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-

A Distributed Framework for Mobile Mixed Reality System (2)
-Improvement of Content Control Mechanism-

縄谷侑司，山下智紀，柴田史久，木村朝子，田村秀行
Yuji Nawatani, Tomonori Yamashita, Fumihisa Shibata, Asako Kimura and Hideyuki Tamura

立命館大学大学院 理工学研究科（〒525-8577 滋賀県草路津市野東 1-1-1）

Abstract: We have improved the content control mechanism of the distributed framework for mobile mixed reality system that we proposed. The new mechanism enables each mobile device to present different contents individually by introducing an idea to make groups of mobile devices. In addition, the difference in performance of mobile devices and heavy network traffic sometimes cause inconsistency of the contents presentation. To cope with this problem, we have developed an interpolation algorithm based on Bézier curve and a communication protocol.

Key Words: *Mixed Reality, Mobile device, Content Control Mechanism*

1. はじめに

我々はサーバ・クライアント方式を採用したモバイル MR システム構築のためのフレームワークを開発している[1]. 本フレームワークでは，様々なアプリケーション開発に対応可能とするため，仮想物体などのコンテンツを柔軟に制御するコンテンツ制御機構を開発した．本機構では，コンテンツの制御をサーバに集約することで，クライアント間で MR 空間共有を可能としている．

しかし，既開発のフレームワークでは，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，全クライアントに対して同一の MR 情報を提示するため，端末毎に異なる情報提示ができないという問題を有していた．また，クライアントがサーバから情報を取得する間隔が大きくなってしまう場合，サーバで生成した情報を完全に復元することができず，MR 提示の破綻を起こす可能性がある．本研究では前者の問題に対して，端末のグルーピングという概念を導入し，端末毎に異なる情報を提示する機構を開発する．また後者については，本フレームワークのコンテンツ提示に特化した情報の復元手法と独自の通信方式の開発を目指す．

2. コンテンツ制御機構

コンテンツ制御機構とは，コンテンツの動きを制御する機構である（**図 1**）．本機構では，コンテンツの動きをプログラムとして記述し，記述に応じてコンテンツの位置・姿勢などの状態を変更する．この際，コンテンツをサーバに集約し，クライアントが一元管理された情報を参照する仕組みをとることで，クライアント間で整合性を保ち MR 空間を共有可能とする．さらに，サーバ・クライアント間で通信を行う際に発生する情報欠損の問題に対して，クライアント側でコンテンツ情報を復元する．本機構は，以下の 2 つの処理系によって構成する．

【スクリプトエンジン】

MR 空間中に存在するコンテンツを管理し，各コンテンツ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．スクリプトは独自に設計したスクリプト言語を用いて記述する．

本言語は，記述が容易であり，複雑なコンテンツの表現が可能であることを要件とし設計した．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とは，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．オブジェクトの情報は，Object クラスを継承した個々のオブジェクト毎の雛形となるクラスを定義し，そこにメソッドとしてオブジェクトの動きを記述する．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．具体例として，**図 1** に示す鳥が直線運動するスクリプトの記述例を**図 2,3** に示す．鳥のオブジェクトを表す Object クラスを継承した BlueBird クラスでは，

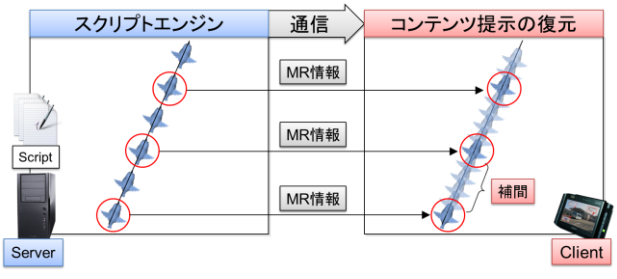

図 1 コンテンツ制御機構の概略

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
Position pos	位置
Orientation ori	姿勢（各軸の回転）
Scale 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
Position pos	位置
Orientation ori	姿勢（各軸の回転）
int interaction	キー情報
int state	状態

action メソッドに直線運動の動きを記述する．MRWorld クラスでは，コンストラクタにおいて，予め定義したフィールド変数である_objectList に BlueBird のインスタンスを追加することで，MR 空間にオブジェクトを配置する．そして，mainloop メソッドで BlueBird クラスの action メソッドを呼び出すことで，鳥は直線運動する．

【コンテンツ情報の復元】

クライアントがMR情報を提示するには，クライアントのフレーム更新時間と同間隔で毎フレームの情報をサーバから取得する必要があるが，通信量や通信速度の問題から実現は困難である．そこで本研究では，サーバはMR空間の更新を行い，クライアントはサーバからMR空間の情報を一定間隔で取得する．その上で，クライアントは次の通信までの間，自身が保持する現時点の位置姿勢情報とサーバから取得した位置姿勢情報を補間することで現在時刻でのMR空間の情報を生成し，MR画像を提示する．

3. 個別対応型情報提示機構

3.1 概要

これまでのフレームワークでは，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，全クライアントに対して同一の MR 情報を提示することで，複数の端末間で MR 空間を共有していた．そのため，端末毎に異なる MR 情報を提示することを考慮しておらず，アプリケーション開発の幅を狭めていた．そこで本研究では，端末のグルーピングという概念を導入し，端末毎に異なる情報を提示する機構を実現する．また，端末毎に異なる情報を提示する機構を実現するにあたり，端末の画面上への情報を提示することがアプリケーション開発の幅を拡げると考え，端末の画面上へコンテンツを提示する機構を実現する．これら 2 つの機構を含む個別対応型情報提示機構を実現するために，独自スクリプト言語を拡張する．

```
public class BirdGarden extends MRWorld
{
  /*コンストラクタ*/
  public BirdGarden (){  //オブジェクトの初期化
    Position pos;
    Orientation ori;
    /*省略:位置姿勢情報の入力*/
    _objectList.add(new BlueBird(pos, ori));
  }
  /* メソッド*/
  public void mainloop(){  //各オブジェクトの動きを呼び出し
    for(int i = 0; i < _objectList.size(); i++){
      Object obj = _objectList.get(i);
      obj.action();
    }
  }
}
```

図 2　MRWorld クラス

```
public class BlueBird extends Object
{
  /*フィールド*/
  private double MOVEMENT_DISTANCE_X = 1;
  private double MOVEMENT_DISTANCE_Y = 2.5;
  /*コンストラクタ*/
  public BlueBird (Position pos, Orientation ori){
    this.pos = pos;
    this.ori = ori;
  }
  /* メソッド*/
  public void action(){  //直線運動
    this.pos.x = this.pos.x + MOVEMENT_DISTANCE_X ;
    this.pos.y = this.pos.y + MOVEMENT_DISTANCE_Y ;
  }
}
```

図 3　Object クラス

3.2 独自スクリプト言語の拡張

拡張した本言語を**表 2** に示す．端末毎に異なるコンテンツを提示するために，Object クラスに Client クラスのクライアントの識別子である ID を管理するリストを追加する．リストに ID が格納されたクライアントに対してオブジェクト情報を提示する．また，クライアント独自の属性や振る舞いを持たすことで，クライアントに拡張性を持たすため，クライアントの情報は Client クラスを継承して個々のクライアント毎の雛形となるクラスを定義する仕組みを採用する．Client クラスには，端末のグルーピングを可能とするため，グループ ID と端末の種類や性能を表す属性情報を追加する．さらに，端末毎の画面上へコンテンツを提示するため，Client クラスに画面上のオブジェクト情報を管理する ScreenObject クラスと画面上へ提示する全オブジェクトを管理するリストを追加する．

4. 端末の処理能力を吸収するための制御手法

4.1 概要

本機構では，サーバは常時 MR 空間の更新を行い，クライアントはサーバから MR 空間の情報を一定間隔で取得し，次の通信までの間，自身が保持する現時点の位置姿勢情報とサーバから取得した位置姿勢情報を補間する復元手法を採用していた．

しかし，情報を取得する間隔が大きくなってしまう場合にサーバで生成した情報を完全に復元することができず，結果 MR 提示の破綻を起こす可能性がある．通信間隔が大きくなる原因には，処理能力の低い端末で通信にかかる処理時間が大きくなることや，ネットワーク環境によって通信にラグが発生する場合，また他の処理負荷の変動が影響することなどが考えられる．

表 2　拡張言語仕様の一部

フィールド名	概要
Object クラス	
Link<int> cltList	提示するクライアントの ID リスト
Client クラス	
int group	グループ ID
int type	端末の属性
List<ScreenObject> scrList	スクリーンオブジェクトのリスト
ScreenObject クラス	
int ID	ID
Position2D pos	位置
double roll	姿勢（各軸の回転）
Scale2D scale	拡大率
boolean presen	表示（再生）／非表示（停止）
int state	状態

そこで本研究では，以下の 2 つの手法を考案し，この問題の解決を図る．

- 通信間隔の変動を吸収可能な補間処理アルゴリズム
- 新情報通信方式

4.2 補間処理アルゴリズム

提案する制御手法では，クライアントは描画フレーム更新時間よりも長い間隔でサーバから最新のコンテンツ情報を取得する．次の通信までの間，前回取得した情報と今回取得した情報から現在時刻の情報を補間処理によって生成する．補間処理には，3 次ベジエ曲線のアルゴリズムを用いた．3 次ベジエ曲線は，制御点と呼ばれる複数の点 $\{P_i\}$ に基づいて定義される多項式曲線である．点 P_0～P_n を滑らかに結ぶベジエ曲線 R(t) は，n+1 個の制御点に n 次の多項式空間であるベクトル空間の基底をなすバーンスタイン基底関数を混合比とした以下の**式 1** で表される．

$$R(t) = \sum_{i=0}^{n} B_i^n(t) P_i \qquad (1)$$

本制御手法は，サーバ上でコンテンツの 1 フレーム間の移動量からその点での速度ベクトルを算出し，位置姿勢情報と併せてクライアントに送信する．クライアントは既知の 2 点とその点での速度ベクトルからアニメーションの再現に必要な残りの 2 点を算出し，3 次ベジエ曲線のアルゴリズムにより補間を行うことで滑らかなアニメーションを実現した．詳細な手順は**図 4** 用いて以下に解説する．

(1) クライアントが保持するコンテンツの現在の位置を P_x，t 秒後にサーバから与えられたコンテンツの次の位置を P_{x+t} とする．サーバ上では図中の実線で示す曲線を描くアニメーションを制御している．この曲線の再現をクライアント上で行うことが目標である．

(2) サーバはクライアントに対して，位置姿勢情報とその位置での通信間隔 t 秒あたりの速度ベクトル情報をクライアントに送信する．これらの情報を元に P_x から直線的に運動すると仮定した場合の t 秒後の点 $P_x'_{+t}$，P_{x+t} へ直線的に到達すると仮定した場合の t 秒前の点 P_x'を算出する．

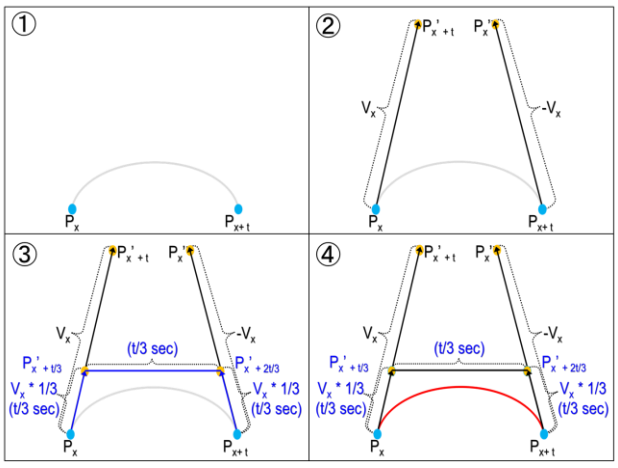

図 4　補間処理手順

(3) P_x から $P_x'_{+t}$ までの間，等速で直線的に動くと仮定し，t/3 秒通過する点 $P_x'_{+t/3}$ と，P_x'から P_{x+t} までの間，2/3t 経過する点 $P_x'_{+2/3t}$ を算出する．

(4) P_x から P_{x+t} と生成した $P_x'_{+1/3t}$，$P_x'_{+2/3t}$ に対して 3 次ベジエ曲線のアルゴリズムを用い，補間処理を行う．

このように通信時間 t に依存しない補間アルゴリズムを実現することで，処理能力差や処理負荷によって変化する通信頻度差を吸収することが可能となる．

4.3 新情報通信方式

従来の通信方式では，XML で通信情報を表現し，通信してきた．この通信方式を用いる場合，以下の 2 つの問題が考えられる．

- MR 空間中に存在するコンテンツの数が増加するとクライアントが受信する情報量が大きくなり，通信に負荷がかかる
- 通信する情報を XML に生成し，また受信側で解析する処理が必要となるため処理負荷が大きい

特に処理能力が低い端末では上記の要因による影響が顕著であり，通信間隔が大きくなることで結果的に意図しない補間結果を生成する問題が発生し易い．

そこで本機構では，現在用いている XML による通信を見直し，データの並びをあらかじめ定義したバイナリデータの配列を通信に用いる手法を検討する．バイナリ情報列のみを扱うので，XML では必要であったタグ情報を削減した分，通信量が大幅に削減できる．また XML データの生成と解析にはデータ量が増加するにつれ，処理時間が大きく増加していた．それに対し，提案手法ではバイナリの並びを定義した解析なので，XML 生成・解析に対して処理時間は大きく削減することが可能となる．

5. 実験

以上の設計からコンテンツ制御機構を拡張し，提案機構が想定通り動作するか確認する．実験内容は以下の 3 項目である．実験には，1 台のサーバ機器，4 台のモバイル端末と位置姿勢検出に ARToolKit を用いた．

- 実験 1：個別対応型情報提示機構の動作確認

サーバ 1 台とクライアント A,B,C,D の 4 台を用いて，個別対応型情報提示機構が想定通り動作するか確認する．実験は，クライアント A が UFO に対してミサイルを撃つというインタラクションを行う．記述したスクリプトの一部を**図 5,6** に，実験結果を**図 7** に示す．画面上にクライアントの得点，グループの得点を表す画像を提示できていることで，端末の画面上へコンテンツを提示することが実現できた．また，インタラクション結果から，クライアント A の得点，グループ 1 の得点が増えており，端末毎に異なるコンテンツを提示することが実現できた．

- 実験 2：提案補間処理アルゴリズムの動作確認

従来の補間手法と提案手法を比較することで，前章で挙げた通信間隔の変動における問題を解決できたか確認する．実験は，通信間隔を 1000msec，2000msec で固定し，サインカーブの動きの補間処理の比較結果を**図 8** 示す．青

色線がサーバで生成したコンテンツの位置情報，赤色線が従来の直線補間によって生成された情報，緑色線が提案手法によって復元した情報である．従来手法と比べて，通信間隔が 1000msec で変曲点が 1 つ以下の曲線は良好な結果を得ることができた．しかし，通信間隔が 2000msec で 2 点間の軌跡に変曲点が存在する場合，サーバの動きを再現することは不可能であった．

- 実験 3：新通信機構と従来の通信機構との比較

  従来の通信機構と新通信機構との処理時間を比較することで，前章で挙げた通信量の問題が解決できたか確認する．比較結果を**表 3** に示す．従来の通信機構と比べて，新通信機構では通信量を約 80%削減できた．

```
public class CattleMutilation extends MRWorld
{
  /*フィールド*/
  private CLIENT_TYPEA = 1;              //個別の種類
  private GROUP_TYPE1 = 1;               //グループの種類
  /*省略:その他の変数*/
  /*コンストラクタ*/
  public CattleMutilation(){
  /*省略:オブジェクトのインスタンスを生成*/
  }
  /* 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 = _clientList.get(i);
      for(int i = 0; i < _objectList.size(); i++){      オブジェクトの提示
        Object obj = _objectList.get(i);
        if(obj instanceof NewTarget1){
          if(client.getID() == 1){
            obj.addPresenID(client.getID());  //クライアント毎に情報提示
            /*省略: 他のターゲット表示をクライアント毎に提示*/
          }else{
            obj.addPresenID(client.getID());  //全クライアントに情報提示
          }
        }
      }
      /*省略:クライアントがUFOをミサイルで撃墜*/          得点加算
      if(client.getType() == CLIENT_TYPEA){
        client.addClientPoint(); //クライアント毎の得点の加算
      }else if(client.getGroup() == GROUP_TYPE1){
        client.addGroupPoint();  //グループ毎の得点の加算
      }
      /*省略: 他の個別とグループの得点提示*/
    }
  }
  public void addClient(){
  /*省略:クライアントのインスタンスを生成*/
  }
}
```

図 5　MRWorld クラスの一部

```
public class ClientTypeA extends Client
{
  /*コンストラクタ*/                    スクリーンオブジェクトの配置
  public ClientTypeA(){
    _screenObjectList.add(new CltPoint(cPointPos, cPointRoll));
    _screenObjectList.add(new GrpPoint(gPointPos, gPointRoll));
  }
  /* メソッド*/
  public void addClientPoint(){
  /*省略:クライアント毎の得点を加算*/
  }
  public void addGroupPoint(){
  /*省略:グループの得点を加算*/
  }
}
```

図 6　Client クラスの一部

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 システムを構築するための機能分散型フレームワークにおいて，コンテンツ制御機構を拡張した結果について報告した．本フレームワークは，端末のグルーピングという概念を導入し，端末毎に異なるコンテンツを提示する機構を開発した．また，本フレームワークのコンテンツ提示に特化した情報の復元手法と独自の通信方式を開発した．これにより，端末の処理能力の差を吸収可能なコンテンツ提示を実現した．今後の展望は，前章で述べた問題の解決を図り，汎用的なフレームワークの構築を目指す．

参考文献

[1]　山下智紀他.：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, 9.2009

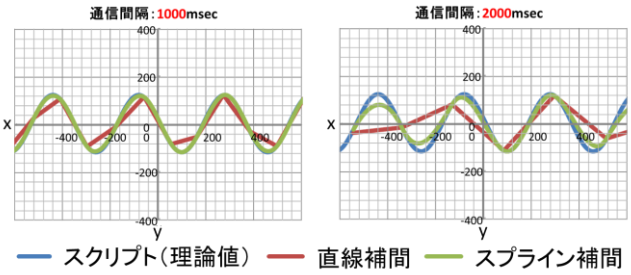

図 8　サインカーブの実験結果

表 3　通信量の比較

	従来方式 [byte]	新方式 [byte]
必須タグ（ヘッダ）	151	10
コンテンツ (A)	1,078	210
子コンポーネント (B)	379	105
合計（最小構成 A:1, B:1）	1,608	325

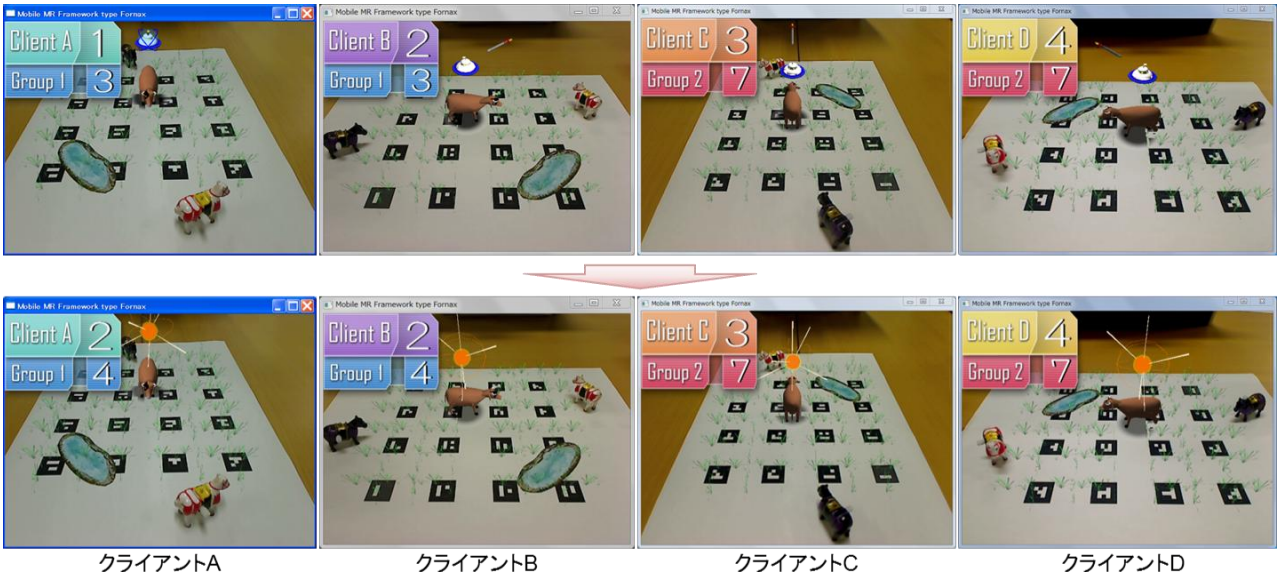

図 7　個別対応型情報提示機構の実験結果