I-069

# 複数カメラによる実世界マーカ読み取り補完システム

## A Detection System for Visual Markers using Multiple Cameras

宮寺 和彦† 岩井 将行‡* 戸辺 義人‡*

Kazuhiko Miyadera Masayuki Iwai Yoshito Tobe

## 1. はじめに

実世界のオブジェクトを，カメラにより認識するアプリケーションにおいて一般的に利用される ARToolKit を使用した時，ビジュアルマーカの全体を 1 台のみのカメラに収めなければならず，一部分がカメラの視野から外れたり，角度が不適切なとき認識できなくなる問題が生じていた．そこで，われわれは，1 台のカメラでは捉えきれないマーカを複数台のカメラを利用して読み取り，ネットワークを介してデータを相互に利用する AReyes を開発した．本稿では AReyes の設計と実装について報告する．

## 2. 拡張現実と ARToolKit の制約

拡張現実とは現実環境にコンピュータを用いて，情報を付加提示する技術である．ARToolKit は拡張現実アプリケーションの実装を助ける C 言語用ライブラリである[1]．コンピュータに接続されたカメラでビジュアルマーカを認識し，マーカの位置や向きなどの情報を計算し，OpenGL でカメラからの画像に 3D オブジェクトを上書きすることを可能にする．そのため，一般的に拡張現実のアプリケーションを作成するライブラリのディファクトスタンダードとなっている．しかし，1 台のみのカメラでビジュアルマーカを認識しなければならず，図 1 のように一部分がカメラの視野外にある場合や，角度が不適切な場合はマーカを認識することが不可能である．

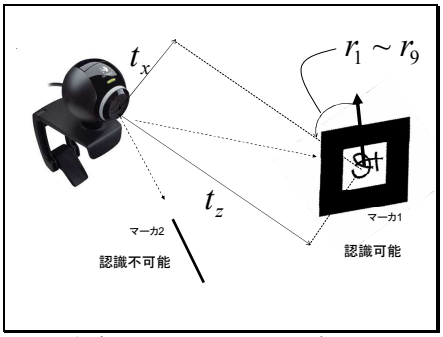


図 1. 角度によるマーカ認識のブラインド問題

ARToolKit で認識したマーカのデータのうち，使用するデータはマーカの中心を中心としたマーカ座標系からカメラ画像の中心を中心としたカメラ座標系へ変換するための 4×4 の変換行列である．変換行列を $P$ とすると

$$P = \left(\begin{array}{ccc:c} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right) \quad \cdots (1)$$

と表すことができる．$r_1$ から $r_9$ までは回転成分，$(t_x,\ t_y,\ t_z)$ は並進成分を表している．並進成分はカメラ画像を中心とした右手系の直交座標で表されている．

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = P \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \quad \cdots (2)$$

式(2)によって，マーカ座標系の $(X_m,\ Y_m,\ Z_m)$ からカメラ座標系の $(X_c,\ Y_c,\ Z_c)$ に変換する．

## 3. AReyes

### 3. 1 AReyes の提案手法

1 つのカメラで捉えることができないマーカを複数のコンピュータとカメラを用いて認識する．

コンピュータの処理をサーバとクライアントに分け，クライアントのコンピュータが認識したマーカのデータはネットワークを介してサーバに送られる．サーバはマーカのデータを各クライアントのコンピュータに送る．

クライアントのコンピュータは自分がマーカを認識できない場合，サーバから送られたデータを用いてマーカを認識する．

### 3. 2 AReyes データ通信方式

クライアントはマーカの ID と回転成分と並進成分をサーバに送信する．送信パケットを $P_1$ とすると

$$P_1 = \{ID,\ r_1,\ r_2,\ r_3,\ r_4,\ r_5,\ r_6,\ r_7,\ r_8,\ r_9,\ t_x,\ t_y,\ t_z\} \quad \cdots (3)$$

と表すことができる．

AReyes のデータ通信は主に 3 つのスレッドから成り立っており，擬似コードで表すと図 2 のようになる．

```
Thread LoopLocalDetect{            // 送信用スレッド
  Loop{
    P1 = DetectMarker();           // マーカの認識
      if(P1 != NULL){              // ローカル認識成功
        marker = selectMarker(P1); // マーカの選定
        Node2.send(P1);            // ネットワーク送信
        marker.updatePosition(P1); // 位置情報の更新
        marker.selfDetect(time());
                                   //最終ローカル認識時間の更新
      }
    }
  }
}

Thread LoopWait{                   // 受信用スレッド
  Loop{
    receive(P2);                   // ポジションの受信
    marker = selectMarker(P2);     // マーカの選定
    if(!marker.selfDetectCheck()){ // ローカル認識失敗の確認
      translate(P2);               // ポジションの行列変換
      marker.updatePosition(P2);   // 位置情報の更新
      marker.remoteDetect(time());
                                   //最終リモート認識時間の更新
    }
  }
}

Thread LoopVisualize{              // 描画用スレッド
  Loop{
    while(EachMarker.m){
      if(m.checkTimer())           // 更新時間のチェック
        visualize(m);              // 描画
    }
    sleep();
  }
}
```

図 2. AReyes 通信アルゴリズム

† 東京電機大学大学院 工学研究科 情報メディア学専攻

‡ 東京電機大学 未来科学部

* JST CREST

## 4. 実験

2台のAMD Athlon 64 X2 Dual Core Processor 4400+ 2.31GHzを搭載したデスクトップPCとカメラを用いた実験を行った．1台のコンピュータにつき1台のカメラを接続し，2台のカメラを図3のように配置した．

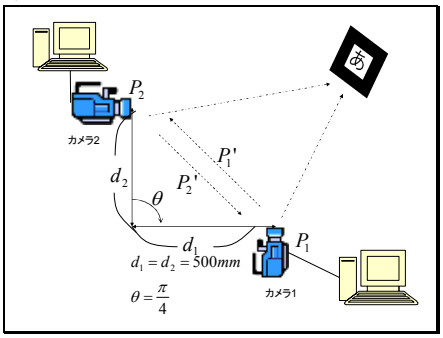


図3.　配置図

図3において，$P_1$，$P_2$はそれぞれのコンピュータが認識したマーカのデータの変換行列を表している．2台のコンピュータはネットワークでつながっている．マーカはカメラ1では認識できないが，カメラ2で認識できる位置に配置している．カメラ1を接続したコンピュータは，クライアントとサーバの処理の両方を実行し，カメラ2を接続したコンピュータはクライアントの処理を行っている．

サーバから送られたデータを用いる時，$P_1$のままマーカ座標系からカメラ座標系に変換すると，カメラの位置と角度の違いにより，正しく表示することができない．そこで，カメラの違いによるデータの差を修正する必要が出てくる．今回はカメラの位置と角度は$d_1$，$d_2$，$\theta$の値で固定しているため，x軸座標に関するデータとz軸座標に関するデータを入れ替えて位置の差を足している．

カメラ1からみたカメラ2の位置の差は$(-500,\ 0,\ 500)$と表されるので，修正された変換行列$P'$は

$$P' = \begin{pmatrix} r_7 & r_8 & r_9 & t_z - 500 \\ r_4 & r_5 & r_6 & t_y \\ -r_1 & -r_2 & -r_3 & -t_x + 500 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \cdots (4)$$

と表すことができる．

図3において，$P_1'$，$P_2'$は修正された変換行列を表し，破線矢印はデータ通信を表している．

図4から図6に実験結果の画像を示す．図4において，左側の画像はマーカの右端が画面外にあるので認識することができないが，右側の画像ではカメラ2からのデータを用いて表示することができている．また，AReyesを使用することにより，図5のようにマーカが全くカメラに写っていない場合でも，表示することができている．図6は実験中のカメラ2からの画像である．AReyesによってカメラ2で取得したマーカのデータを使用して表示することが可能となっていることがわかる．

図4.　AReyesを使用しないカメラ1の画像(左)とAReyesを使用したカメラ1の画像(右)

図5.　マーカの全体が写っていない場合の画像

図6.　カメラ2からの画像

## 5. 今後の展望

ネットワークを介して得たデータを使用して表示するとき，図4の右側や，図5のように静止画では正しく表示されるが，動画にしたとき，点滅して表示された．これは，実験の更新時間のチェックが厳しすぎたため，通信処理が描画処理に追いついていないためである．そのためデータを受信したフレームのみではなく，数フレーム間有効とするなどの解決策が上げられる．また，1400MHzのCPUを搭載したノートPCを用いて実験を実施したところ，CPU稼働率が上昇し，デスクトップPCで実験を行った時のような，正常な動作をすることができなかった．今後は，処理の高速化も考慮に入れる必要がある．

今回の実験ではカメラの位置が固定されているが，今後はカメラを自由に配置し，動かせるようにしたい．

## 6. まとめ

本研究では，ARToolKitにおけるビジュアルマーカの認識の問題を解決するために，複数のコンピュータとカメラを使用してマーカを認識し，ネットワークを介してデータを共有するAReyesを提案しプロトタイプを開発した．マーカ認識の制限を低減することによってARToolKitを利用して作成できるアプリケーションの可能性が広がった．

## 参考文献

[1] Kato, H., Billinghurst, M.: Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99). (1999)

[2] Woods, E., Mason, P., Billinghurst, M.: Magic Mouse: an Inexpensive 6-Degree-of-Freedom Mouse. Proceedings of Graphite 2003. (2003)