見る特性を利用し *Z* 座標を決定する。計算に用いる座標は人差し指の MP 関節（5 番）と手首（0 番）の 2 点に絞り、1 フレーム前との線分比を用いる。

- *X* 座標

決定した *Z* 座標、A-Frame で設定した視野角、ブラウザ上で表示する領域のアスペクト比を用いて、MediaPipe Hands で得られたスクリーン座標をワールド座標へ変換する。算出方法は、以下の図 8 及び数式の通りである。

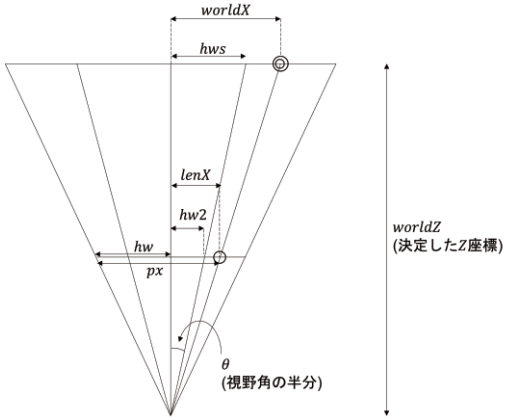

図 8　*X* 座標の算出方法

$$hws = worldZ \times \tan\theta$$
$$(本研究では、\theta = 20^{\circ})$$

$$hw = \frac{window.innerWidth}{2}$$
$$(window.innerWidth：ブラウザ上で表示する領域の横の長さ)$$

$$hw2 = \frac{hw}{asp}\left(※asp = \frac{window.innerWidth}{window.innerHeight}\right)$$
$$(window.innerHeight：ブラウザ上で表示する領域の縦の長さ)$$

$$px = x \times window.innerWidth$$
$$(x: \text{MediaPipe Handsで得た人差し指の}x\text{座標})$$

$$lenX = px - hw$$

$$\therefore worldX = lenX \times \frac{hws}{hw2}$$

● *Y* 座標

Y 座標も *X* 座標と同様に、決定した *Z* 座標、A-Frame で設定した視野角、ブラウザ上で表示する領域のアスペクト比を用いて、MediaPipe Hands で得られたスクリーン座標をワールド座標へ変換する。算出方法は、以下の図 9 及び数式の通りである。

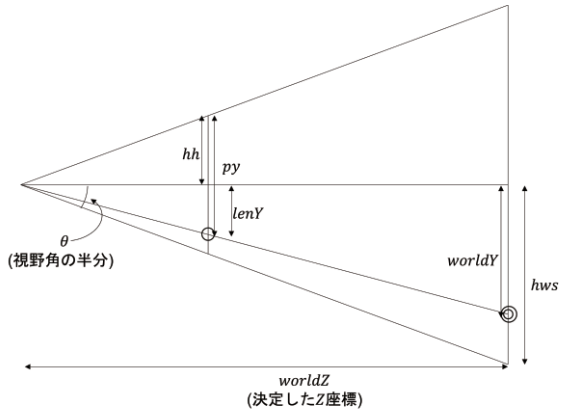

図 9.　*Y* 座標の算出方法

$$hws = worldZ \times \tan\theta$$
(本研究では、$\theta = 20^{\circ}$)

$$hh = \frac{window.innerHeight}{2}$$

$$py = y \times window.innerHeightlen$$

$$Y = py - hh$$

$$\therefore worldY = lenY \times \frac{hws}{hh}$$

次に、本研究では、3DCG モデル（蝶）の向きを、手首（0 番）・人差し指の MP 関節（5 番）・薬指の MP 関節（13 番）の 3 点からなる三角形（図 10）の法線ベクトルと、真正面へ向かうベクトル（0 , 0 , 1）との *Z-Y* 平面上での角度差を *X* 軸、*Z-X* 平面上での角度差を *Y* 軸の回転とし算出する。また、本研究では、回転を右手の無理のない動き（*X,Y* 軸ともに 180 度）に限定する。

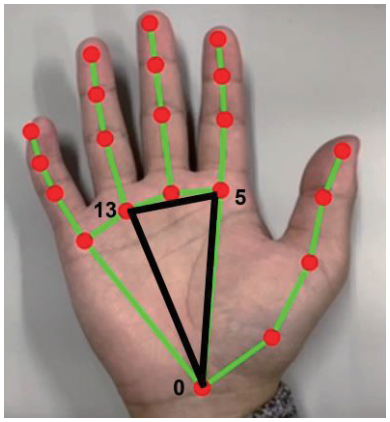

図 10.　計算で用いる 3 点

以下、計算式である。

● 定義

$point[i].j$: i番目のj座標　　(例　$point[0].x$: 手首のx座標)

$V013$: 手首から薬指のMP関節へ結んだベクトル

$V05$: 手首から人差し指のMP関節へ結んだベクトル

$Vnormal$: 算出した法線ベクトル

$Vnomal.j$: 算出した法線ベクトルのj成分 (jがxの時、x成分)

X: Z－Y平面上での角度差

Y: Z－X平面上での角度差

● 計算式

$$V013 = \begin{pmatrix} point[13].x - point[0].x \\ point[13].y - point[0].y \\ point[13].z - point[0].z \end{pmatrix}$$

$$V015 = \begin{pmatrix} point[5].x - point[0].x \\ point[5].y - point[0].y \\ point[5].z - point[0].z \end{pmatrix}$$

$$Vnormal = V013 \times V015 \quad (外積)$$

$$\cos X = \frac{Vnormal.z}{\sqrt{(Vnormal.z)^2 + (Vnormal.y)^2}}$$

$$\therefore X = \arccos X$$

$$\cos Y = \frac{Vnormal.z}{\sqrt{(Vnormal.z)^2 + (Vnormal.x)^2}}$$

$$\therefore Y = \arccos Y$$

2.4.4　手順④

GitHub Pages で公開し、テスト用端末で動作確認（図 11）を行う。

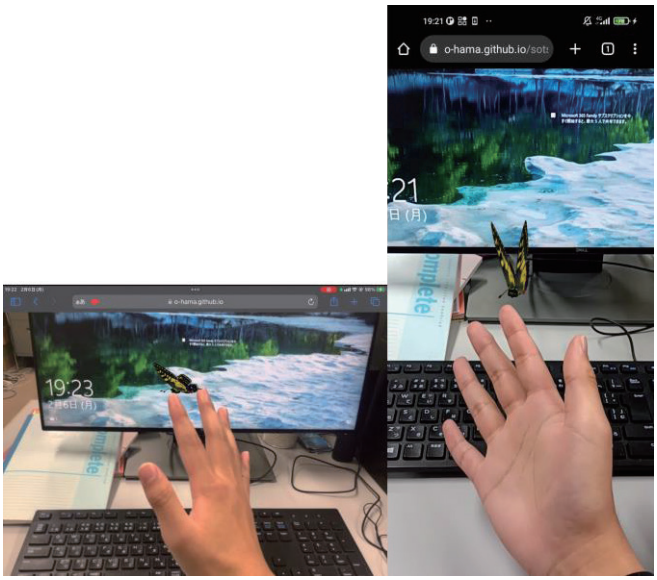

図 11.　動作確認（左：iPad mini、右：Redmi Note 9T）

動作確認は、iPad mini は Safari、Google Chrome、Firefox を、Redmi Note 9T は Google Chrome のブラウザを用いて行った。人差し指の指先に追従し、手のひらの向いている方向に 3DCG（蝶）が向くことを確認できた。

3.　評価実験

3.1　評価方法

3DCG モデル（蝶）の位置や回転を含む本システムの客観的な使用感を検証する目的で、小中学生 5 名、大学生 13 名、40 代の社会人 2 名の計 20 名を対象にアンケート式の評価実験を行った。小中学生の 5 名はテスト用端末の iPad mini（第 5 世代）を使用してもらい、他の 15 名にはそれぞれ所有している端末で行ってもらった。以下の表 2、表 3、図 12 は、その 15 名の実行環境である。

表 2.　端末

OS	端末名	人数
iOS	iPhoneSE	1
	iPhone11	1
	iPad Air（第 4 世代）	1
	iPhone12	5
	iPhone12 mini	1
	iPhone13	1
	iPhone13 Pro	1
Android	Galaxy S9	1
	Google Pixel 6a	1
EMUI	HUAWEI nova 5T	1
Windows	PC	1
合計		15

表 3.　OS のバージョン

OS バージョン	人数
iOS 15	2
iOS 16	9
Android 10	1
Android(バージョン記述なし)	1
EMUI 12	1
Windows 10	1
合計	15

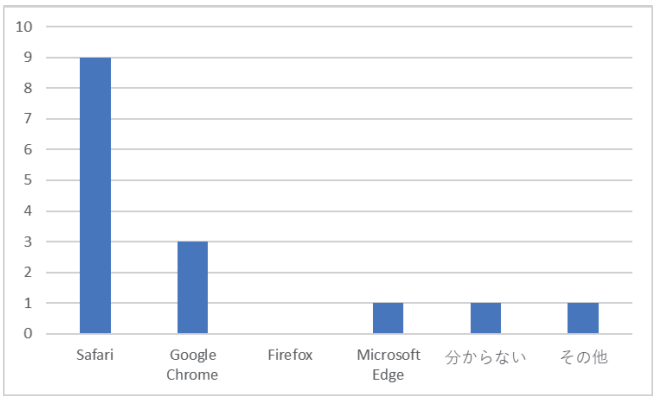

図 12.　使用したブラウザ

実行環境としては、iPhone12 の iOS16、Safari の組み合わせが最も多かった。

以下、選択式のアンケート内容である。それぞれ、最もあてはまると思ったものを選択してもらった。

- 蝶の動きはスムーズでしたか？
  - ✓ そもそも見ることができなかった
  - ✓ かなりカクカクしていた
  - ✓ カクカクしていた
  - ✓ 少しカクカクしていた
  - ✓ スムーズだった
  - ✓ とてもスムーズだった
- 手を止めたとき、人差し指の位置と蝶の位置にずれがありましたか？
  - ✓ そもそも見ることができなかった
  - ✓ かなりずれていた
  - ✓ ずれていた
  - ✓ 少しずれていた
  - ✓ ずれていなかった
  - ✓ 全くずれていなかった
- 蝶の回転に違和感がありましたか？
  - ✓ そもそも見ることができなかった
  - ✓ 回転しなかった
  - ✓ かなりあった
  - ✓ あった
  - ✓ 少しあった
  - ✓ なかった
  - ✓ 全くなかった

3.2 実験結果

以下の図13、14、15に結果を示す。

- 蝶の動きはスムーズでしたか？

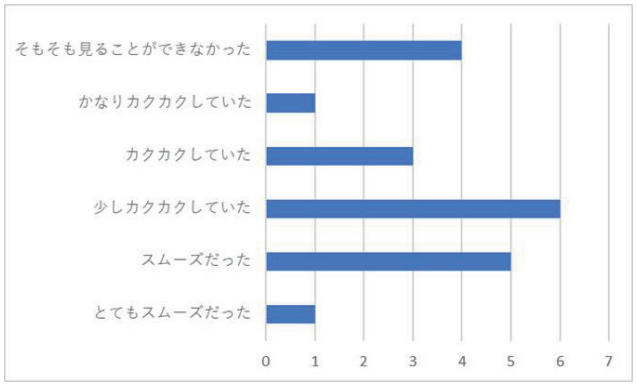

図13. 蝶の動きのスムーズさへの質問結果

- 手を止めたとき、人差し指の位置と蝶の位置にずれがありましたか？

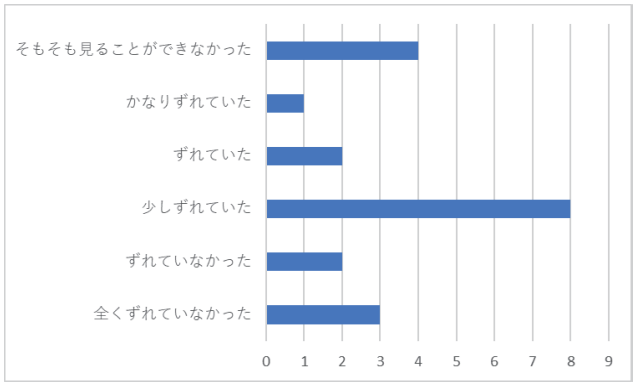

図14. 人差し指と蝶の位置への質問結果

- 蝶の回転に違和感がありましたか？

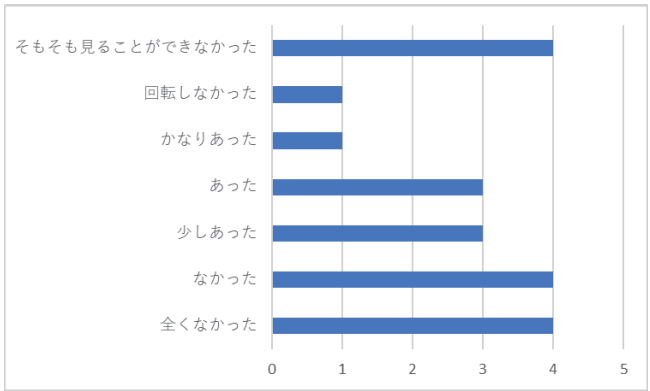

図15. 蝶の回転への質問結果

4. 考察

まず、3つの質問で「そもそも見ることができなかった」と回答があったのはそれぞれ4名であり、いずれも回答者が一致している。以下の表4にその4名の実行環境を示す。

表4. 見ることができなかったと回答があった実行環境

端末	OS	ブラウザ
HUAWEI nova 5T	EMUI 12	Google Chrome
iPhone13	iOS15.6.1	Safari
iPhoneSE	iOS15.5	Safari
iPhone12	iOS16	Safari

HUAWEI nova 5T の被験者は、本評価実験の被験者の中では、1名のみの端末及びバージョンであり、本システムが対応していなかったと考えられる。自由記述の欄に「そもそもカメラが起動しなかった」と回答があったためブラウザから端末のカメラへのアクセスが拒否された可能性が高い。

iPhone13、iPhoneSE の被験者は、2名とも OS のバージョンが iOS15 で、本システムは iOS15 で動作しないことが分かった。テスト用端末の iPad mini（第5世代）に関しても開発の初期段階の際は iOS15 だったため動作しなかった。iOS15 で動作しない原因は、判明していない。

iPhone12 に関しては、全く同じ環境の被験者が4名存在し、いずれも正しく動作しているため、システム上の原因はないと考えられる。

次にスムーズさへの質問は、「かなりカクカクしていた」「カクカクしていた」と回答があったのは合計で4名だけであり、比較的良い結果が得られた。しかし、「少しカクカクしていた」が6名で最も多く、スムーズに動かない原因としてはMediaPipe Handsのfpsが挙げられる。MediaPipe Handsは、動作環境によっては、スムーズに動かすために十分なfpsが出せない場合がある。

次に、人差し指と蝶の位置への質問は、「かなりずれていた」「ずれていた」と回答があったのは合計で3名だけであり、比較的良い結果が得られた。しかし、「少しずれていた」と回答があったのは8名で最も多かった。考えられる原因として挙げられるのは、カメラのレンズの歪みを考慮していないことである。レンズの特性上画面中央から離れるほど歪みが強くなり、実際の座標と画面に表示される座標では違いがうまれる。よって、本システムは端末のレンズの歪みによって位置のずれ方が大きく変化してしまう可能性が高いと考えられる。

最後に、蝶の回転への質問は、「回転しなかった」「かなりあった」「あった」「少しあった」と回答があったのは合

計で8名であり、「なかった」「全くなかった」は合計で8名であったため、おおよそ半々の結果となった。考えられる原因として挙げられるのは、MediaPipe Handsが認識しづらい手の姿勢があり、回転途中で蝶が消えてしまうことである。特に、蝶の真横を見るためには、手を横にする必要があり、指が重なってしまうため21個の指先及び関節ランドマークが認識しづらくなってしまう。途中で蝶が消えることは回転の違和感に大きく影響すると考える（図16）。

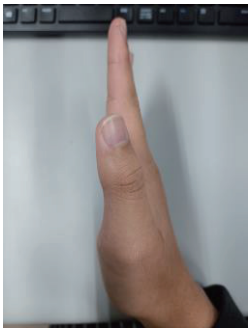

図16.　MediaPipe Handsの苦手な姿勢

5.　結論

本研究では、以下のマーカー型 AR アプリケーションの問題点を解決する目的で、右手を認識対象としたマーカーレス WebAR を開発し、評価実験を行った。

- 来場者のスマートフォンでは体験できない
- 貸し出すスマートフォンを使いまわすことによる新型コロナウイル感染症感染の危険性
- 貸し出すスマートフォンの充電
- マーカーの欠損、マーカーの設置場所の確保

しかし、評価実験の結果から、対応していない実行環境への対応、MediaPipe Hands の fps 及び苦手な姿勢への対応、レンズの歪みへの対応などの改善点が挙げられ、十分に目的を達成することができなかった。しかし、正しく動作した環境下での客観的な使用感としては、「少しずれていた」などの回答者数が多く、全体を通して見ると、比較的良い結果が得られたため、目的を十分に達成するための可能性を示唆することができたと考える。

6.　今後の研究

今後の研究としては、前述した改善点への対応を行いたい。特に、展示会やイベントなどにおいて、対応していない実行環境が多く存在することは致命的であると考える。

また、本システムでは3DCGモデルを単に表示するだけであったため、新たに、手を使ったシミュレーションなどができるようなARシステムの開発にも挑みたい。

7.　謝辞

本研究にあたり、ご支援およびご協力頂いた宮崎大学ならびに米良電機産業株式会社の皆様に心より感謝の意を表する。

参考文献

1) https://www.apple.com/jp/iphone,（accessed 2023/4/23）
2) https://www.low-ya.com/features/ar,（accessed 2023/4/23）
3) https://google.github.io/mediapipe/solutions/hands.html,（accessed 2023/4/23）
4) 加藤　晴久:てのひら AR, 映像情報メディア学会誌, Vol.66, pp.866-871, 2012.
5) https://aframe.io,（accessed 2023/04/24）
6) 生野　優輝, 外村佳伸: 手指ジェスチャー認識に向けた Leap Motion と MediaPipe の比較検討, 2020 年度 情報処理学会関西支部 支部大会 講演論文集, 2020.