モバイル MR システム構築のための機能分散型フレームワーク(7)
ースマートフォンのためのインタラクション機構の拡張ー

A Distributed Framework for Mobile Mixed Reality System (7)
- Improvement of Interaction Mechanism for Smartphone -

小泉全弘，柴田史久，木村朝子，田村秀行
Masahiro Koizumi, Fumihisa Shibata, Asako Kimura and Hideyuki Tamura
立命館大学大学院 理工学研究科（〒525-8577 滋賀県草津市野路東 1-1-1）

Abstract: We are aiming at the realization of a distributed framework for mobile mixed reality system which supports various kinds of mobile devices. To support smartphones in our framework, it is necessary to improve an interaction mechanism of our framework because smartphones have many restrictions due to their specification. One of the problems is latency in the interaction and another one is how to establish interaction means for mobile device without keyboard. To solve the former problem, we have implemented the communication thread which was only used for transmitting the information related to the interaction. Moreover, to solve the latter problem, we introduce the interaction method suitable for the touch panel equipped with smartphones to improve variety of interaction.

Key Words: *Mixed Reality, Mobile device, Interaction Mechanism, Smartphone*

1. はじめに

我々はこれまで，複数のクライアントが同一の MR 空間を共有可能なモバイル MR システム構築のためのフレームワークについて検討してきた [1] [2]. 我々が提案するフレームワークは，サーバが MR 空間中に存在する仮想オブジェクトなどのコンテンツを一元的に管理するサーバ・クライアント方式を採用しており，各クライアントは定期的にサーバと通信することでコンテンツの同期を図る．しかしながら，近年急速に普及しつつあるスマートフォンをベースに本フレームワークのクライアントを実現するためには，端末性能に起因するインタラクション時のレイテンシ増加への対処や，キー入力に代わる新たなインタラクション手段の確立が課題となる．

そこで本研究では，前者の課題については通信機構を再設計する．具体的には，インタラクションに関する専用の通信スレッドを作成し，インタラクションの影響があった情報のみを通信する方式を採用する．後者の課題に対しては，インタラクションにおける多様性の向上を図るため，タッチパネルを用いたインタラクション手法を導入する．

2. コンテンツ制御機構

2.1 コンテンツ制御機構の概要

本フレームワークは，MR を実現する上で必要な位置合わせやレンダリングの機能を隠蔽し，仮想オブジェクトの動きのみを制御するための簡易なスクリプト言語を提供することで，MR システム開発の利便性を向上させている．このスクリプト言語を逐次実行し，MR 空間中に存在する仮想オブジェクトの動きを制御しているのがコンテンツ制御機構である．図 1に示すように，コンテンツはすべてサーバに集約されており，クライアントは一元管理された情報を定期通信によって参照することで，クライアント間で整合性が保たれた MR 空間を共有する．クライアントは，定期通信で受信した MR 情報を補間することで，仮想オブジェクトの滑らかな動きを実現する．以下，サーバ・クライアント間の処理と，サーバのスクリプトエンジンの処理を詳述する．

2.2 サーバ・クライアント間の処理

クライアントはユーザが行ったインタラクションの解析及びサーバとの定期通信を行う．図 2に示すように，クライアントが解析したユーザのインタラクション情報は，定期通信によってサーバへ送信され，MR 空間に登録される．一方，クライアントはその時点，すなわちインタラクション前の MR 空間のコンテンツ情報を受信する．登録するンタラクション情報は，サーバが周期的に実行するスクリプトエンジンによって処理され，スクリプトエンジンはその結果を MR 空間に上書きすることで，MR 情報を更新しインタラクションを反映する．定期通信とスクリプトエンジンの実行は同一周期で行わないため，インタラクション情報を反映した MR 空間のコンテンツ情報をクライア

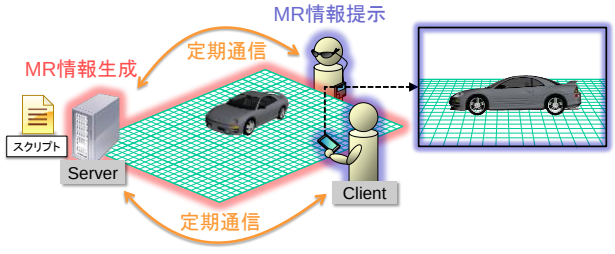

図1 コンテンツ提示の仕組み

ントが受信するのは，次回通信時となる．

2.3 スクリプトエンジン

サーバは各仮想オブジェクトの位置姿勢などの状態変化を記述したスクリプトを解析・実行することでコンテンツの情報を更新する．

スクリプトの言語仕様は Java をベースに設計しており，簡単な記述で複雑な動作が表現可能である．本言語で主に使用するクラスを**表 1**に示す．本フレームワークで扱うコンテンツは，実物体や仮想物体を表すオブジェクトとクライアントから構成される．オブジェクトは 3DCG や 2DCG などの仮想オブジェクトと，空間中に存在する実物体に対応する実オブジェクトから構成される．オブジェクトの情報は Object クラス，2DCG の場合は ClientObject クラスをそれぞれ継承したクラスを定義し，そこにオブジェクト毎の動作を記述する．MR 空間を表現した MRWorld クラスの中でこれらのインスタンスを生成することによって，オブジェクトを配置する．一方，クライアントの情報は，システムに対してクライアントが最初に接続した際に，自動的にそれぞれその情報を格納する Client クラスのインスタンスを生成する．

2.4 インタラクション手段

インタラクションの手段は，キー操作とインタラクション用デバイスの 2 つである．前者は端末のキーボードを押下することでオブジェクトやクライアントに対してインタラクションを行い，後者は位置姿勢を検出するデバイスを使用し，オブジェクトに対して直接インタラクションを行う．

3. 通信機構の再設計

3.1 概要

従来のフレームワークは，サーバが MR 空間中に存在するコンテンツを一元的に管理し，クライアントはサーバと定期通信することでコンテンツの同期を図っていた．クライアントは，インタラクション情報をサーバへ送信し，サーバは，インタラクション情報をスクリプトで解析するこ

表1 言語仕様の一部

フィールド名	概要
MRWorld クラス	
_objectList	オブジェクトを管理するリスト
_clientList	クライアントを管理するリスト
Object クラス	
int ID	ID
Position3D pos	3 次元の位置
Orientation ori	姿勢（各軸 回転）
Scale3D scale	拡大率
boolean presen	表示（再生）／非表示（停止）
int state	状態
ClientObject クラス	
int ID	ID
Position2D pos	2 次元の位置
double roll	回転
Scale2D scale	拡大率
boolean presen	表示（再生 ／非表示（停止）
int state	状態
Client クラス	
int ID	ID
Position3D pos	位置
Orientation ori	姿勢（各軸 回転）
int interaction	キー情報
int state	状態

とで，MR 空間にインタラクション情報を反映していた．しかし，定期的に通信するため，インタラクション情報の送信が通信間隔やインタラクションの時機に依存し，インタラクションにおけるレイテンシが増加する．特にスマートフォンをベースにクライアントを実現する場合は，端末の性能に起因するレイテンシ増加が問題となる．そこで本研究では，端末性能に起因するインタラクションにおけるレイテンシ増加の問題を改善するために，2 つのアプローチをとる．1 つ目は，インタラクション情報を即座にサーバへ送信するために，新たに通信用スレッドを作成し，定期通信とは別にインタラクション情報を通信する（インタラクション専用通信）． 2 つ目は，通信時間や通信解析に要する時間を削減するために，インタラクション専用通信では，クライアントが受信する情報をインタラクションの影響があった情報のみに限定する．

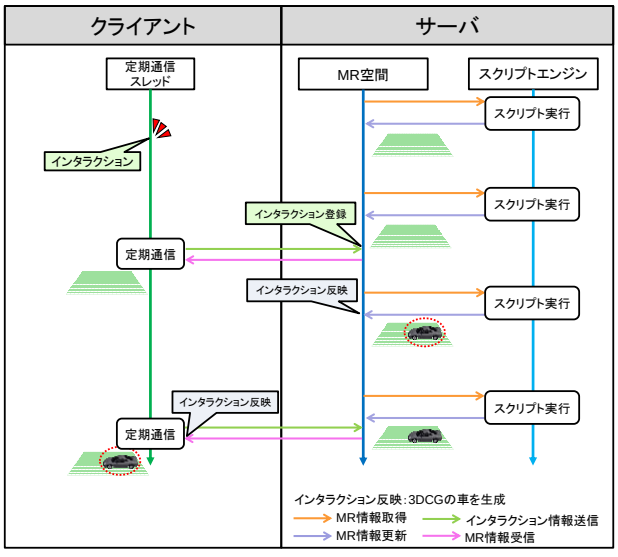

図2 従来のサーバとクライアントの処理の流れ

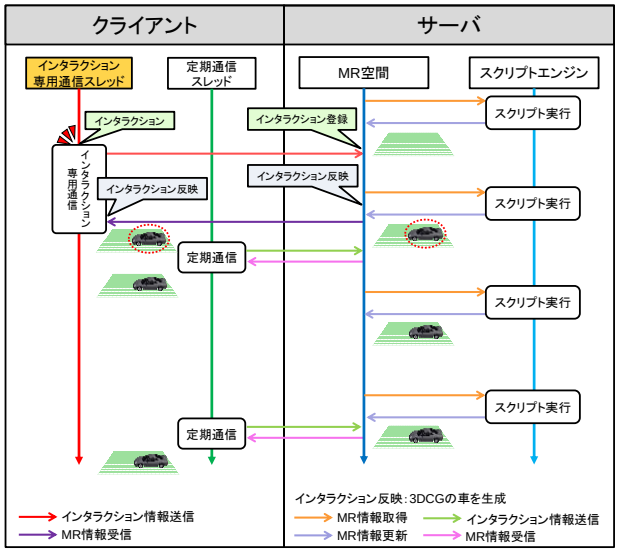

図3 サーバとクライアントの処理の流れ

3.2 インタラクション専用通信

本手法は，新たにインタラクション情報を通信するためのインタラクション専用通信を導入する．2 章で述べたように，定期通信はスクリプトの実行と同一周期で行わないため，インタラクションの結果をクライアントに提示するのは次回の定期通信時であった．そこで，クライアントがインタラクションの結果を速やかに取得可能とするために，**図 3**に示すようにインタラクション専用通信による処理の流れを再設計した．

クライアントはインタラクション専用通信で即座にインタラクション情報をサーバへ送信し，サーバはそれを MR 空間に登録する．従来のフレームワークは，クライアントがこの時点の MR 情報を受信していたが，インタラクション専用通信では，スクリプトエンジンがインタラクション情報を処理するまでクライアントは受信を待機する．そして，MR 空間にインタラクションが反映された時点で，クライアントは MR 情報を受信する仕組みをとることで，インタラクション反映後の MR 情報を速やかに取得可能とする．

3.3 受信するオブジェクト情報の限定

前項で述べたインタラクション専用通信において，サーバはインタラクションの影響を受けたオブジェクトの情報だけを限定する．インタラクションの影響を受けたオブジェクトは，例えば座標が変化するなどの，インタラクションが原因で保持するパラメータが変化したオブジェクトのことを示す．インタラクションの影響を受けたオブジェクトを判定する手法として，インタラクションの影響が及んだオブジェクトに識別 ID を付加することで判定する．識別 ID とは，インタラクションの影響が及んだオブジェクトに付加する目印であり，サーバはこの識別 ID が付加されたオブジェクトのみをクライアントへ送信することで，オブジェクトの限定を図る．また，識別 ID にはどのクライアントのインタラクションかを判別するために，クライアント固有の ID 値を用いる．

4. タッチパネルによるインタラクション手法

4.1 概要

従来のフレームワークは，使用可能なインタラクションの種類としては，キー操作とインタラクション用デバイスの 2 つであった．しかし，スマートフォンはキーボードを搭載していない端末も多く，キーボード非搭載の端末ではインタラクションにおけるコンテンツ制御の多様性が減少する．そこで，キーボード非搭載の端末におけるインタラクション手段として，タッチパネルを利用したインタラクション手法を導入する．本インタラクション手法は，タッチジェスチャを用いることでコンテンツを操作する．今回導入するタッチジェスチャの種類は，**図 4**に示すスマートフォンで一般的に用いられているタップ，ダブルタップ，ドラッグ，フリック，マルチタッチの 5 つである．

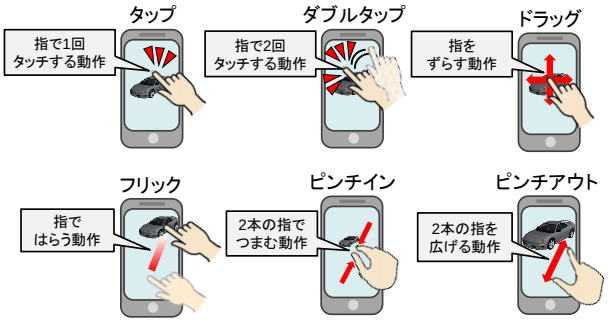

図4 導入するジェスチャの種類

表2 スクリプト言語に追加するパラメータ・関数

パラメータ・関数	概要
Client クラスに追加するパラメータ	
Position2D touchPosA	タッチした画面上の座標 1
Position2D touchPosB	タッチした画面上の座標 2
Position3D touchPosA3D	タッチした位置の 3 次元座標 1
Position3D touchPosB3D	タッチした位置の 3 次元座標 2
VelocityVector2D velocity	フリック時のベクトル 1
VelocityVector2D velocity	フリック時のベクトル 2
System に追加する関数	
boolean isTouch (Position2D pos, ClientObject obj2d)	2DCG をタッチしたか判定
boolean getTouchDistance (Position2D pos1, Position2D pos2)	2 本の指の間の距離を取得
boolean isTouch3D (Position3D clientpos, Position3D touchpos,Object obj3d)	3DCG をタッチしたか判定

4.2 スクリプト言語の拡張

ジェスチャを用いてオブジェクトを制御するために，スクリプト言語を拡張する．スクリプト言語は，タッチジェスチャの種類毎に制御を記述するため，**表 2**に示すパラメータと関数をスクリプト言語に追加する．タッチしたスクリーン上の座標は，2DCG をタッチしたか否か判定する際などに必要になり，フリックのベクトルは，X 方向，Y 方向それぞれにおける 1 秒間に進んだピクセル量を示すパラメータである．また，スクリプトの記述をより簡単にするために，オブジェクトをタッチしたか否かを簡単に取得するための関数，マルチタッチ時に 2 本の指の間の距離を取得するための関数をシステム側で用意する．更に，3DCG の制御は，タッチした位置の 3 次元座標を利用する．タッチした位置の 3 次元座標は，「タッチした位置の 2 次元座標×逆ビューポート行列×逆射影変換行列×逆モデルビュー行列」で求めることができ，これをワールド座標系に変換したものをサーバへ送信する．オブジェクトのタッチ判定は，**図 5**に示すようにクライアントの座標とタッチした位置の 3 次元座標を結んだ線分の，奥行き方向への延長線を計算し，この延長線と対象の 3DCG が交差した際にタッチしたと判定する．

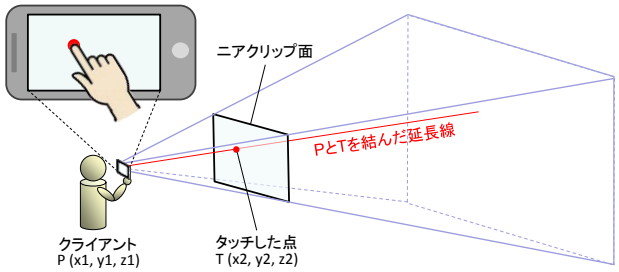

図5 3DCG タッチ判定に用いる延長線の算出

5. 実験

以上の設計からインタラクション機構を拡張し，提案機構が想定通りに動作するか確認する．実験には，**表 3**に示す 1 台のサーバ機，Android OS を搭載したスマートフォン，Windows 7 を搭載したノート PC を用い，位置姿勢検出に ARToolKit を利用した．

- **実験 1：通信機構再設計の動作確認**

通信機構を再設計したことで，インタラクション反映までに要する時間を削減できたか確認する．実験はユーザがタップ，キー入力することで，仮想オブジェクトを 1 つ生成するインタラクションを行い，それがクライアントに反映されるまでの時間を計測する．その際，定期通信の間隔は，100ms，300ms，500ms と変更して実験する．この実験を再設計前と再設計後において，それぞれの通信間隔で 20 回試行し，その平均値を算出した．実験結果を**表 4** に示す．実験結果より，通信間隔以下の速度でインタラクションの反映をすることができ，通信間隔に依存しないインタラクションの反映ができたことを確認した．

- **実験 2：タッチパネルによるインタラクション手法の動作確認**

タッチした位置の 3 次元座標を正しく算出できるか確認する．**図 6**に示すように、実験は MR 空間の原点に X,Y,Z の座標軸を表す 3DCG を表示し，タッチした位置の 3 次元座標を画面に出力することで，タッチした位置の 3 次元座標が正しく算出できたか確認する．また，タッチ判定関数が正しく動作するかを確認する．クライアントの座標が (200,500,450)，姿勢 (40,0,240) の場合にスクリーン上の座標 (350,200) をタッチした場合，理論上 MR 空間中の座標 (-98,70,0) をタッチできる．実験はこの理論値である座標 (-98,70,0) に 3DCG を配置し，実際に正しく 3DCG をタッチできるか確認する．画面上には現在のクライアントの位置姿勢と，タッチすべき座標に目印となる 2DCG を配置する．記述したスクリプトの一部を**図 7**に，実験結果を**図 8**に示す．タッチした位置の 3 次元座標を正しく算出でき，タッチ判定関数が正しく動作することを確認した．

表3 実験環境

端末	OS	CPU	RAM	通信
自作機（サーバ）	Windows7	Core i5 3.20GHz	4.00GB	有線
DELL XPS ADAMO	Windows7	Core 2 Duo 2.13GHz	4.00GB	IEEE 802.11n
SAMSUNG GALAXY S II	Android 2.3	S5PC210 1.20GHz×2	1.00GB	IEEE 802.11n

表4 実験 1 の結果

通信間隔 [ms]	再設計前 [ms]	再設計後 [ms]
SAMSUNG GALAXY S II での計測		
100	150.1	56.2
300	450.7	55.2
500	768.7	52.2
DELL XPS ADAMO での計測		
100	148.7	44.9
300	380.4	45.7
500	609.1	48.8

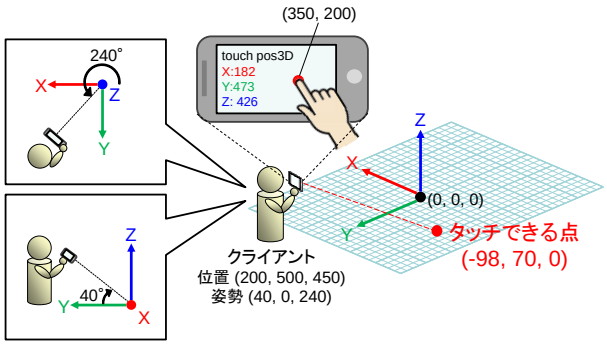

図6 実験 2 の概要

```
public class CarDrive extends MRWorld{
    public CarDrive(){   /*コンストラクタ*/
    /*省略：仮想オブジェクトのインスタンスを生成*/
    }
    /*メインループメソッド*/
    public void mainloop(){      /*コンテンツ同士の関係性を記述*/
        for(int i = 0; i < _clientList.size(); i++){
            Client client = _client.get(i);
            for(int j = 0; j < _objectList.size(); j++){
                Object obj = _objectList.get(j);
                if(obj instanceof Car){
                    //3DCG のタッチ判定
                    if(System.isTouch3D(client.pos,client.touchPosA3D, obj)){
                        obj.state = 1;        //選択状態に切替
                    } else{
                        obj.state = 0;
                    }
                }
            }
        }
    }
}
```

図7 MRWorld クラスの一部

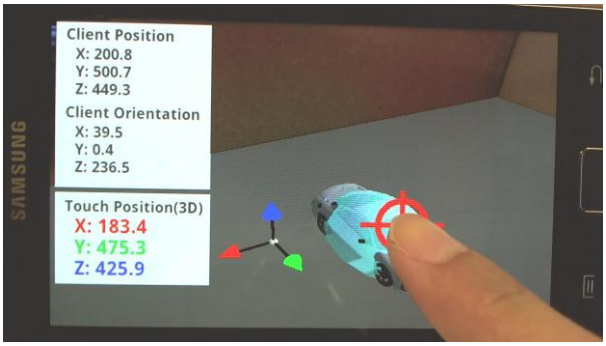

図8 実験 2 の結果

6. むすび

本稿では，我々がこれまで設計・実装を進めてきたモバイル MR システム構築のためのフレームワークにおいて，スマートフォンのためにインタラクション機構を拡張した結果について報告した．2 つのアプローチから通信機構を拡張し，インタラクションにおけるレイテンシ増加の問題を改善した．また，タッチパネルを利用したインタラクション手法を導入したことで，スマートフォンにおいてインタラクションによるコンテンツ制御の多様性が向上した．今後の展望としては，ドラッグなどの連続したインタラクションを制御する新たな手法の開発と，スマートフォンのクライアント向けにインタラクション機構以外の処理を最適化する．

参考文献

[1] 山下他：“モバイル MR システム構築のための機能分散型フレームワーク - システムアーキテクチャとコンテンツ制御機構 -”，第 14 回日本バーチャルリアリティ学会大会論文集，3A2-3, 2009.

[2] 縄谷他：“モバイル MR システム構築のための機能分散型フレームワーク(2) - コンテンツ制御機構の拡張 - ”，第 15 回日本バーチャルリアリティ学会大会論文集，pp. 374 -377, 2010.