解説

RoboCup サッカー 2D シミュレーションリーグ 解説：仕組みと環境構築†

秋山 英久*

1. はじめに

本稿では，RoboCup サッカーシミュレーションリーグの一部門である2Dリーグについて解説する．本連載では，サッカーシミュレーションのもう一つの部門である3Dリーグの解説が前回までに掲載されている[4, 5]．2Dリーグと3Dリーグの大まかな違いについてはそちらを参照してもらいたい．

2Dリーグの解説は3Dリーグと同様に以下の2編で構成する予定である．

- シミュレータの仕組みと環境構築
- 競技会への参加，チーム開発の手引き

2. 2D サッカーシミュレータの概要

2.1 2D サッカーシミュレータとは

2D サッカーシミュレータは，その名が示すとおり2次元平面上でのサッカーシミュレーションを実現するソフトウェアである．計算機上に仮想的なフィールドを用意してボールやプレイヤーの物理運動をシミュレーションすることは3Dリーグと変わらないが，ボールやプレイヤーが空中を飛ぶことはない．

図1 2Dサッカーシミュレータの実行画面

† RoboCup Soccer 2D Simulation League : Introduction to the RoboCup Soccer Simulator
Hidehisa AKIYAMA

* 福岡大学工学部
Fukuoka University Faculty of Engineering

図1はシミュレータの実行画面を示している．2Dサッカーシミュレータでは，ボールやプレイヤーはシミュレータ上では円で表現されている．他にもさまざまな点で簡略化，抽象化されたモデルが採用されており，現実のデバイスや物理現象を忠実に再現しようとはしていない．ロボットやセンサの制御技術そのものには注目しておらず，マルチエージェントによるチームワーク研究に焦点を合わせているのが2Dリーグの特徴である．

2.2 シミュレータの歴史

2Dリーグはロボカップ発足当初から存在している最古参のリーグである．競技内容でも，1996年当初から11対11でのほぼ人間のサッカーのルールによるゲームが実現されている唯一のリーグである．2Dサッカーシミュレータは1993年に電子技術総合研究所（現在は産業技術研究所に統合）の野田五十樹氏によって最初のバージョンが開発された[1]．以後，さまざまな人の手による改良を経て，2011年時点ではバージョン15がリリースされている．シミュレータ一式はオープンソースソフトウェアとして公開されており[2]，サッカーシミュレーションリーグのMaintenance Committeeによって保守されている．

シミュレータのメジャーバージョンは毎年上げられることが通例となっている．RoboCup世界大会終了後に次年度のTechnical Committeeが組織され，ルール改正議論が進められる．競技レベルの進歩に伴ったルール変更だけでなく，新規研究テーマを発掘するための仕様追加が続けられており，今後も毎年のメジャーバージョンアップは続くと思われる．

一方で，2Dリーグでは後方互換性の維持も重視されており，設定ファイルを修正するだけで過去のチームプログラムをほぼそのまま利用することができる．過去のソフトウェア資産を利用しやすいことも2Dリーグの特徴である．

3.4 時間モデル

2Dサッカーシミュレータは離散時間モデルを採用しており、すべての物体の位置や速度などの情報はシミュレーションサイクル更新時に一斉に更新される。標準の設定では、サイクルは100ミリ秒に一回更新される。サーバはプレイヤーからのコマンド送信を待たずにサイクルを進めるため、プレイヤーはサイクル更新に遅れないようにコマンド送信を完了させなければならない。

3.5 運動モデル

2Dサッカーシミュレータでは、非常に簡略化された運動モデルが採用されている。物体と地面との摩擦係数は考慮されておらず、速度減衰率というパラメータで置き換えられている。

シミュレーションサイクル更新時、ボールまたはプレイヤーの移動の計算は以下に行われる：

$$\begin{aligned} (u_x^{t+1}, u_y^{t+1}) &= (v_x^t, v_y^t) + (a_x^t, a_y^t): \text{加速} \\ (p_x^{t+1}, p_y^{t+1}) &= (p_x^t, p_y^t) + (u_x^{t+1}, u_y^{t+1}): \text{移動} \\ (v_x^{t+1}, v_y^{t+1}) &= \text{decay} \times (u_x^{t+1}, u_y^{t+1}): \text{速度減衰} \\ (a_x^{t+1}, a_y^{t+1}) &= (0, 0): \text{加速度リセット} \end{aligned} \quad (1)$$

(p_x^t, p_y^t) と (v_x^t, v_y^t) はそれぞれサイクル t における物体の位置と速度、 decay は物体ごとに設定されている速度減衰率である。 (a_x^t, a_y^t) は物体の加速度で、プレイヤーのコマンドによって自分自身やボールに対して与えられ、サイクル更新後は常に0へリセットされる。

式1で速度ベクトルが更新される際に、さらにノイズが加えられる：

$$\begin{aligned} (r_x^t, r_y^t) &= (\tilde{r} \cos \theta_{rand}, \tilde{r} \sin \theta_{rand}) \\ (u_x^{t+1}, u_y^{t+1}) &= (v_x^t, v_y^t) + (a_x^t, a_y^t) + (r_x^t, r_y^t) \end{aligned}$$

$\tilde{r}$ は、 $\vec{v}^t$ の大きさと物体のノイズパラメータ rand に応じて発生する一様な乱数である。 θ_{rand} は $[-2\pi, 2\pi]$ の一様乱数である。 $\tilde{r}$ の分散 rmax は以下のように求められる：

$$\text{rmax} = \text{rand} \cdot |(v_x^t, v_y^t)|$$

物体同士の衝突のモデルは非常に単純である。サイクル更新時に物体が重なっていれば、物体が互いに重ならない位置まで移動させられ、それぞれ速度が-0.1倍されるだけである。衝突モデルの不自然さについてはしばしばルール改正議論に上っており、今後、妥当な衝突モデルが提案されれば変更されるかもしれない。

物体の移動位置の予測では、移動に伴うノイズ、物

体同士の衝突、観測誤差などを考慮しなければならない。さらに他のプレイヤーは自分の意思で加速度を得られるため、物体の移動位置の予測を精度良く行うことは簡単ではない。

3.6 センサモデル

2Dサッカーシミュレータではセンサ情報も抽象化されている。サーバからは、`see`, `hear`, `sense_body` という三つのセンサ情報メッセージがプレイヤーへ自動的に送信されるようになっている。

3.6.1 seeメッセージ

`see` メッセージによって、他の物体(ボール、プレイヤー、ランドマーク)の識別情報、距離、角度、相対速度などの情報を得られる。デフォルト設定では、`see` メッセージはシミュレーションサイクルと非同期にサーバから送信される。`synch_see` コマンドを実行することで、各サイクルの開始直後に `see` メッセージを受信することができるようになる。

プレイヤーはそれぞれ視界を持っており、視界の範囲外の物体は観測できない。さらに、観測対象物体との距離に応じて距離情報は量子化され、正確な位置を得られにくくなる。プレイヤーの識別情報(チーム名、背番号)も距離が遠くなるほど観測しにくくなる。このような不完全な情報からプレイヤーは世界の状態を再構築しなければならない。

3.6.2 hearメッセージ

`hear` メッセージによって、審判からのプレイモード通知、他のプレイヤーが `say` コマンドで喋った内容、オンラインコーチからのアドバイス、を聞くことができる。

他のプレイヤーが `say` コマンドで喋った内容は、コマンドが受信された次のサイクル開始直後にプレイヤーへ送信される。ただし、一サイクルに聴ける情報はひとつのみである。そのため、プレイヤー間での `hear` メッセージによるコミュニケーションを試みる場合、闇雲に全員が喋っているのは重要な情報を得られにくくなる。今誰が喋るか、どのプレイヤーからのメッセージに注目するかといった情報共有を効率的に行うための工夫が必要となる。

3.6.3 sense_bodyメッセージ

`sense_body` メッセージはプレイヤーの状態を示すさまざまな情報を含む。現在の自分の速度、首の角度、他の物体との衝突状態といった情報だけでなく、送信したコマンドがサーバに受け付けられた回数などの補助的な情報も得られる。

3.7 プレイヤの制御コマンド

2Dサッカーシミュレータではプレイヤの制御は少数のコマンドによって抽象化されている。これらのコマンドは、ロボットの設計、例えば車輪型や二足歩行型、によって大きく異なるものとなる運動制御を隠蔽し、サッカーに必要な最低限の行動コマンドとして選択されたものである。プレイヤの体を制御するコマンドは1サイクルに1回しか使用できないため、ドリブルなどの一定時間の継続が必要な動作を実現するにはコマンド列を適切にプランニングする必要がある。

以下でプレイヤの制御コマンドを簡単に説明する。これらのコマンド以外にもプレイヤの初期化や設定変更のためのコマンドがいくつか存在するが、本稿では扱わない。

3.7.1 kickコマンド

ボールがプレイヤのキック可能範囲に存在する場合、プレイヤはkickコマンドでボールに加速度を与えられる。ボールの中心位置とプレイヤの中心位置との距離がプレイヤごとに定められた値以下であれば、プレイヤはボールをキック可能となる。一回のキックで与えられる加速度の大きさはボールとの相対的な位置関係によって変化する。プレイヤごとにキックの精度は異なり、また、ボールのスピードが大きいほどキック動作に加えられるノイズは大きくなる。

ボールがプレイヤのキック可能範囲に存在する限り、プレイヤは連続してボールに加速度を与えることができる。その結果、複数回キックでボールを加速させることが可能となっている。複数のプレイヤによるキックが同時に発生した場合は、すべての加速度の合成ベクトルが最終的な加速度となる。ただし、加速度の大きさの最大値が設定されており、その値以上の加速度がボールに与えられることはない。ボールのスピードの最大値も設定されているため、ボールスピードが無制限に大きくなることはない。

このモデルでは、ボールを強く蹴り出すためには2～3回の連続キックでボールを加速させなければならない。また、複数のプレイヤが同時にボールをキックすると加速度が打ち消されてボールが止まってしまう現象も発生する。このような現実世界ではありえないキックモデルは不自然であるとの意見が多いものの、妥当な代替モデルの導入が難しく現在も使用されている。

3.7.2 dashコマンド

dashコマンドはプレイヤ自身に加速度を与えるコマンドで、引数としてダッシュパワーと加速度の方向

を取る。加速度の方向がプレイヤの体の向きと一致しない場合はその角度差に応じてダッシュの効果が減少する。負のダッシュパワーが与えられると、加速度の方向は逆向きになる。rcssserverバージョン15では加速度の方向は8方向へ限定されているが、今後は完全な全方位移動が導入される予定である。

dashコマンドが実行されると、指定したダッシュパワーの分だけプレイヤのスタミナが減少する。ダッシュパワーが負の場合はその絶対値の2倍のスタミナが消費される。プレイヤのスタミナに関して、スタミナとスタミナキャパシティの2つのパラメータを特に考慮しなければならない。dashコマンドで消費されたスタミナは常に回復しているが、スタミナを回復する際にスタミナキャパシティが消費され、スタミナキャパシティが尽きるとスタミナは回復しなくなる。このように、短期的、長期的の両面でのスタミナ管理が要求されるようになっている。

3.7.3 turnコマンド

turnコマンドは引数として回転角度を取り、プレイヤ自身の体の向きを変更する。現在のプレイヤのスピードに応じて回転できる最大角度が制限される。回転動作にはノイズが加えられるため、指定した回転角度どおりの結果は得られない。

3.7.4 catchコマンド

catchコマンドはキーパープレイヤ専用のコマンドで、ペナルティエリア内でボールをキャッチすることができる。引数として体の向きからの相対方向を取り、その方向の一定範囲内にボールが存在する場合にキャッチ動作が作用する。ボールとプレイヤとの距離に応じて、確実にキャッチが成功する範囲とキャッチ成功確率が設定される範囲が存在する。確率的な判定でキャッチが失敗した場合、ボールはキャッチコマンドで指定された方向に対して跳ね返るような加速度を与えられる。

3.7.5 tackleコマンド

tackleコマンドはプレイヤの体の向いている方向の一定範囲で作用し、キックのようにボールに加速度を与える。方向を限定すればタックルの有効範囲はキックやキャッチよりも大きいため、ボールを保持している敵プレイヤの妨害や、キャッチできないボールに対してキーパーがタックルを試みるという使い方ができる。ただし、ボールとの位置関係に応じてタックルの成功確率が設定されており、タックルの有効範囲にボールが存在しても確実にボールを蹴られるわけでは

ない。また、タックル実行後は一定時間動けなくなるというペナルティが設定されている。

タックルにはファウルの概念が導入されており、意図的にファウルを実行することもできる。意図的なファウルではタックルの成功確率が上昇するかわりに、審判にファウルを取られやすくなる。背後からのファウルのような危険な状況での意図的ファウルにはイエローカードが与えられることもある。

3.7.6 moveコマンド

moveコマンドはフィールドの任意の位置へ瞬間移動する特殊なコマンドである。このコマンドは、試合開始前、ゴールが入った直後、ボールをキャッチした直後のキーパー、といういずれかの条件でしか使用できない。

3.7.7 turn_neckコマンド

turn_neckコマンドはプレイヤーの視界の方向を変更する。プレイヤーは体の向きに対して-90度から+90度の範囲に首を向けることができる。turn_neckコマンドは1サイクルに一回しか実行できないが、kick, turn, dash, tackleなどの体を制御するコマンドと同時に使用できる。

3.7.8 change_viewコマンド

change_viewコマンドはプレイヤーの視界の広さを変更する。change_viewコマンドは1サイクルに何度でも使用でき、その効果が即座に現れる。プレイヤーの視界の広さを変更することでseeメッセージの受信間隔が変化する点に注意しなければならない。視界が広がるほどseeメッセージの受信間隔が大きくなるというトレードオフが設定されており、視覚情報による情報収集を効率的に行うためには、turn_neckコマンドとchange_viewコマンドを駆使しなければならない。

3.7.9 pointtoコマンド

pointtoコマンドはフィールド上の特定位置を指さすコマンドである。pointtoコマンドが実行されると、プレイヤーは一定時間その位置を指さし続ける。他のプレイヤーは指さしの方向のみを視覚情報として得られる。

3.7.10 attentiontoコマンド

attentiontoコマンドは特定のプレイヤーへ注意を向けるコマンドである。通常、複数のプレイヤーがsayコマンドを実行した場合、その中からランダムに選択されたものがhearメッセージとしてプレイヤーへ送信され

る。attentiontoによってこのランダム性を除去し、特定のプレイヤーからのメッセージを確実に受け取ることができるようになる。

3.8 ヘテロジーニアスプレイヤー

シミュレータの起動時に能力が異なる18種類のヘテロジーニアスプレイヤーが生成される。チームはこの中から11体を選び、各プレイヤーに割り当てなければならない。この割り当て処理は、オンラインコーチがchange_player_typeコマンドによって行う。

change_player_typeコマンドはプレイヤー交代にも相当しており、交代されたプレイヤーはスタミナやファウルの状態がリセットされる。試合終盤でスタミナが尽きたプレイヤーを交代するなどといった戦術が可能となっている。

4. 2Dサッカーシミュレーションの環境構築

2Dサッカーシミュレータは以下の3つのソフトウェアで構成されている。

- rcssserver
- rcssmonitor
- rcsslogplayer

これらのうち、シミュレーションの実行に最低限必要となるのはサーバプログラムであるrcssserverである。rcssmonitorはフィールドの状態を可視化するためのビューワ、rcsslogplayerは記録した試合を再生するためのログプレイヤーである。rcssmonitorとrcsslogplayerに関しては非公式のソフトウェアで代替することも可能である。次稿でより高機能なビューワを紹介する。

サッカーシミュレーター一式は移植性を考慮して開発が進められており、Linux, Mac OS X, FreeBSD, Windowsといったほとんどの環境での動作が確認されている。本稿ではUbuntu Linuxへのインストール手順を説明する。Ubuntuのバージョンは10.04以降を想定している。32ビット、64ビットのいずれでもシミュレータの動作には問題ない。

4.1 シミュレータの入手

シミュレータの各ソフトウェアパッケージはシミュレータの公式配布サイト[2]から入手可能である。本稿執筆時点では、

- rcssserver-15.0.1.tar.gz

- rcssmonitor-15.0.0.tar.gz
- rcsslogplayer-15.0.0.tar.gz

の3つのファイルをダウンロードすればよい。より新しいバージョンがリリースされていれば、新しいバージョンのダウンロードが推奨される。

4.2 依存ライブラリのインストール

2D サッカーシミュレータをコンパイル・インストールするには、幾つかの依存ライブラリを先にインストールしなければならない。Ubuntuであれば、以下のコマンドを実行するだけでよい：

```
$ sudo apt-get update
$ sudo apt-get install build-essential flex
bison autoconf automake libtool
libboost-all-dev libqt4-core libqt4-dev
libqt4-gui libqt4-opengl libqt4-opengl-dev
libqt4-network libaudio-dev
libfontconfig1-dev libfreetype6-dev
libglib2.0-dev libpng12-dev libxext-dev
libxt-dev libxrender-dev
```

4.3 各パッケージのインストール

各パッケージのインストール手順は共通で、以下のようになる：

```
$ tar zxvf rcssserver-15.0.1.tar.gz
$ cd rcssserver-15.0.1
$ ./configure
$ make
$ sudo make install
```

各パッケージのインストール作業を行う際、作業場所の上位のディレクトリ名に空白が含まれないように注意する。空白が含まれているとconfigureスクリプトを実行する際にエラーが発生する。日本語名も使わないほうが良い。

何も指定しないと、各ソフトウェアは/usr/local以下へインストールされる。インストール先の変更が必要であれば、'--prefix' オプションによって指定することができる。例えば、ホームディレクトリ直下のlocalというディレクトリにインストールしたければ以下のように指定する：

```
$ ./configure --prefix=$HOME/local
```

configureスクリプトには他にもさまざまなオプションが用意されており、コンパイルとインストールの条

件を指定することができる。'--help' オプションを付けて実行すると、利用可能なオプションが表示される。

4.4 シミュレータの実行

rcssserverとrcssmonitorのインストールが完了すれば、シミュレータを実行して画面表示できるようになる。それぞれ単体でも起動できるが、端末で以下のコマンドを実行することでサーバとビューワを同時に起動することができる：

```
$ rcsoccersim
```

インストールに成功していれば図1のようなウィンドウが現れ、黒く塗りつぶされたプレイヤが画面上部に表示される。rcssmonitorのメニューからKick Offを選ぶと試合が開始する。rcssmonitorを閉じるとrcssserverも自動的に停止する。rcssserverを単体で起動した場合は、端末でCtrl-Cをタイプして手動で停止させなければならない。

シミュレータを終了させると、rcsoccersimを実行したディレクトリに拡張子がrcg、rclというファイルが生成されているはずである。rcgファイルが試合を再生するためのログファイル、rclファイルはクライアントから送信されたコマンドを記録したログファイルである。rcsslogplayerで試合を再生するには、rcgファイルを引数で与えて起動すれば良い：

```
$ rcsslogplayer xxxxxxxx.rcg
```

rcssserverを一度起動すると、ホームディレクトリ直下に設定ファイルを格納する.rcssserverというディレクトリが生成されている：

```
$ ls ~/.rcssserver
CSVsaver.conf player.conf server.conf
```

ディレクトリ内にいくつかの設定ファイルが生成されていることが確認できるはずである。player.confはヘテロジーニアスプレイヤに関する設定ファイル、server.confは全般的な設定ファイルである。各設定ファイルにさまざまなパラメータが記述されており、シミュレーションの設定を変更することができる。これらのパラメータは、rcssserver起動時にコマンドラインオプションとして与えることもできる。

4.5 過去のチームバイナリ、ログファイルの入手

シミュレータを起動しただけでは試合を実行することができないので、動作するチームを入手しなければならない。過去の競技会に参加したチームのLinuxバ

イナリが公開されており, 利用可能である[3]. 展開して起動スクリプトを実行するだけで簡単に動作させられるチームが幾つかあるので(例えばFCPortugalなど), 自分の環境で動作するチームを見つけ試してみたい。

データリポジトリ[3]では, 過去の競技会のログファイルも公開されている。rcg.gzという拡張子のファイルをダウンロードし, rcsslogplayerで開けば試合を再生することができる。

5. まとめ

本稿ではRoboCupサッカー2Dシミュレーションリーグのシミュレータの概要と, その導入手順について解説した。次稿にて, 実際に自分のチームを開発するための手順について解説する。

2Dサッカーシミュレータは抽象シミュレータであるものの長い歴史の中で仕様が複雑になっており, 本稿だけでは全容を解説しきれていない。より詳細な解説書[7]などの日本語ドキュメントも存在しているので, それらを活用してサッカーシミュレーションリーグへの参加を検討していただければ幸いである。

参 考 文 献

- [1] Itsuki Noda, Hitoshi Matsubara, "Soccer Server and Researches on Multi-Agent Systems", Proceedings of IROS-96 Workshop on RoboCup, pp.1-7, 1996.
- [2] The RoboCup Soccer Simulator,
<http://sourceforge.net/projects/sserver/>
- [3] RoboCup Soccer Simulation League Official Data Repository,
<http://www.socsim.robocup.org/files/2D/>
- [4] 中島智晴, 中邨陽介, "RoboCupサッカー3Dシミュレーションリーグ解説: 仕組みと環境構築", 知能と情報, Vol.23, No.3, pp.312-319, 2011.
- [5] 中島智晴, 中邨陽介, 炭谷剛志, "RoboCupサッカー3Dシミュレーションリーグ解説: 競技会の概要とエージェント開発", 知能と情報, Vol.23, No.4, pp.612-619, 2011.
- [6] サッカーサーバマニュアル (日本語版),
http://sourceforge.jp/projects/rctools/releases/?package_id=3081
- [7] 秋山英久, "ロボカップサッカーシミュレーション2Dリーグ必勝ガイド",
http://sourceforge.jp/projects/rctools/releases/?package_id=10680, 秀和システム, 2006.
(2011年8月29日 受付)

[問い合わせ先]

〒814-0180 福岡市城南区七隈8-19-1

福岡大学工学部電子情報工学科

秋山 英久

TEL: 092-871-6631 6413(内線)

E-mail: akym@fukuoka-u.ac.jp

著 者 紹 介

あきやま ひでひさ
秋山 英久 [非会員]

2008年東京工業大学大学院総合理工学研究科知能システム科学専攻博士後期課程修了。2008年独立行政法人産業技術総合研究所情報技術研究部門特別研究員, 2011年福岡大学工学部助教, 現在に至る。博士(工学)。マルチエージェントシステム, マルチロボットシステムの研究に従事。人工知能学会, 情報処理学会会員。