

ネットワークを介した複数ロボットの協調制御

Cooperative Control of Multi-Robots via Internet

○滝川 一*, 羅 志偉**, 渡部 慶二*, 遠藤 茂*

○Hajime Takikawa*, Zhi-wei Luo**, Keiji Watanabe*, Shigeru Endo*

*山形大学 大学院 理工学研究科,**理化学研究所

*Yamagata University,**Riken

キーワード：群ロボット、ネットワーク、協調制御、3Dシミュレーション、ロボットサッカー

連絡先：〒992 米沢市城南4-3-16 山形大学 工学部 応用生命システム工学科 渡部研究室

滝川 一, Tel: (0238)26-3178, E-mail: taki@ewata.yz.yamagata-u.ac.jp

1. はじめに

本研究では移動ロボットの動的協調問題について考察し、協調作業としてロボットサッカーを取り上げる。ロボットサッカーとは移動ロボット群が二つのチームに分かれ、サッカーフィールド内を自律的に移動し、互いに点を取り合うというゲームである。このゲームにおいて協調が必要とされる。問題解決の手段として、まずコンピュータ上でのシミュレーションを考え、仮想ロボットサッカー用フィールドを構築して、実験を行う。

2. シミュレータ概要

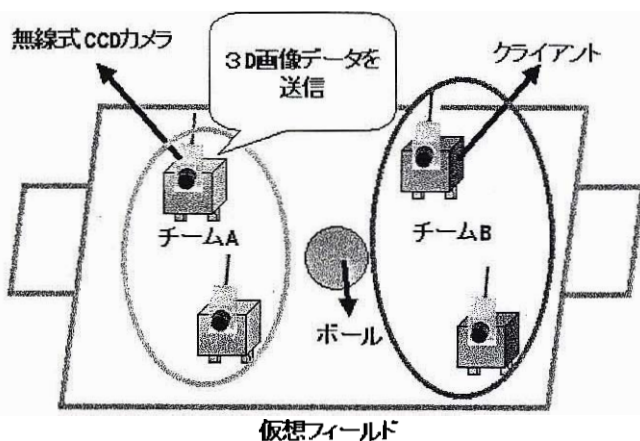


図1. シミュレータ概要図

シミュレータ世界を図1に示す。図1では平面的な世界で表現されているが、シミュレータは3次元データを扱い、表示(ユーザインタフェース)も3Dである。仮想フィールド上をA,B二チームがボ

ールを蹴り合い、点数を競う。フィールド上の移動ロボットを仮にクライアントと呼ぶこととする。クライアントが得られるセンサ情報は、上部に搭載されているカメラによる**視覚情報**である。クライアントは全方向移動可能なロボットとして設計される。また搭載されているカメラは現在固定しているものとして設計され、可動式への変更も可能である。

3. TCP/IPによるサーバ・クライアント通信

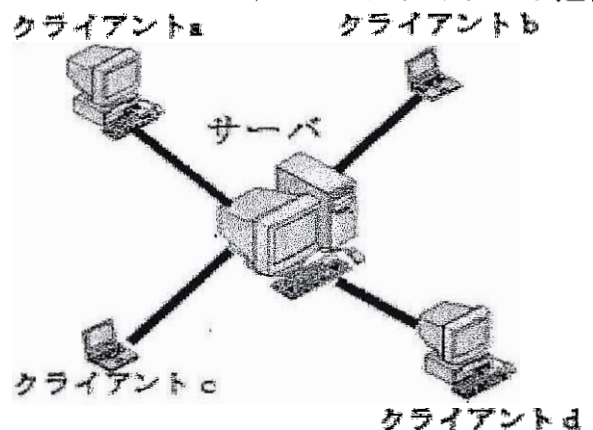


図2. サーバ・クライアントモデル

本研究で作成したシミュレータは、図2のサーバ・クライアント型の通信モデルを用いる。このようなネットワークを用いたモデルでは、処理が分散されることによりシミュレーションがスムーズに進むことが期待できることや、クライアントにロボットの動きを担当させた場合、異なる戦略を用いたり、ユーザが直接操作可能であるといった利点がある。

通信規格としては TCP/IP (Transmission Control Protocol/Internet Protocol) を用いている。これはデータの送受信の際にエラーを起こしにくいプロトコルだが、転送速度の点では UDP(User Datagram Protocol)というプロトコルに劣る。しかし転送データの量が少なくない場合には転送速度は大して問題にならないので TCP/IP を用いる。シミュレータにおける具体的なデータのやりとりについては、6 章で説明する。

4. 3D 仮想空間における物理演算処理

3D 世界における剛体の運動、衝突の計算と表示を正確にシミュレーションに組み込むことは、シミュレーションをよりリアルなものにするために必要なことである。3 次元空間における物体の運動および衝突計算を正確にプログラミングすることは大変な作業である。本研究では、既に開発されている演算モジュールを組み込むことによりシミュレータを作成した。その際利用した演算モジュールは、ネット上で公開されていた『MathEngine』を用いた。

MathEngine の特徴として、仮想 3 次元空間におけるオブジェクトの動作を一定ステップ毎に演算し、ユーザはその結果をレンダリング (3D 描画) されたウィンドウで確認できる。オブジェクトとは、3D 空間において作成された物体のことで、すべて剛体として取り扱う。剛体とは、物体同士の衝突の際に発生する塑性変形が全く起こらない物体ということである。MathEngine は 3 つの基本モジュール: Dynamics(運動), Collision(衝突), Viewer(3D 表示)で構成されている。Dynamics モジュールは、オブジェクトの運動演算を行い、次ステップにおけるオブジェクトの位置と姿勢を算出する。Collision モジュールは各剛体間の衝突を判定する。Viewer モジュールは 3D 世界をユーザが目で見分けるように、レンダリングする。

MathEngine は以上の 3 つの機能を内包しているので、プログラマが簡潔でプログラムを構築できる。具体的には、それぞれのモジュールに対する初期化、必要なオブジェクトの作成配置、各設定などである。その際、重力や摩擦なども設定するだけで簡単に実現できる。

5. 3次元空間表現

実際の 3 次元空間シミュレーションの説明に入る前に、3 次元空間座標系について説明する。

図 3 に示す 3 次元空間において、大別して 3 つの座標系を用いる。1 つ目がワールド (世界) 座標系で、基準となる座標系である。2 つ目がオブジェクト座標系で、オブジェクトそれぞれに対して与えられるものである。3 つ目はカメラ座標系であり、レンダリングという 3 次元空間をユーザに対し可視化するためのものである。ユーザが 3D 空間上のオブ

ジェクトを操作する場合やカメラ自身を動かしたい場合、このカメラ座標系を基準とし、命令を与えることとなる。

オブジェクト座標系

ワールド座標系

図 3. 3次元世界の3つの座標系

このような 3D 空間を構築する上で座標変換が重要である。具体的に、x-y-z 軸周りの回転、平行移動などである。X 軸周りに θ の回転を例に取ると、

$$\begin{aligned} x' &= x \\ y' &= y \cos \theta - z \sin \theta \end{aligned} \quad (1)$$

$$\begin{aligned} z' &= y \sin \theta + z \cos \theta \end{aligned}$$

のようになる。これを行列で表すと、

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{pmatrix} \quad (2)$$

となる。(2)式の 3*3 行列は回転行列 (R) で、x-y-z 軸周りで回転行列は、オブジェクトがワールド座標に対してどの程度傾いているのかを記述する上で重要となる。一般的には回転行列 (R) は x-y-z の 3 軸それぞれの周りでものを掛け合わせたものとなっている (R_{xyz})。

一方、平行移動は

$$\begin{aligned} x' &= x + dx \\ y' &= y + dy \end{aligned} \quad (3)$$

$z' = z + dz$ のように表せ、行列の形にするために、4*4 行列

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} dx \\ dy \\ dz \\ 1 \end{pmatrix} R \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (4)$$

に拡張して表現する。

(4)式は回転 (R) と、平行移動を合成した行列表現

で、(4)式の行列を同次変換行列(TM:Transformation Matrix)という。これによりオブジェクトの位置と姿勢が表現でき、シミュレータ内部でもこれをもとに3D処理が行われている。

6. シミュレーション処理の流れ

以上の技術を用いて本シミュレータは以下に設計されている。まずサーバとクライアントの処理タスクについて説明する：

- サーバは、①他のクライアントからの通信処理、②一定ミリ秒ごとにクライアントからの移動要求に基づいてロボット位置を更新(運動演算)、③衝突演算の3つが主な仕事である。またサーバはクライアントに結果を伝えるために、すべての移動可能なオブジェクトの変換行列(TM)を送信する。これによりクライアントは正確に環境を再構築できる。
- クライアントは、①サーバにロボットの移動要求を伝えること、②サーバからの結果情報を受け取ること、③学習パラメータなどを更新するようになっている。また、双方に共通する処理として、仮想空間がユーザに見えるように3次元表示を行う。

このように、現在は2チーム合計8台のクライアントを仮想空間上に配置している。クライアントは単純に箱型ロボットモデルを用いている。

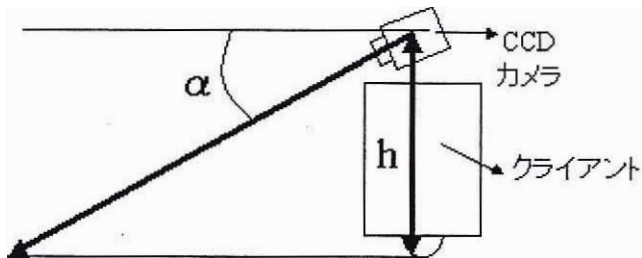
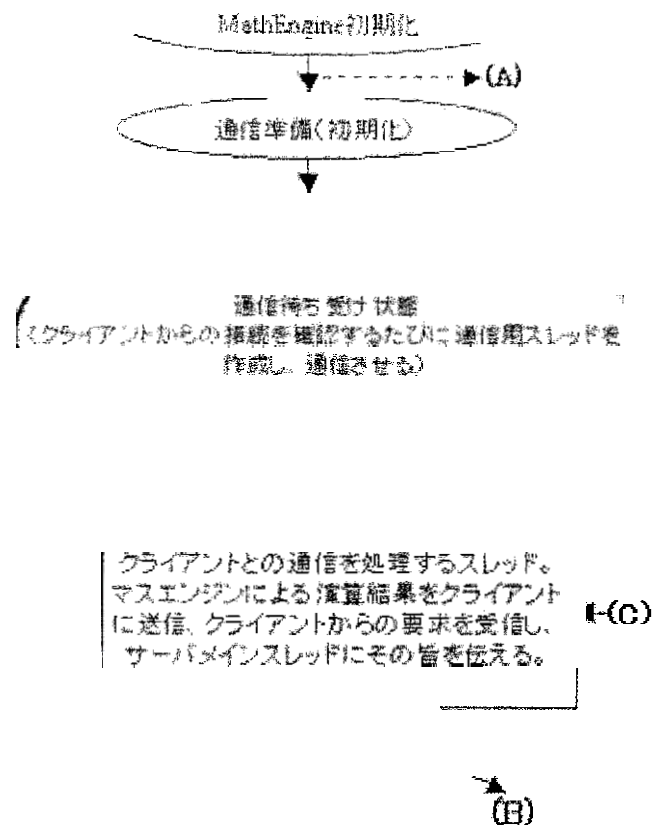


図4. クライアント箱型ロボットモデル

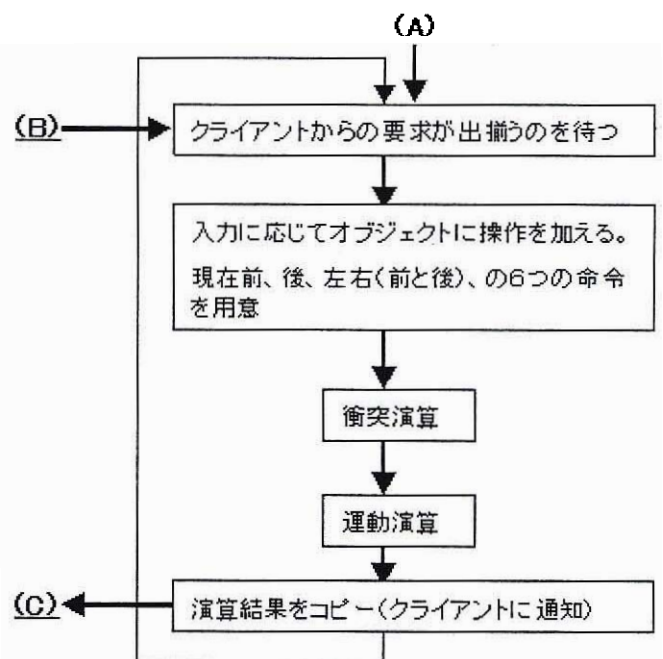
図4のように、クライアントのカメラは基準面からhの高さに、地面に対して角度 α だけ傾けて固定されていると仮定している。また現状のシミュレータでは、クライアントロボットはユーザがキーボード操作により動かせるようになっている。

以下では、本シミュレータの具体的な処理アルゴリズムについて述べる。

サーバ側フローチャート



<1>初期化と通信



<2>シミュレーションループ

まず、このプログラムで使用するマルチスレッドについて簡単に述べる。シングルスレッドのプログラム(通常のプログラム)処理は、流れに沿って逐次行われる。一方、マルチスレッドでは、複数の処理を(見かけ上)並列に行うことができ

る。これによりこのプログラムのように、クライアントからの通信を待ちつつ、シミュレーション処理を進める並列処理が可能となる。この際プログラム開始時のスレッドをメインスレッドと呼ぶこととする。

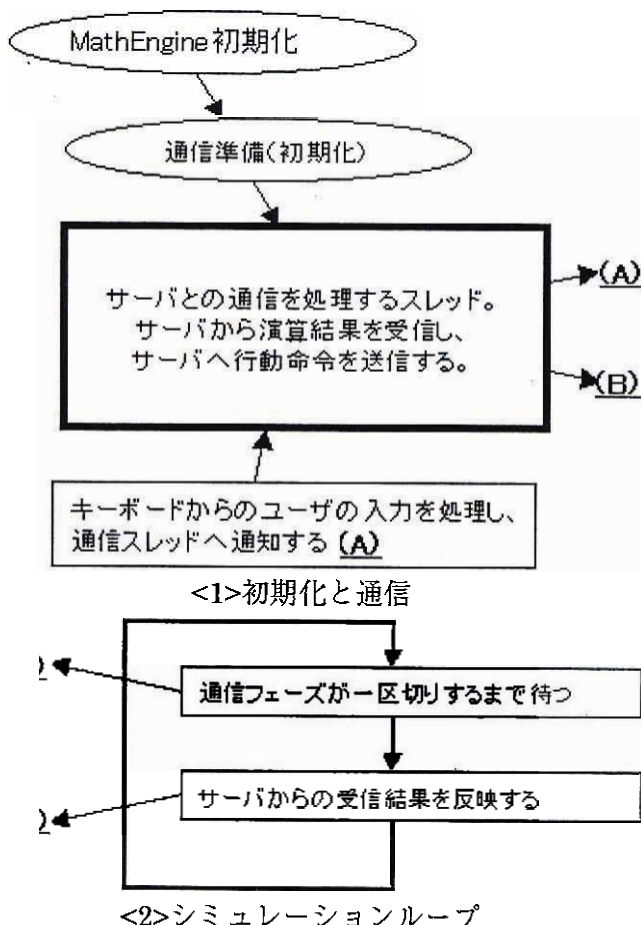
<1>の初期化と通信では、MathEngine および通信関連の初期化を行っている。(A)はメインスレッドの流れである。つまり通信関連は別スレッドに行わせている。(B)は各クライアントとの通信をやりとりするサーバ側のスレッドが、クライアントから受信した情報である（この場合は、各ロボットに対する行動要求である）。また、(C)はクライアントからの行動要求を元に行った物理演算結果であり、以下のようなフォーマットで定義される。

クライアントID | Transformation Matrix(ボール) TM(クライアント)×8

これがすべてのゲームに参加中のクライアントに対して送信される。ちなみにシミュレーションにおける物理演算の時間間隔は 200ms と設定している。

<2>のシミュレーションループでは、メインスレッドはループ処理を開始し、<1>で作成された通信スレッドと相互に情報を交換しながらシミュレーションが進行する。

ロ クライアント側フローチャート



クライアント側の処理では、まず、<1>の初期化と通信で、サーバとの通信用スレッドをひとつ作成し、通信を行う。また、メインスレッドは<2>のようなループを行う。(A)はクライアントがサーバに送信するロボットの行動要求情報で、(B)はサーバから受信した演算結果である。これによりすべてのクライアントが同じ環境を共有できることとなる。

7. シミュレータの動作検証

本研究で設計したシミュレータを実際に動かしてみたときのサーバ画面を図5に示す。

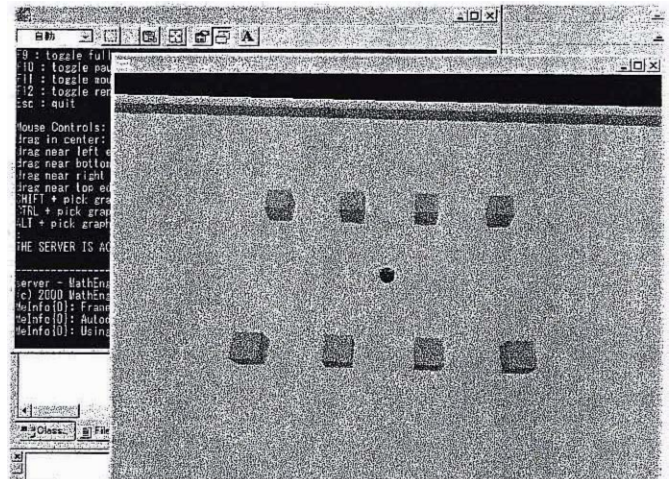


図5. シミュレータ画面

またクライアントのロボット一台から見た画面を図6に示す。

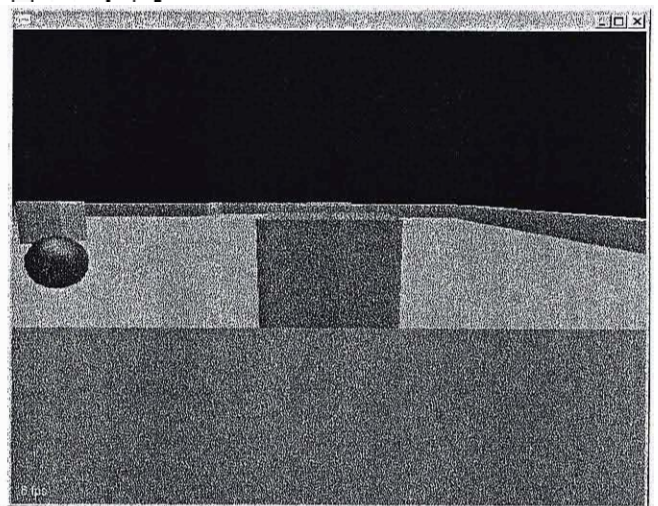


図6. クライアントシミュレータ画面

結果としてTCP/IPを用いたサーバ・クライアントシステムでスムーズに通信を行い、サッカーシミュレーションが進行可能であることが確認できた。また運動、衝突などの複雑な物理演算がほぼ正確に処理されていることも確認できた。