令和 5 年度

修　士　論　文

AR ヘッドマウントディスプレイによる
空間マッピングを用いた温度可視化ツールの作成

情報科学プログラム　計算機基礎学研究室
M221157　　穂高　正

指導教員

教授　　中西　透
教授　　近堂　徹
准教授　　北須賀　輝明
准教授　　今井　克暢　(福山大学)

令和 6 年 2 月 6 日

広島大学大学院先進理工系科学研究科

あらまし

本研究では, Microsoft 社の AR HMD デバイスの Hololens 2 を用いて, 空間内の温度を可視化するツールを作成した. 可視化の更新速度や温度の解像度の低さを改善するため, 新たに高精度かつ高視野角で温度を取得できる赤外線アレイセンサを導入し, ユーザの視野の移動に応じた可視化を実装した. また, 空間マッピングの機能を用いて周囲の空間やオブジェクトの表面メッシュを取得し, 表面メッシュに対して温度テクスチャを投影テクスチャマッピングで投影し, 表面メッシュに合った温度可視化ツールを構築した. さらに, 温度可視化ツールのユーザビリティを向上させるため, HandMenu を用いてユーザーが自由に可視化の開始や終了, カラーマップの透明度を変更可能する機能も追加した.

目次

図目次

表目次

第 1 章

はじめに

AR ヘッドマウントディスプレイ (HMD) とは, 現実世界に仮想情報を重ねて拡張させるデバイスであり, Hololens 2[1] や Magic Leap 2[2] のような製品がある. また, 空間内の温度を可視化させるシステムやプロジェクトとして環境ウォッチ [3] やコロナ予防システム [4] があり, 感染予防や建設などの様々な場面で役立てられている. さらに, 関連研究として, 「環境 3D モデルによるカメラトラッキングを用いた AR 表示システム」[5] や「室内温熱環境設計フィードバックのための CFD と AR の統合」[6] がある.

しかし, これらの製品や研究では, 空間的な温度の視覚化において視認性が悪く, ユーザが物理的な環境における温度変化を十分に把握しにくい. そこで本研究では, 空間マッピングと投影テクスチャマッピングを組み合わせて空間内に温度を関連付け, 空間温度の視認性を向上させる温度可視化ツールを構築した.

まず, 赤外線アレイセンサ [7] という温度センサを用いることで物体等が放出する赤外線を検知し, 表面温度データを取得する. この温度データを空間で可視化するためには立体的に表示させる AR HMD が必要である. 本研究では, AR HMD デバイスの Hololens と温度センサの制御や AR デバイスへのデータ送信の役割として Raspberry pi[8] を用いた. 卒業研究 [9] では, Hololens2 と Raspberry pi, 簡易的な赤外線アレイセンサを用いた温度可視化ツールを作成した. しかし改善点として, 可視化の更新速度や温度の解像度の低さが挙げられた. そこで, より高精度かつ高視野角で温度を取得するため, 新たに赤外線アレイセンサとして MLX90640[10] を導入した. Hololens で受信したデータをもとにユーザの視線先に応じた可視化実装を行った. また, この温度データを空間内で関連付け視認させるために, 空間マッピングと投影テクスチャマッピングを用いた可視化実装を行い, Hololens で動作させて評価した.

第 2 章

事前知識

2.1　Hololens 2

Hololens 2 は Microsoft 社が開発した AR 型の HMD である. 図 2.1に Hololens2 の写真を示す. Hololens では図 2.2に示すように, 架空の 3D オブジェクトや UI を現実空間に組み合わせて自由に表示や操作をすることができる.

図2.1　Hololens 2

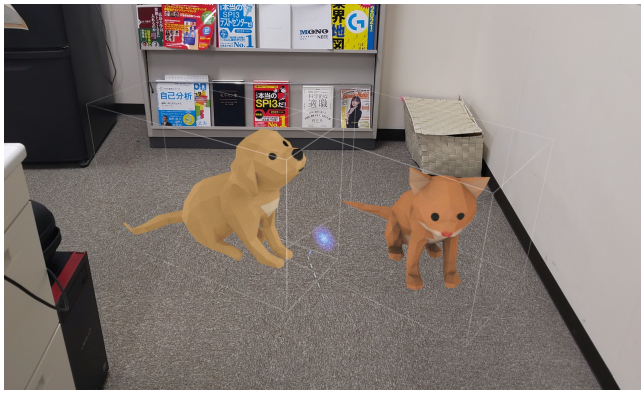

図2.2　3D オブジェクトの表示

また, 空間認識が搭載されており, 図 2.3のように表面メッシュを表示することも可能である. さらに, UI コンポーネントがあり, 図 2.4のようにボタン等でオブジェクトを操作することができる.

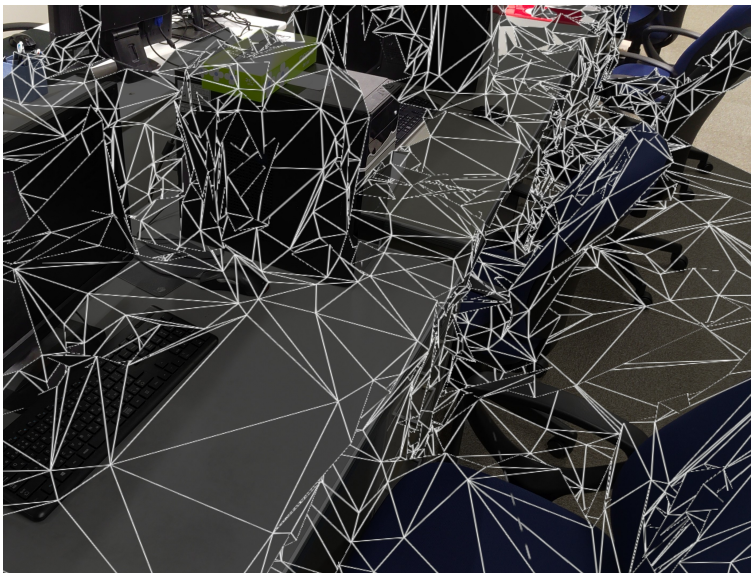

図2.3　空間認識によるメッシュの表示

図2.4　ボタン操作

2.2 Raspberry pi

Raspberry pi とは, イギリスのラズベリーパイ財団によって開発されたワンボードコンピュータである.Raspberry pi を用いることで, 温度センサとのデータの送受信が可能となる.

本研究で Raspberry pi を用いた要因として, 以下が挙げられる.

- 低消費電力
  低電力で動作するため, サーバー等の用途で稼働させるのに適している.
- 拡張性
  GPIO ピン, USB ポート等があり, 様々なハードウェアに接続して柔軟に拡張できる.
- 汎用性
  Python や Javascript 等の様々なプログラミング言語やアプリケーションに対応しており, 多くの用途に活用可能である.

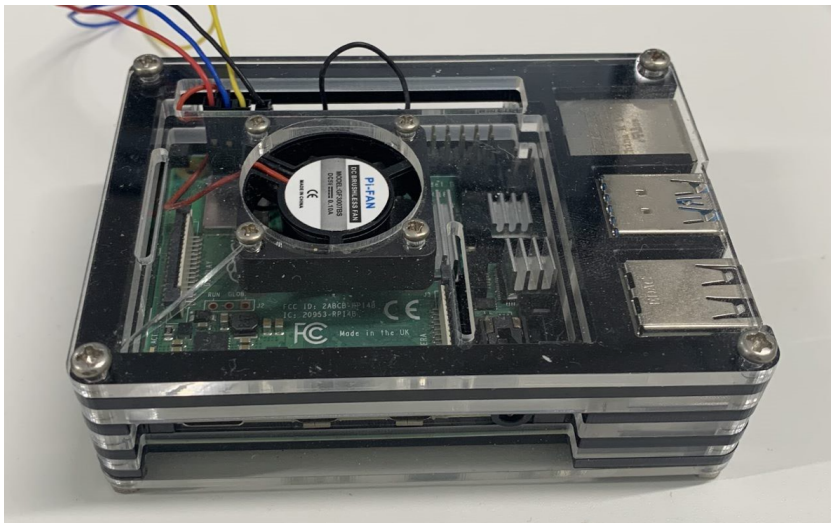

図2.5　Raspberry pi

2.3　赤外線アレイセンサ

赤外線アレイセンサとは, 物体から出る赤外線を検出し温度分布を検知するセンサであり, 非接触での温度測定等に用いられる. 図 2.6に赤外線アレイセンサの仕組みを示す.

本研究では赤外線アレイセンサとして MLX90640 を用いる. 卒業研究で用いたセンサ SMH-01B01[11] に比べ, 32×24 のピクセルアレイで視野角が 55 度 ×35 度であり, 短距離で高解像度かつ高視野角での温度測定に最適である. 図 2.7に MLX90640 センサを示す. また, 表 2.1に MLX90640 と SMH-01B01 の 2 つのセンサの比較を示す.

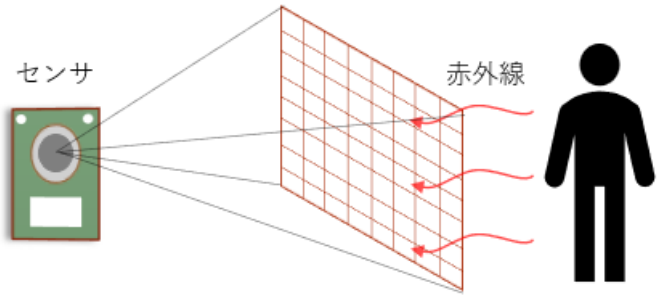

図2.6　赤外線アレイセンサの仕組み

図2.7　MLX90640 - FOV 55°

表2.1　センサの比較

	SMH-01B01	MLX90640
解像度	8 × 8 (128byte)	32 × 24 (1536byte)
視野角	35 × 35	55 × 35
測定距離	1m	数 m

2.4　I2C 通信

I2C 通信 [12] とは, 同期式シリアル通信の一つで, クロック用のシリアルクロック (SCL) とデータ用のシリアルデータ (SDA) の二つの信号線を用いて通信を行う. マスタ側とスレーブ側で双方向でデータの送受信を行い, 一つのマスタで複数のスレーブのデバイスと通信することが可能である. I2C 通信を用いることで, センサから温度データ取得が可能となる. 以下の流れでマスタはスレーブからデータを受信することができる.

1. マスタが通信を開始
2. マスタがスレーブアドレスを指定
3. マスタがスレーブからデータを受信
4. マスタが通信を終了

また, I2C 通信の仕組みを図 2.8に示す.

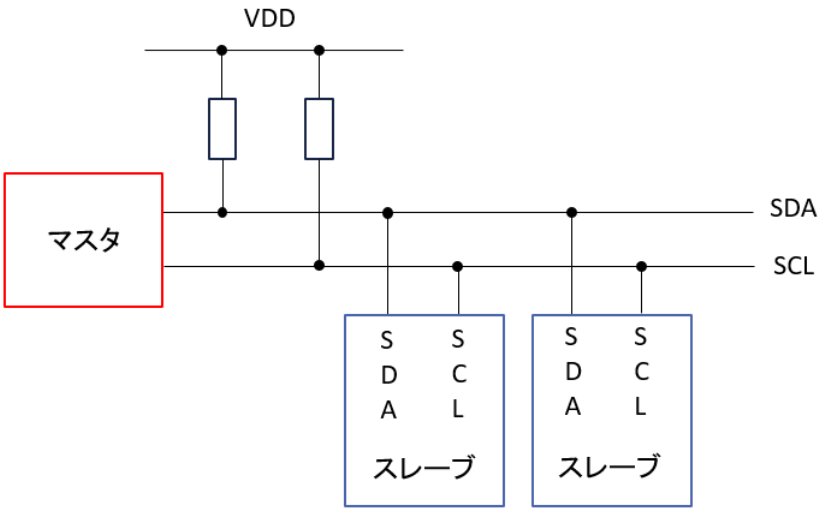

図2.8　I2C 通信の仕組み

2.5 WebSocket 通信

Raspberry pi と Hololens 間の通信として卒業研究では BLE(Bluetooth Low Energy) 通信を扱っていたが, 新たに WebSocket 通信 [13] を用いた. WebSocket 通信とは, クライアントとサーバ間でリアルタイムに双方向通信を行うための規格であり, クライアントから WebSocket 要求を送り, コネクションを確立 (ハンドシェイク) を行うことで双方向通信が可能となる.

本研究で WebSocket 通信を用いた要因として, 以下が挙げられる.

- リアルタイム性
  サーバはクライアントに好きなタイミングで即座にデータが送信できる.
- データ量
  オーバーヘッドの低減によりデータを効率的に転送可能である.
- コネクション維持
  コネクションが確立すると維持されるため, データの継続的な送信に適している.

また, WebSocket 通信の流れを図 2.9に示す.

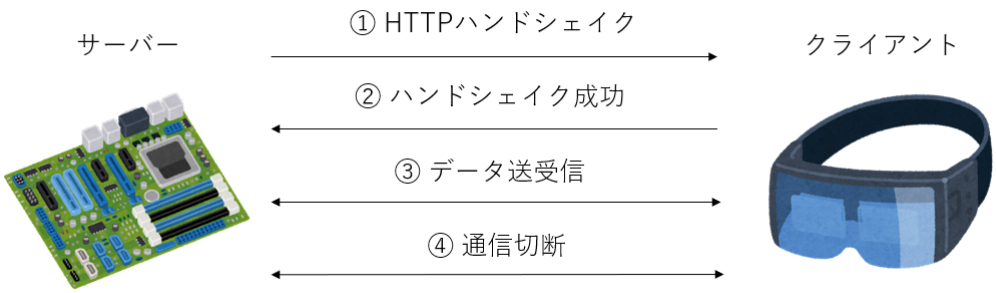

図2.9　WebSocket 通信の流れ

第 3 章

視野の移動に応じた温度の可視化

3.1　実装概要

実装の全体像として以下の図 3.1に示す. 温度センサのデータを I2C 通信で取得し, その温度データを WebSocket 通信で送信する.Hololens で温度データを受信し, 可視化制御を行うことで温度データを可視化することができる.

卒業研究では Hololens に搭載されている Bluetooth を用いることで Raspberry pi と Hololens を繋いでデータの送受信を可能としていたが, 新たに導入するセンサのデータ量の増加やリアルタイムな通信を考慮するため, 新たに WebSocket 通信を用いて可視化実装を行った.

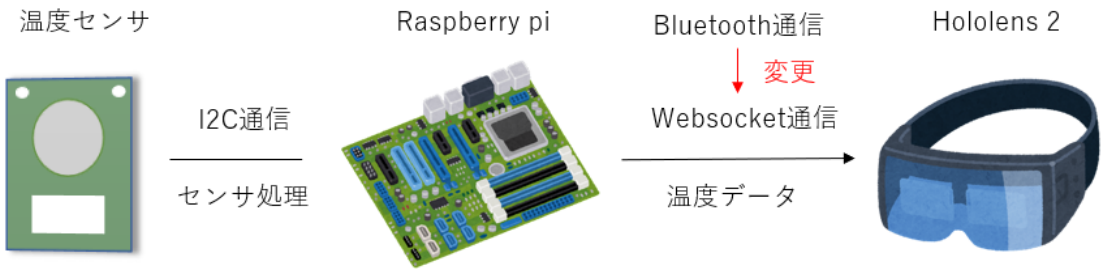

図3.1　視線先の可視化概要

3.1.1　カラーマップへの変換

Hololens で受信した温度データを可視化するためには, カラーマップに変換することで温度表示を行うことが可能となる. 以下の流れで受信した温度データからカラーマップに変換する. 図 3.2はカラーマップの変換ログで, 図 3.3はカラーマップを可視化した例である.

1. 温度データの最小値と最大値を取得し, データの範囲を特定
2. 温度データの各要素に対して, 最小値と最大値を考慮した 0 から 1 の範囲に正規化
3. 0 に近い場合は青に、1 に近い場合は赤に近い色を割り当てる
4. カラーマップに変換した配列を返す

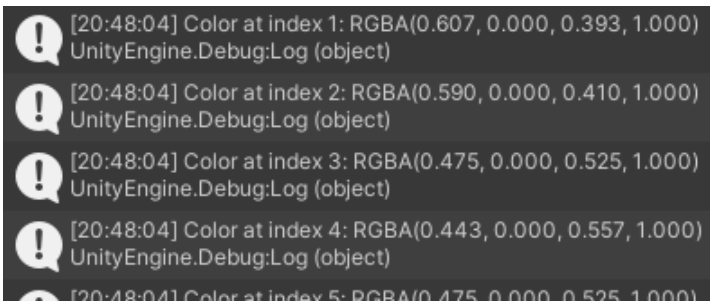

図3.2　生成されたカラーマップのログ出力

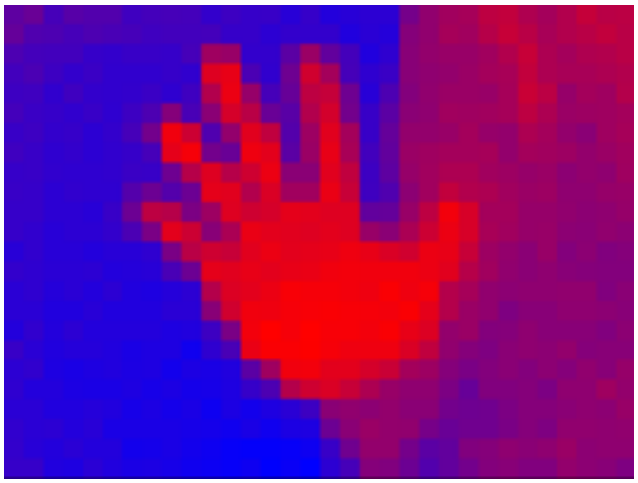

図3.3　カラーマップの可視化

3.1.2　データ通信の遅延考慮

赤外線アレイセンサと Hololens は直接的に通信が行えないため, I2C 通信と WebSocket 通信を介して間接的に温度データの送受信が行われている. つまり, Hololens で温度データを取得したときの視線先に温度表示オブジェクトを配置して可視化させると, 実際にセンサで温度測定した場所と可視化させた場所にずれが生じてしまう. WebSocket はリアルタイムに通信が可能であり, I2C 通信でずれが生じているため, この通信の遅延を考慮した可視化を行う.

卒業研究では, 以下の流れで一定時間の遅延を考慮していた.

1. Hololens のカメラセンサの位置情報の履歴をこまめに取得しておく
2. Hololens で温度データを取得したときに一定時間前の視線先に温度表示オブジェクトを移動させ, 正しい位置に可視化を行う

しかし, この方法では I2C 通信での温度データ取得時間に依存するため, 正しい位置に温度が表示できなくなる場合があった.

そこで, 本研究ではタイムスタンプ [14] を活用して遅延の考慮を行った. タイムスタンプとは, 特定のイベントやデータポイントが発生した時刻や日付を表す情報であり, イベントの時間的な順序を記録し, データの時間情報を管理するために使用される. 具体的な遅延の考慮は, 以下の流れで行った. 図 3.4は受信した温度データとタイムスタンプのログ出力である.

1. Raspberry pi で, 温度データを I2C 通信で取得する際に, time 関数を使って温度データ取得時刻も取得し, それをタイムスタンプとして温度データに追加して送信する
2. Hololens では受信したデータを温度データとタイムスタンプに分割する
3. Hololens のカメラセンサの位置情報と時刻をこまめに取得しておき, 分割したタイムスタンプと比較して正しい位置に温度表示オブジェクトを配置する

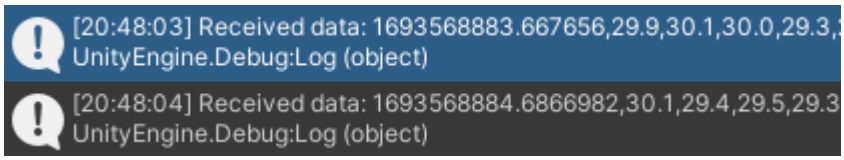

図3.4　タイムスタンプと温度データのログ出力

3.1.3　データ送信プログラムの自動化

Raspberry pi で温度データを送信する際に, 温度送信のプログラムをターミナルで必要に応じて実行する必要があった. そこで, Raspberry pi の電源起動時に Python プログラムを自動起動させることによって, データの自動送信が可能になる. 本研究では crontab でプログラムを自動起動する方法を用いた. crontab とは, Unix で使用される定期的なタスクスケジューラ cron を設定するコマンドである. crontab で起動してから一定時間後にプログラムを起動する際には以下の形式で crontab エントリに記述する必要がある.

```
@reboot sleep 15 && python /path/mlx90640_send.py
```

上記は, 起動後 15 秒後に指定したコマンドが自動起動するように設定しており, path に存在する mlx90640_send.py を実行するように指定している. Raspberry pi がネットワークに繋がった状態でないとデータの取得と送信が行えないため起動して一定時間後にデータ送信プログラムを起動する設定にしている.

3.1.4　サーバの IP アドレスの自動取得

WebSocket 通信ではクライアントはサーバとの接続を確立するためにサーバの IP アドレスが必要となる. Raspberry pi では ifconfig コマンドによって自身の IP アドレスを取得することができるが, ネットワークの変更等により IP アドレスが変更されることがある. そこで, IP アドレスを自動取得することによって IP アドレスを確認せずに WebSocket 通信でデータの送受信ができる. 具体的には以下の流れで行った.

- Raspberry pi 側
  subprocess.run[15] という外部プロセスを実行するための標準ライブラリを用いて ifconfig コマンドを実行して IP アドレスを自動取得する
- Hololens 側
  SSH.NET[16] という Unity のライブラリを用いて, Raspberry pi に SSH 接続を行う.SSH クライアントの作成後、ifconfig コマンドで IP アドレスを自動取得する (図 3.5参照)

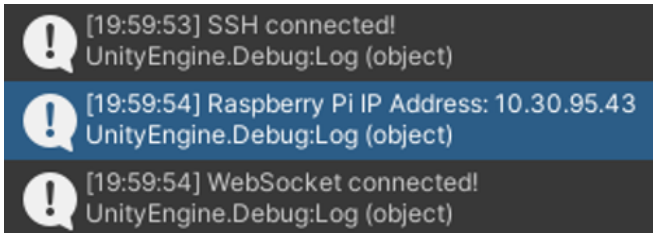

図3.5　IP アドレスの自動取得のログ出力

3.2 開発環境

Raspberry pi での開発環境を表 3.1に示す.

表3.1 Raspberry pi での開発環境

開発言語	Python
コンピュータ	Raspberry pi 4 Model B
ライブラリ	Adafruit_MLX90640 websockets

MLX90640 のデータを取得するためのライブラリとして Adafruit_MLX90640[17] と, WebSocket 通信で取得したデータを送信するためのライブラリとして websockets[18] を用いた.

Hololens 2 での開発環境を表 3.2に示す.

表3.2 Hololens 2 での開発環境

開発言語	C#
開発エンジン	Unity 2021.3.31f1
ライブラリ	Mixed Reality Toolkit v2.8.3 websocket-sharp

Hololens でのオブジェクトや UI の操作を効率的に行うために Mixed Reality Toolkit[19] というライブラリと, WebSocket で温度データを受信するライブラリとして websocket-sharp[20] を用いた.

3.2.1 Adafruit_MLX90640

Adafruit_MLX90640 は, Adafruit という企業が提供している赤外線温度センサモジュール MLX90640 を操作するための Arudino 用のライブラリである. 本研究では Python で MLX90640 のデータを取得するため, Adafruit_CircuitPython_MLX90640[21] という Python で簡単に MLX90640 のデータを取得できるライブラリを用いた.

3.2.2 websockets

websockets は, Python で WebSocket が扱えるライブラリである.WebSocket のサーバとクライアントの両方を構築することができ, 簡潔かつ柔軟性の高い機能や非同期通信, SSL/TLS のサポートが特徴のライブラリである. 今回はサーバとして利用しており, サーバでデータをクライアントに送信するコードをコード 3.1に挙げる.

コード 3.1　サーバ側コード

```
1  import asyncio
2  import websockets
3
4  // クライアントとの接続処理
5  async def handle_client(websocket, path):
6      // 接続が続く限りtemperaturedataを送信
7      while True:
8          try:
9              await websocket.send(temperaturedata)
10         // クライアントが接続を閉じると終了
11         except websockets.exceptions.ConnectionClosedError:
12             break
13
14 // Websocketサーバを指定のアドレスとポート番号で起動
15 start_server = websockets.serve(handle_client, IPAddress, PORT)
16 asyncio.get_event_loop().run_until_complete(start_server)
17 asyncio.get_event_loop().run_forever()
```

3.2.3 Mixed Reality Toolkit(MRTK)

Mixed Reality Toolkit (MRTK) は, Microsoft が提供するオープンソースであり, MR アプリ開発に必要なコンポーネントが揃っており, 図 3.6のように手を使用して物体や UI の操作をすることができる. 図 3.6に MRTK のボタンとスライダーのサンプル例, 図 3.7にハンドトラッキング操作のサンプル例を示す.

図3.6　ボタンとスライダー

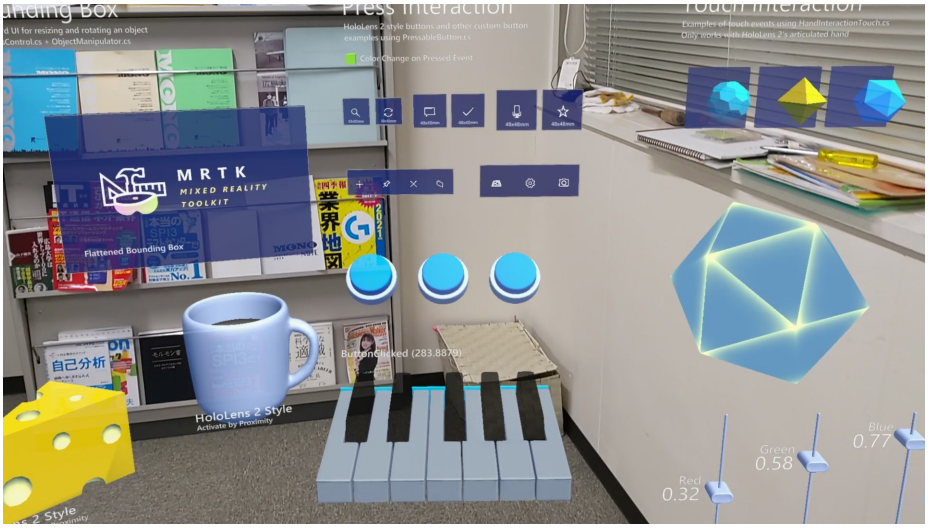

図3.7　ハンドトラッキング操作

3.2.4 websocket-sharp

websocket-sharp は, C# で WebSocket が扱えるライブラリである. WebSocket のサーバとクライアントの両方を構築でき, 非同期通信や簡潔な構築, クロスプラットフォーム (Windows, Linux, MacOS 等), SSL/TLS のサポートが特徴のライブラリである. 今回はクライアントとして利用しており, クライアントがサーバのデータを取得するコードをコード 3.2に挙げる.

コード 3.2　クライアント側コード

```
1  using System;
2  using WebSocketSharp;
3
4  public class TempVisualizer : MonoBehaviour
5  {
6    // WebSocketクライアントの参照変数
7    private WebSocket webSocket;
8
9    private void Start()
10   {
11     // サーバで指定したIPアドレスとポート番号のWebSocketクライアントを作成
12     webSocket = new WebSocket("ws://" + IPAddress + ":" + PORT);
13     // クライアントのメッセージ受信用のイベントハンドラを設定
14     webSocket.OnMessage += OnWebSocketMessage;
15     // 非同期でWebSocketサーバーに接続
16     webSocket.ConnectAsync();
17   }
18
19   // クライアントがメッセージを受信したときに実行されるイベントハンドラ
20   private void OnWebSocketMessage(object sender, MessageEventArgs e)
21   {
22     // WebSocketからデータを受信
23     string receivedData = e.Data;
24
25     // 受信データの可視化処理
26                 ⋮
27   }
28 }
```

3.3　動作例

視線先に応じた可視化の動作例を図 3.8と図 3.9に示す. 図 3.8は, 卒業研究での簡易的なセンサ SMH-01B01 を用いた可視化実装による動作例であり, 図 3.9は新たなセンサ MLX90640 を用いた可視化実装による動作例である.

下記の実装を比較すると, 可視化の視野角が向上していることが分かる. また, 可視化の精度は, 図 3.8では物体が検出できる精度であるのに対し, 図 3.9では物体の形状まで検出できる精度まで向上させることができた. さらに, 可視化の更新頻度は, 約 2 秒で 1 回から約 1 秒で 1 回の可視化が行うことが可能となった.

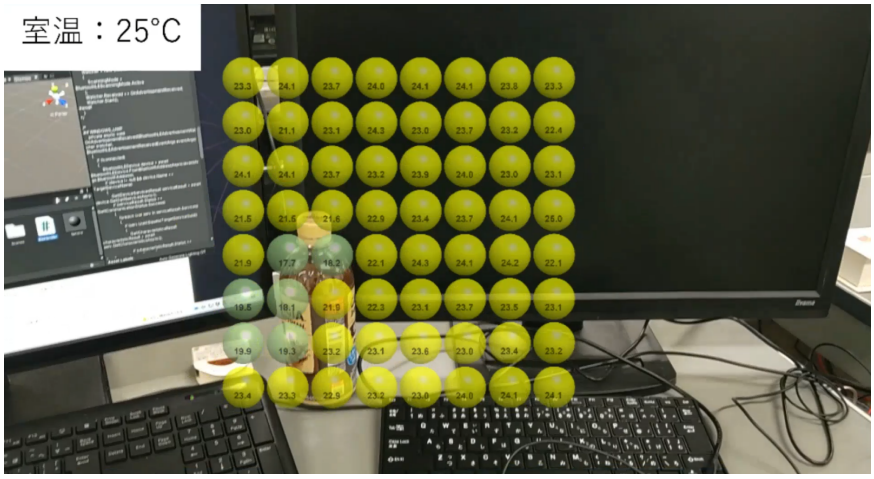

図3.8　簡易センサを用いた可視化

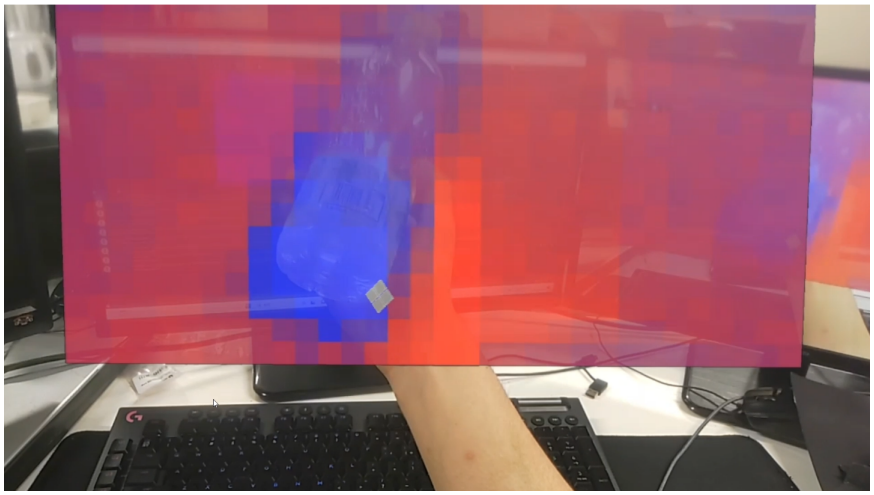

図3.9　新たなセンサを用いた可視化

第 4 章

空間マッピングを用いた温度の可視化

4.1　空間マッピング

空間マッピング [22] とは, 現実世界の 3 次元空間をデジタルデータで表現する技術であり, カメラやセンサを使用して現実世界の物理的な空間を 3D でスキャンすることで, 現実世界と同じようなデジタル環境を作成することができる. MRTK では, Spatial Awareness という空間マッピング機能が搭載されており, 周囲の表面メッシュを生成し表示することができる. この機能を利用して, 前章のように視野に温度分布を単に表示するだけではなく, スキャンしたオブジェクトの表面温度として測定した温度データをマッピングする. Hololens で表面メッシュを可視化させた例を図 4.1に示す.

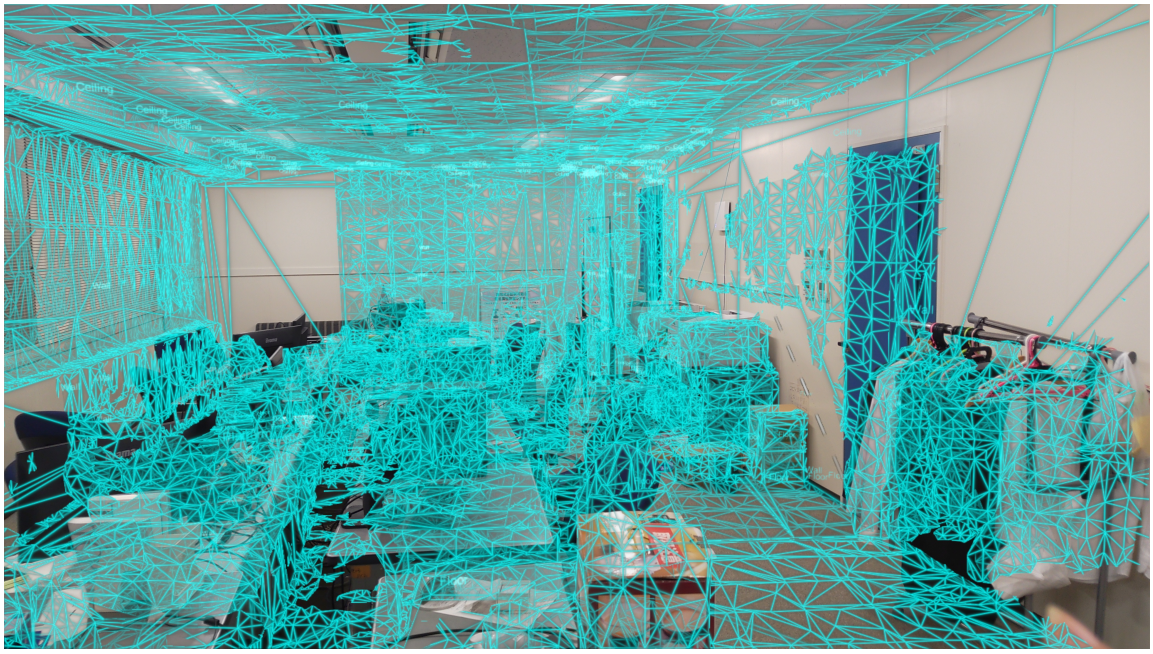

図4.1　空間マッピングの表面メッシュ

4.2　実装概要

空間マッピングを用いた可視化概の流れを以下に示す.

1. 空間マッピングで周囲の表面メッシュを取得
2. 投影テクスチャマッピングを行うプロジェクターを用意
3. プロジェクターを遅延を考慮した位置に配置
4. 表面メッシュに対して, カラーマップで色付けしたテクスチャを投影

本研究では空間マッピングで取得した 3 次元の表面メッシュに対して 2 次元のカラーマップを適用したテクスチャを貼り付ける処理が必要であるため, 投影テクスチャマッピングという手法を用いて赤外線アレイセンサの温度テクスチャを投影することを可能とした. 空間マッピングを用いた可視化概要を図 4.2に示す.

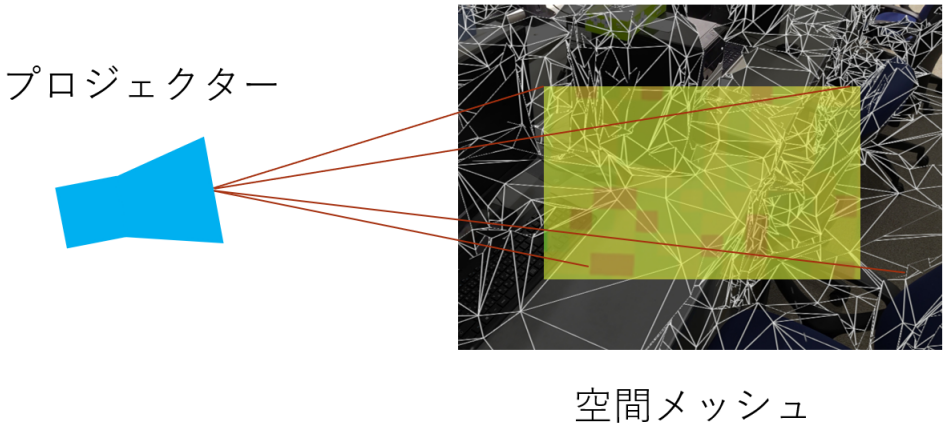

図4.2　空間マッピングを用いた可視化概要

4.3　投影テクスチャマッピング

投影テクスチャマッピング [23] とは, 3D オブジェクトに 2D のテクスチャをマッピングする手法の一つである. テクスチャという 2 次元の画像データを座標変換し 3 次元空間上に投影するプロジェクターのような技術である.

基本的な仕組みや流れを図 4.3に示す.

1. カメラのような視錐台を持つプロジェクターを配置 [24]
2. 3D オブジェクトに対して, プロジェクターのビュー行列とプロジェクション行列を渡す
3. 受け取った行列を用いて, 3D オブジェクトの各頂点を座標変換で 2 次元座標に変換
4. 座標変換された座標を投影するテクスチャにサンプリングする

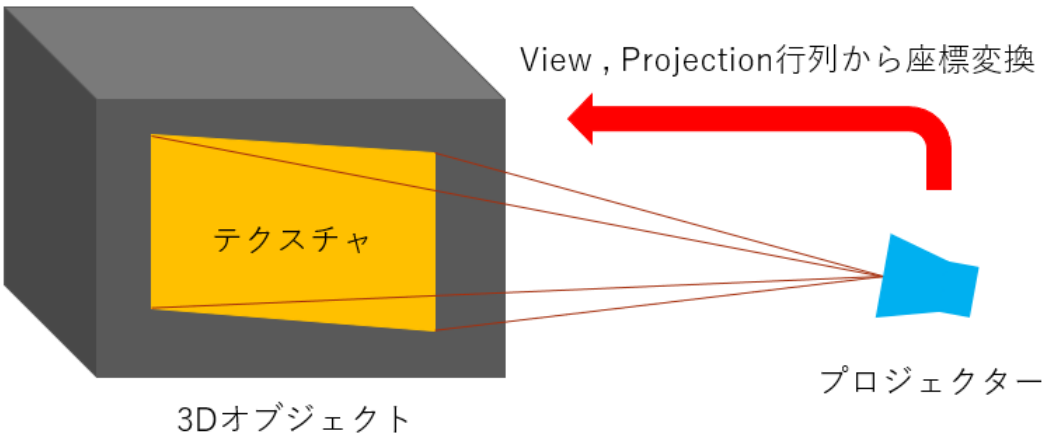

図4.3　投影テクスチャマッピング

4.4 HandMenu による可視化制御

本研究では, 視線先に応じた可視化を HandMenu[25] によって自由に制御や操作可能にした. HandMenu とは, HandTracking で検知した手に沿って UI を表示し, 空間内のオブジェクトの表示や操作等が可能となる. HandMenu は MRTK の UI コンポーネントの 1 つとして提供されている.

本研究では, 以下の機能を HandMenu で制御できるようにした.

- 可視化の開始と終了
  Start や Stop ボタンを押すことで, ユーザの任意のタイミングで可視化が可能となる.
- カラーマップの透明度 (Alpha) の変更
  可視化するテクスチャのカラーマップの透明度をスライダーで自由に変更可能とした. 透明度の初期値は 0.5 となっている.

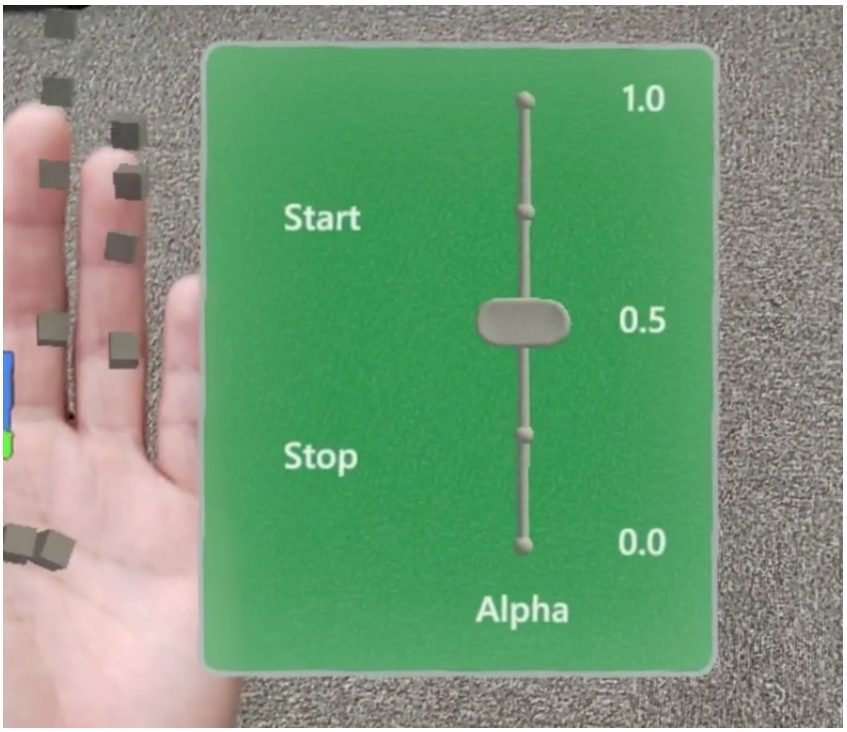

図4.4　HandMenu による可視化制御

4.5　動作例

空間マッピングを用いた可視化の動作例を図 4.5, 図 4.6に示す. 図 4.5は表面メッシュにカラーマップを適用したテクスチャを投影した動作例で, 図 4.6は HandMenu で可視化を制御する動作例である. 空間マッピングを用いることで, 空間内の壁面やオブジェクトの形状や位置の視認性が向上した. また, HandMenu を追加することでユーザが温度を視認しながら可視化を制御することができた. しかし改善点として, 温度差が小さい領域やメッシュの形状によっては視認性に欠けることが挙げられた.

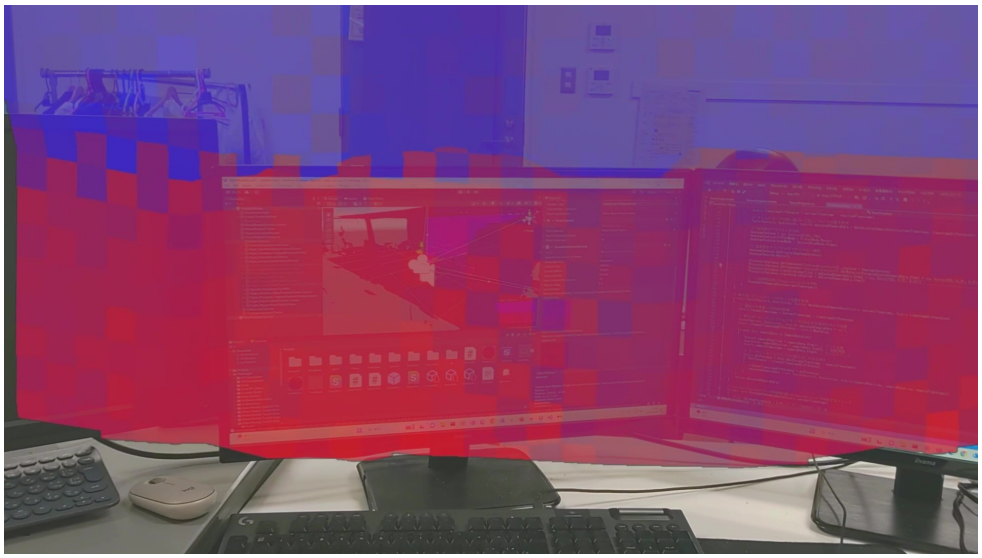

図4.5　空間マッピングを用いた可視化動作例 1

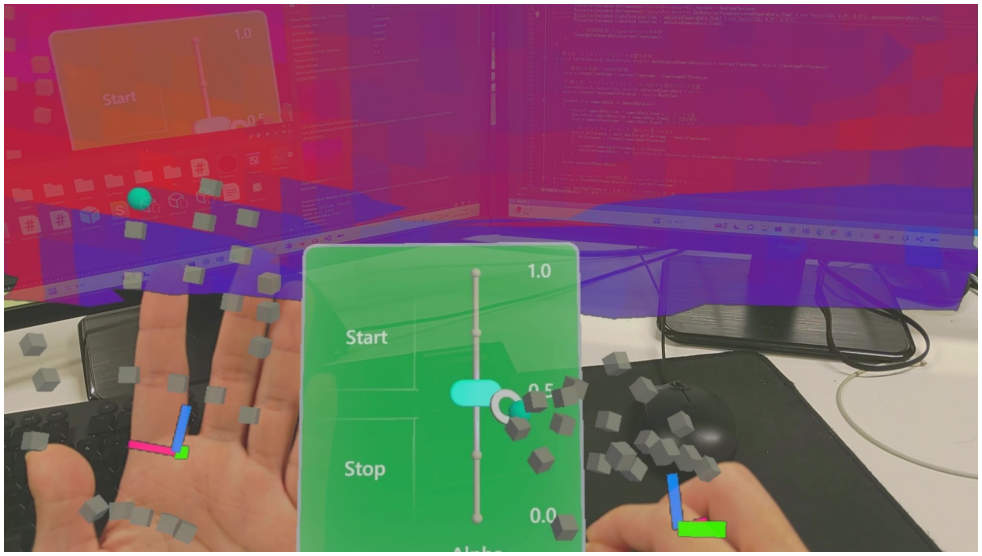

図4.6　空間マッピングを用いた可視化動作例 2

第 5 章

まとめ

本研究では Hololens 2 による空間マッピングを用いた温度可視化ツールを作成した. 卒業研究に比べ, 視野角や可視化の精度, 更新頻度を向上させることができた. また, 空間マッピングとテクスチャマッピングを用いることで温度の視認性を向上させることができた. 今後の課題としては, 領域間での温度補間機能の実装, 新たに温度センサを導入しリアルタイムで高精度な温度の可視化, UI の追加によるユーザビリティの改善等が挙げられる.

謝辞

本研究は広島大学先進理工系科学研究科情報科学プログラム計算機基礎学研究室において行ったものです. 本研究を進めるにあたり熱心かつ丁寧なご指導を賜りました中西透教授に深く感謝申し上げます. また, 本論文作成にあたって様々な助言を頂いた, 近堂徹教授, 北須賀輝明准教授, 今井克暢准教授並びにご協力いただきました計算機基礎学研究室の大学院及び学部生の皆様にも深く感謝申し上げます.

学術研究論文実績

- 穂高正, 中西透, 今井克暢, SIGNAC 第 39 回研究会人工知能合同研究会, 2023

参考文献

[1] Microsoft, ”Hololens 2”, https://www.microsoft.com/hololens/hardware, (2024/01/24 参照).

[2] MagicLeap, ”Magic Leap 2”, https://www.magicleap.com/ja-jp/magic-leap-2, (2024/01/24 参照).

[3] 富士ソフト株式会社, 安藤ハザマ, ” 環境ウォッチ ver.2”, https://www.fsi.co.jp/company/news/20220329_2.html, (2024/01/24 参照).

[4] Wilson, A.D., ”Combating the Spread of Coronavirus by Modeling Fomites with Depth Cameras”, Proc, the ACM on Human-Computer Interaction, 4, Issue ISS, pp 1–13.

[5] 中川航, 松本一紀, ドゥソルビエフランソワ, 杉本麻樹他: ” 環境 3 Dモデルによるカメラトラッキングを用いた温度分布の AR 表示システム”, 映像情報メディア学会誌 69(4), J160-J168(2015).

[6] 横井一樹, 福田知弘, 矢吹信喜, Ali Motamedi: ” 室内温熱環境設計フィードバックのための CFD と AR の統合－緑化を対象にして”, 日本建築学会 39, 89-92(2016-12).

[7] アズビル株式会社, ” 赤外線アレイセンサシステム”, https://www.azbil.com/jp/product/building/system/sensor/infrared-array-sensor-system/index.html, (2024/01/24 参照).

[8] Raspberry pi, https://www.raspberrypi.org/, (2024/01/24 参照).

[9] 穂高正, ”AR デバイスを用いた空間温度状態の可視化” 広島大学情報科学部論文, 2021.

[10] spartfun, ”SparkFun IR Array Breakout - 55 Degree FOV, MLX90640 (Qwiic)”, https://www.sparkfun.com/products/14844, (2024/01/24 参照).

[11] セイコー NPC 株式会社, ”SMH-01B01[赤外線アレイセンサモジュール]”, https://www.npc.co.jp/products/437, (2024/01/24 参照).

[12] 東阪電子機器株式会社, ”I2C(Inter-Integrated Circuit) について”, https://tohan-denshi.co.jp/technical-information/1835/, (2024/01/24 参照).

[13] Tutorialspoint, ”WebSockets - Implementation”, https://www.tutorialspoint.com/websockets/websockets_implementation.htm, (2024/01/24 参照).

[14] 日本データ通信協会タイムビジネス認定センター, ” タイムスタンプのしくみ”, https://www.dekyo.or.jp/tb/contents/summary/system_2.html, (2024/01/24 参照).
[15] Python Docs, ”subprocess - サブプロセス管理”, https://docs.python.org/ja/3/library/subprocess.html, (2023/08/03 参照).
[16] SSH.NET, ”SSH.NET”, https://github.com/sshnet/SSH.NET, (2023/08/03 参照).
[17] Adafruit, ”Adafruit_MLX90640”, https://github.com/adafruit/Adafruit_MLX90640, (2023/07/06 参照).
[18] Read the Docs, ”websockets documentation”, https://websockets.readthedocs.io/en/stable/, (2024/01/24 参照).
[19] Microsoft, ”Microsoft Mixed Reality Toolkit v2.8.3”, https://github.com/Microsoft/MixedRealityToolkit-Unity/releases, (2024/01/24 参照).
[20] STA, ”websocket-sharp”, https://github.com/sta/websocket-sharp, (2023/07/06 参照).
[21] Adafruit, ”Adafruit_CircuitPython_MLX90640”, https://github.com/adafruit/Adafruit_CircuitPython_MLX90640, (2023/07/06 参照).
[22] Microsoft, ” 空間マッピング - Mixed Reality”, https://learn.microsoft.com/ja-jp/windows/mixed-reality/design/spatial-mapping, (2024/01/24 参照).
[23] Haruki Yano, ”3D モデルにテクスチャを投影する投影テクスチャマッピングを実装する”, https://light11.hatenadiary.com/entry/2020/02/22/181705, (2024/01/10 参照).
[24] ”【Unity/ShaderGraph】テクスチャをプロジェクタのように投影する”, https://qiita.com/jyakushiiii/items/5e70f7801e1a65cfb384, (2024/01/10 参照).
[25] Microsoft, ” ハンドメニュー - MRTK2”, https://learn.microsoft.com/ja-jp/windows/mixed-reality/design/hand-menu, (2024/01/24 参照).