# 「サッカーシミュレーションリーグ」

第2回

■ 秋山英久（産業技術総合研究所）

## サッカーシミュレーションリーグの概要

サッカーシミュレーションリーグはロボカップの中でも最も歴史の古いリーグである．1996年のプレロボカップから競技が行われており，2010年は15回目の世界大会となった．2010年現在，サッカーシミュレーションリーグには，2Dリーグと3Dリーグの2つのサブリーグが存在している．本稿では，特に2Dリーグについて解説する．

### 2Dリーグ

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．競技内容においても，1996年当初から11対11による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．使用されているシミュレータの歴史は古く，その成り立ちは1996年のプレロボカップよりもさらに遡る．ロボカップの公式シミュレータとして採用されて以降，ルール変更に伴う機能追加が行われてきたが，基本的な設計はほとんど変更されていない．2Dリーグのシミュレータは，研究プラットフォームとしてはほぼ完成されていると言えるだろう．

2Dリーグのシミュレータは，2次元の仮想フィールド上でサッカーを行う競技として設計されている（図-1）．ボールやプレイヤなどの物体はすべて2次元平面上でモデル化されており，ボールを空中に飛ばしたり，プレイヤがジャンプすることはできない．さらに，プレイヤの物理的な形状やアクチュエータは簡略化されており，現実の人間やロボットとは大きく異なったものとなっている．プレイヤの基本的な行動制御コマンドはある程度抽象化されており，プレイヤの体の制御はkick，dash，turnなどのコマンドによって実現する仕様となっている．すなわち，ロボットの設計，たと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

図 -1　2D サッカーシミュレータの実行画面

図 -2　3D サッカーシミュレータの実行画面

として抽象化されている．このような抽象化を行う理由は，2D リーグではロボットの詳細な運動制御を目的としておらず，あくまでチームワーク研究のテストベッドとしての利用を想定しているためである．このような抽象化によって，現実の運動制御との整合性は必ずしも取れていないものの，サッカーとしてのリアリティを実現することにある程度成功している．抽象シミュレータとしての設計方針は当初から一貫しており，今後のルール改正においても変わることはないだろう．

もう 1 つの重要な方針は，後方互換性の維持である．互換性を維持することで，過去の競技会に出場した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☆1．過去のチームプログラムを使用する最大の利点は，パフォーマンス改善の指標となる点である．たとえば，ある手法を適用した結果パフォーマンスが改善されたかどうかを定量的に評価することは，サッカーというゲームでは難しい場合が多い．過去の競技会で使用されたプログラムと対戦させることで，少なくともチームの強さという指標に関しては相対的な評価が可能となる．また，競技参加者によってさまざまなライブラリやツールが開発，公開されており，多くのソフトウェア資産を再利用できる点も重要である．過去のソフトウェア資産が再利用可能となることで技術進歩が加速されやすくなっており，競技レベルは順調に向上し続けている．

一方で，後方互換性の維持には弊害も存在する．新しい機能を追加する際に慎重な議論を要するだけでなく，要望が実現に至らない場合が多い．たとえば，ボールのみを空中に浮かせる 2.5 次元モデルを導入してはどうかという意見は当初から上がっているが，互換性維持が困難という理由で実現には至っていない．

## 3D リーグ

前述のとおり，2D リーグのシミュレータには高さの概念がなく，ロボットのモデルも抽象化されている．ロボット技術との連携を促進するために，より現実的な 3 次元シミュレーションの必要性を主張する意見は，ロボカップ開始当初の段階から聞かれた．しかしながら，2D リーグのシミュレータの設計では 3 次元シミュレーションを実現することが困難であるため，新たな枠組みによる 3 次元シミュレータが提案された（図 -2）[4]．

3D リーグのシミュレータは，物理エンジンによる厳密な 3 次元シミュレーションを実現するだけでなく，さまざまなロボットモデルをシミュレータ上で利用可能とすることを重要視している．3D リ

☆1　競技参加者には，開発したチームプログラムバイナリを競技会終了後に公開することが義務付けられている．

ーグは2004年から開始され，7回の世界大会が開催されている．しかし，その間にシミュレータの仕様の変更，ロボットモデルの変更が繰り返された．3Dリーグ開始当初は球体のエージェントが採用され，2Dリーグに近い抽象化された行動制御コマンドが用意されていた．球体エージェントは2Dリーグからの発展形として受け入れやすいものであり，2Dリーグで培われてきた方法論をそのまま適用することができた．ところが，2007年からはヒューマノイド型のロボットモデルが公式エージェントとして採用されることとなり，シミュレータの仕様は大幅に変更された．そのため，2004年から2006年までの3年間で培われた3Dリーグのノウハウはすべて無駄になってしまっただけでなく，競技参加者は二足歩行制御のための基礎開発を強いられることになった．シミュレータそのものの完成度も高いとは言えず，さまざまな環境で安定して動作させることが難しい状況が数年にわたって続いた．厳密な物理シミュレーションを実行しているために計算機に要求する性能も大きく，11対11の対戦もまだ実現できていない．残念ながら，3Dリーグのシミュレータは研究プラットフォームとして安定しているとは言い難いのが現状である．

しかしながら，失われた3年間の反省を踏まえ，2008年以降はアルデバラン社のNao[☆2]を模したロボットモデルを採用し，2Dリーグと同様に仕様変更を可能な限り小さく抑える努力がなされている．シミュレータの仕様変更が緩やかになったこともあり，エージェントの二足歩行制御技術は徐々に進歩を遂げている．今後は，計算機の性能向上に合わせてエージェントの数を増やし，11対11のゲームを可能な限り早期に実現することが目指されている．

## 抽象シミュレーションとリアルロボットシミュレーション

3Dリーグが開始された当初は，サッカーシミュレーションリーグコミュニティ全体での2Dリーグから3Dリーグへの緩やかな移行が想定されていた．しかしながら，それぞれの方向性が完全に異なるものとなったため，2Dリーグと3Dリーグとで完全な棲み分けがなされることとなった．

2Dリーグの方針は当初から一貫しており，抽象シミュレータによるチームワーク研究が志向されている．一方，3Dリーグは物理エンジンを用いて現実のロボットモデルを可能な限り忠実にシミュレーションすることが目指されており，ロボットシミュレータとしての側面が強い．当然，マルチエージェントシステム，チームワーク研究を目的としていた者は2Dリーグを好み，実ロボットを扱う者は3Dリーグを好むことになる．結果として，サッカーシミュレーションリーグという括りではあるが，実態はまったく異なるサブリーグが共存することになった．

# ロボカップサッカーシミュレータ

2Dサッカーシミュレータ（以下，RCSS）[3]はロボカップ第1回大会よりサッカーシミュレーションリーグの公式シミュレータとして用いられてきた．RCSSは電子技術総合研究所で開発され，現在はThe RoboCup Soccer Simulator Maintenance Committeeによって，オープンソースで開発・維持されている．筆者自身もメンテナンスグループのメンバであり，RCSSの開発全般を担当している．シミュレータ一式は公式サイト[☆3]より入手可能である．

## サッカーシミュレータの仕組み

図-3にRCSSの構造を示す．RCSSはサーバ・クライアントシステムで設計されており，シミュレータ本体(RoboCup Soccer Simulator Server：rcssserver)はサーバプログラムである．試合を実行するには，11体のプレイヤエージェントとコーチエージェントの動作を制御するクライアント

☆2 http://www.aldebaran-robotics.com/
☆3 http://sserver.sourceforge.net/

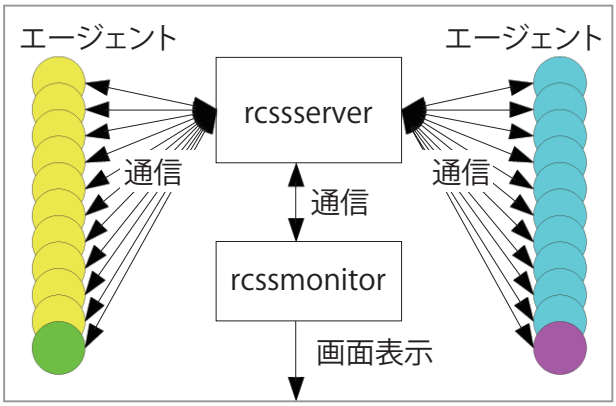


図-3　2D サッカーシミュレータの構造

プログラムを 2 チーム分用意し，RCSS へ接続する．仮想フィールドの状態は，rcssserver と専用の画面表示プログラム (RoboCup Soccer Simulator Monitor：rcssmonitor) とが通信することで視覚化される．rcssmonitor もクライアントプログラムである．rcssserver とクライアントプログラムとの通信は UDP/IP によって行われる．通信プロトコルは LISP の S 式を模したものとなっている．近年はエージェント開発用ライブラリが充実してきており，通信やプロトコルの詳細を意識せず，エージェントの意思決定に注力できるようになっている．各エージェントプログラムは完全に独立したプロセスで制御され，プロセス間の直接的な通信は競技ルールとして禁止されている．エージェント間の通信としては，RCSS を介したコマンドによる仮想的な聴覚コミュニケーションのみが許されている．これらの仕組みにより，RCSS は完全自律分散型のマルチエージェントシステムとなっている．

RCSS は現在の仮想フィールドの状態に基づいて，各エージェントへセンサ情報メッセージを送信する．エージェントは受信したセンサ情報に基づいて内部モデルを再構築し，自分自身を制御するために kick，dash，turn などの基本的な行動コマンドを RCSS へ送信する．RCSS は各エージェントから送信されてきたコマンドに基づいて物体の動きをシミュレートし，サッカーのルールに基づいて仮想フィールドの状態を更新する．RCSS 内の空間は連続で，仮想フィールドの大きさは現実のサッカーフィールドとほぼ同じ大きさに設定されている．各プレイヤの能力も現実の人間に近くなるように設定されているが，物理モデルは現実のものとは対応しておらず，あくまでそれらしく動くように調整されているに過ぎない．プレイヤエージェントが得られるセンサ情報は制限されており，環境の一部しか知覚できない．センサ情報には意図的なノイズが混入される．さらに，プレイヤの行動にもノイズが混入されるため，実行したコマンドが意図どおりの結果をもたらすとは限らない．

RCSS の時間モデルは離散時間で，1 シミュレーションステップは 100 ミリ秒である．ただし，通信は非同期に行われるため，エージェントの意思決定がシミュレーションサイクルよりも遅くなった場合，rcssserver はそのエージェントは何も実行しなかったものとみなしてシミュレーションを進めてしまう．そのため，エージェントをサッカープレイヤとして安定動作させるにはリアルタイムの意思決定が要求される．

このように，RCSS は不完全知覚問題，同時学習問題，報酬分配問題といったマルチエージェント学習における重要な問題を含んでいる．RCSS の仕様についてまとめたドキュメントは公式サイト内に Wiki ページとして整備されている．詳しくはそちらを参照してもらいたい．

## サッカーシミュレータの進化

研究要素を増やし，抽象シミュレータでありつつもより現実的なシミュレーションを実現するために，RCSS にはさまざまな変更が加えられてきた．しかし，ロボカップでは，競技として成立させるだけでなく研究の継続性と毎年の技術発展を明らかにすることも必要であるため，RCSS の仕様変更においては後方互換性を保つことが重要視されている．

RCSS が研究プラットフォームとしてすでに安定していることもあり，近年の仕様変更は慎重に進

められている．大幅な変更は数年にわたって緩やかに導入するように計画されており，仕様変更へ追従するために競技参加者にかかる負担を小さく抑えることも重視されている．

RCSS はオープンソースのプロジェクトとして管理されているが，開発時の仕様変更には制限が設けられている．メンテナンス目的の修正は随時可能であるが，競技ルールの変更を伴う修正に関してはコミュニティ内の議論に基づいて決定される．ルールおよび仕様の詳細については，ロボカップ国際委員会のサッカーシミュレーションリーグ Technical Committee の議論によって決定される．Technical Committee は競技参加者内の投票によって決定される任期 1 年の委員で，例年，ロボカップ世界大会期間中に投票が行われている．Technical Committee のメンバは，世界大会終了後に長期的なシミュレータの方向性および次年度のシミュレータに導入するルールについて議論を行い，仕様の詳細を決定する．最終的にコミュニティ全体の合意が得られれば，決定した仕様に基づいてメンテナンスグループによって実装作業が進められる．

RCSS の主要な機能拡張・ルール変更の変遷を表 -1 に示す．初期のころは，プレイヤの身体能力調整やサッカーとしての試合の体裁を整えるためのルール追加など，サッカーシミュレーションとしての妥当性を高めるための変更が多い．一方で，1999 年にコーチエージェントをクライアントとして使用可能となり，2001 年にはコーチからプレイヤへ向けたアドバイスを発するための共通言語として標準コーチ言語が導入されている．2002 年からはプレイヤ間の言語コミュニケーションの制限が大きくなり，代わりに指さしという視覚的なコミュニケーション手段が導入された．2003 年にはペナルティキックと Keepaway モードという特殊な試合実行環境が導入されている．これらの機能拡張の変遷から，サッカーシミュレータとしての設計は 2000 年ころまでにほぼ完了し，それ以降はチームワーク研究の新たな課題を作り出すための変更が多くなっていることが分かる．

| | |
|---|---|
| 1998 年 | ゴールキーパ<br>オフサイドルール<br>長期的なスタミナモデル |
| 1999 年 | turn_neck（首振り）コマンド<br>オンラインコーチ |
| 2000 年 | パラメータ調整 |
| 2001 年 | キック能力の増加<br>ヘテロジーニアスプレイヤプレイヤ交代<br>標準コーチ言語 |
| 2002 年 | コミュニケーションメッセージ短縮<br>attentionto（聴覚の注意向け）コマンド<br>pointto（指さし）コマンド<br>tackle コマンド<br>バックパスの禁止<br>セットプレイのファウル |
| 2003 年 | ペナルティキック<br>ゴールポスト<br>Keepaway モード |
| 2008 年 | 視覚情報の同期化<br>衝突検出メッセージ<br>プレイヤの走るスピードを抑制<br>ゴールキーパのキャッチ能力低下<br>ボールの最大スピード増加<br>キックノイズモデル<br>タックルモデル変更<br>ヘテロジーニアスプレイヤの使用強制<br>ヘテロジーニアスプレイヤタイプ数の増加 |
| 2009 年 | プレイヤの走るスピードをさらに抑制<br>スタミナキャパシティモデル<br>体の横方向へのダッシュ |
| 2010 年 | 意図的ファウル<br>レッドカード，イエローカード<br>スタミナキャパシティ調整<br>体の斜め方向へのダッシュ<br>ヘテロジーニアスゴールキーパ |
| 2011 年（予定） | 全方位移動<br>TCP/IP 通信<br>シミュレーションの完全再現性 |

表 -1　主なルール変遷

2002 年以降，サッカーシミュレーションリーグ全体での 2D から 3D への移行が計画されたため，RCSS の仕様を凍結することが決定された．2003 年から 2007 年までは，RCSS への変更はバグ修正やルールにかかわらない設計の改良などに限定された．この期間中，3D リーグが実際に立ち上げられたものの，3D リーグへの移行はなかなか進まなかった．さらに，2007 年以降，3D リーグはヒューマ

ノイド型エージェントを用いたロボットシミュレータの方向へ進むことが確定したため，チームワーク研究のプラットフォームとして 2D リーグが再評価されることとなった．その結果，2008 年から再び RCSS の仕様変更が再開された．

2008 年以降の主要な変更方針は，シミュレータとしての設計の見直し，プレイヤの行動モデルの再調整，より長期的な戦略・戦術のプランニングの必要性を要求する機能追加，である．

RCSS は抽象シミュレータとして設計されているが，2008 年以前の RCSS では通信の非同期性に対するロバスト性も強く要求されていた．それまでの RCSS では，プレイヤが得られる視覚情報はシミュレーションステップとは非同期に送信されており，プレイヤエージェントの意思決定のタイミング管理にかかる開発コストが大きな負担となっていた．このような人工的な非同期性に疑問の声が多く上がり，新しく同期モードが導入されることになった．また，RCSS は完全な分散シミュレータとして設計されているために，シミュレーションの再現性に乏しいという問題を抱えている．実験環境として考えた場合，シミュレーションの再現性は重要な要素である．現在の RCSS の通信は UDP/IP で行われているため，通信の安定性の面でも問題がある．これらの問題を解決するために，完全な再現性を実現するためのシミュレーションタイマーモデルの導入と，UDP/IP から TCP/IP への移行が予定されている．

プレイヤの行動モデルの再調整によって，特にプレイヤの移動能力にかかわるパラメータ調整や機能追加が行われつつある．プレイヤの走るスピードを遅くすることで，プレイヤの身体能力に頼った戦術を抑制し，効率良く協調行動を行う必要性が増した．一方で，従来はプレイヤの体が向いている方向にしか走れなかったモデルを拡張し，全方位移動を想定したモデルが導入されている．この変更は，抽象シミュレータでありつつも可能な限りリアリティを持たせることが意図されている．

スタミナキャパシティモデルは，より長期的な戦略・戦術のプランニングを要求するために導入された．従来の RCSS のスタミナモデルでは，プレイヤはフィールドの端から端まで全力で走り続けることができなかった．そのため，チームのフォーメーションを大きく崩して柔軟にポジショニングすることがほぼ不可能であり，多様な戦術を展開することが困難であった．この問題に対応するために，プレイヤのスタミナパラメータの上限値を大きくし，全力で走り続けられる距離を大幅に延ばす変更がなされた．その一方で，一試合中に消費できる総スタミナ量（スタミナキャパシティ）を導入するというトレードオフが導入された．これらの変更によって，短期的にはより複雑な戦術を展開できるようになったものの，戦局を考慮した長期的なスタミナ管理が要求されることになった．

ファウルモデルは，短期的な状況改善と大局的な戦略を考慮した意思決定を要求するために導入された．ファウルモデルは従来のタックルモデルを拡張したものである．ファウルは通常のタックルよりもボールへの干渉が成功しやすいが，その代償として，審判によってファウルを検出された場合には相手チームへのセットプレイが与えられるようになった．さらに，特定の条件下でのファウルが検出されれば，ファウルを実行したプレイヤに対してイエローカードが発行されるようになった．実際のサッカーのルールと同様に，イエローカードを 2 枚発行されるとレッドカードとなり，そのプレイヤは退場させられる．退場となったプレイヤはフィールド外に出され，それ以降，試合にかかわることはできなくなる．ファウルモデルの導入によって，退場の危険を冒してもファウルを実行すべきか否かといった判断が要求されるようになった．

## 今後のルール改正

今後の変更として，完全な全方位移動の導入，後方ダッシュの廃止，が予定されている．これらの変更についてはすでに移行期間に入っており，数年をかけて緩やかに調整が進められることが予定されて

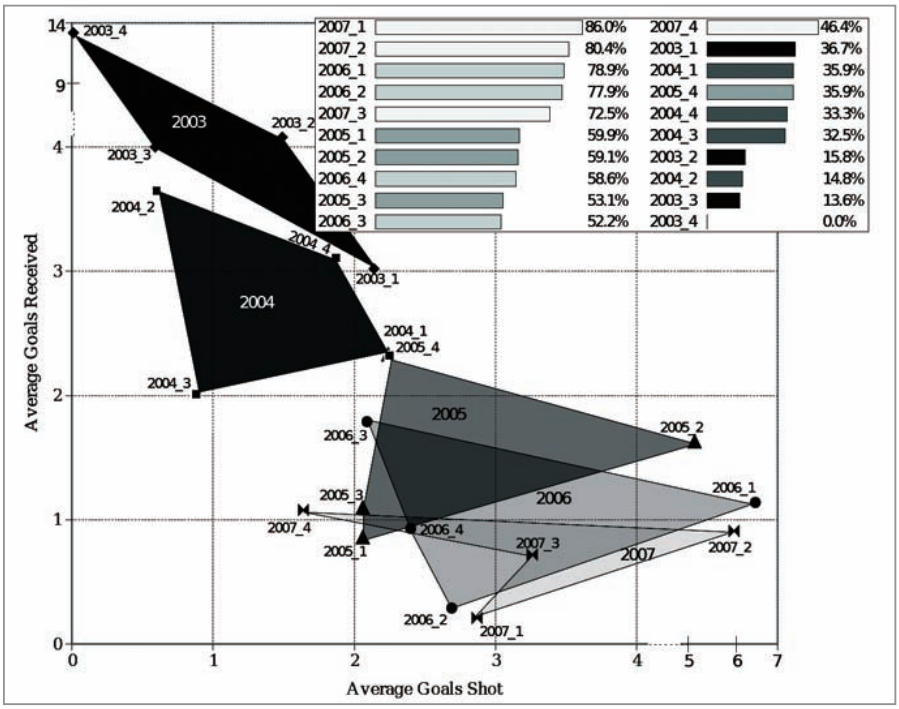


図 -4　2003 年から 2007 年までの上位 4 チームによる対戦結果(文献 2)より

いる．その他，RCSS のキックモデルが現実の人間のプレイと比較して正確過ぎるという点が指摘されており，新しいキックモデルが導入される可能性がある．

今後もルール改正は続くと考えられるが，RCSS が研究プラットフォームとしてすでに高い完成度を持っていることから，大きな仕様変更が決定されたとしても，移行期間を設けて緩やかに移行していくだろう．

## 競技レベルの進歩と 2D リーグの方向性

ロボカップの競技会が開始されて以降，サッカーシミュレーションリーグの競技レベルは年々向上している．数年前の世界大会優勝チームに対して最近のチームが 10 点以上の大差で勝利することも珍しくなく，サッカーチームとしての強さの進歩は驚くほどの早さで進んでいる．

Gabel らは，リーグ全体のパフォーマンスの改善過程を定量的に評価することを試みている[2)]．彼らは，2003 年から 2007 年まで RCSS の仕様が変更されなかったことに注目し，その間の参加チームを用いたパフォーマンス比較分析を行った．図 -4 にその一例を示す．図中の値は，2003 年から 2007 年までの世界大会上位 4 チームによる総当たり戦を 270 回以上繰り返した平均値が示されている．図中の右上の棒グラフは，20 チーム間の相対的な強さを表している．メインのグラフは，横軸が各チームの平均得点，縦軸が平均失点を示している．図からも明らかなように，より新しいチームほど得点が増し，失点は減っている．各年の優勝チームは前年の優勝チームよりも，得点の多さまたは失点の少なさのいずれかで必ず勝っていることが分かる．また，パフォーマンス改善を説明する明確な傾向が観察されないことにも触れ，2D リーグにおけるサッカーチームとしてのパフォーマンスが依然として進歩し続けていると結論づけている．

さらに，Gabel らは 2009 年の世界大会において起こった，競技期間中のマイナーチェンジによる

劇的なパフォーマンス改善についても分析を行っている．Gabel らは Brainstormers というチームでロボカップの競技に参加している．Brainstormers は 2007 年と 2008 年を連覇した強豪である．チームの特徴は，コーチエージェントによるオンライン役割配分を活用した徹底したマンマーク戦術である．これに対して，WrightEagle というチームはマンマークを実行している Brainstormers のエージェントを混乱させることを狙い，試合途中でプレイヤの役割を入れ替える戦略を実践した．この戦略はきわめて有効に機能し，Brainstormers の守備は完全に崩壊させられた．さらに，この試合を観戦していた Oxsy というチームが同様の戦略を一晩で実装し，翌日の対 Brainstormers 戦で使用した．そして，Brainstormers は Oxsy に破れる結果となった．競技期間中，Brainstormers も役割交換戦略に対するカウンター戦略を実装したが，残念ながらそれが有効に利用される機会は訪れなかった．競技会終了後，Gabel らは，役割交換戦略が実行された場合，役割交換戦略に対するカウンター戦略を実行した場合の比較実験を行っている．その結果，特定の戦略に対するカウンター戦略の有効性が実験的に示され，戦略修正によって大幅なパフォーマンス改善が起こることが明らかにされている．

競技期間中の観察に基づいて特定の戦略に対するカウンター戦略を適用する，という取り組みは，2D リーグの今後の方向性の 1 つを示しているだろう．現在は人間による観察と実装が必要であるが，戦略・戦術のバリエーションが豊富になるにつれて，対戦相手に対してどの戦略・戦術を適用するかという判断を自動化する取り組みが可能となる．このような取り組みは，コーチエージェントによる試合分析とプレイヤに対するアドバイス，という形で実現されるだろう．近い将来，試合ログデータを競技期間中にコーチエージェントが分析し，その後の試合でカウンター戦略・戦術を適用するといった試みが始まることが予想される．

## 研究プラットフォームとしてのサッカーシミュレーションリーグ

### 個体制御の獲得

RCSS 上で動作するプレイヤエージェントの開発において，サッカープレイヤとしての個人能力の精度向上は初期のころから多く取り上げられた研究テーマであった．サッカープレイヤとしてのスキル獲得に機械学習を適用する試みは多く，特に強化学習の適用による成功事例が多いようである．近年の計算機能力の向上に伴い，オンライン探索によって行動のプランニングを行う事例も見られる．実際の試合では，他のプレイヤの存在を考慮して妨害を受けないように行動を遂行するためのプランニングが必要である．そのため，プレイヤの意思決定時には，現在の状況に応じて可能な行動の集合をオンライン探索で生成し，オフライン学習で獲得した評価関数に基づいてそれらを評価する，という方法がしばしば採用されている．

### モデリング

2D リーグにおけるサッカープレイヤとしての個体の能力はきわめて高いレベルに達しているが，改善の余地はまだ多く残されている．特に，他のプレイヤのモデリングに関しては十分に研究が進んでいるとは言えない．たとえば，1 対 1 の守備能力を強化学習によって獲得するという研究事例[1)]が成功を収めているが，対戦相手の攻撃能力に合わせて守備の強さを変化させるといったオンライン適応への取り組みについてはほとんど手つかずのまま残されている．守備の例で言えば，もしオンライン適応が成功すれば，エージェントの制御にかかるスタミナコストを抑え，余ったリソースを他の戦術のために割り当てるといったことも可能になるだろう．敵プレイヤの行動モデルを推定できれば，パスコースの成功確率推定などへの応用が可能となり，意思決定における評価関数の動的な修正への発展も期待できる．他のエージェントのモデリングとエージェントの意

思決定を結びつけてオンライン適応での成功を収めた研究事例はまだ存在せず，今後の展開が期待される分野である．

チームの戦略・戦術の分析についても発展途上の分野である．サッカーシミュレーションリーグには2006年までコーチリーグというサブリーグが存在していた．コーチリーグは，コーチエージェントのアドバイスによるオンライン適応が競われる競技として立ち上げられたサブリーグである．しかし，十分な研究成果を出せず，ルール策定が迷走した結果，コーチリーグ自体の存在意義を問われて消滅する結果となってしまった．しかし，前節で述べたように，特定の戦術に対するカウンター戦術で成功を収めるといった事例はようやく観察され始めたばかりである．残念ながらコーチリーグはなくなってしまったが，対戦相手の分析とそれに対する戦略や戦術の選択の自動化といったチーム単位での適応は，今後ますます重要になっていくだろう．

## 探索型エージェント

RCSSは非同期シミュレータであり，エージェントはリアルタイムな意思決定や通信に対するロバスト性を要求される．ロボカップが始まった当初は計算機環境の貧弱さもあり，エージェントの意思決定に割り当てられる計算機リソースは非常に限られたものであった．しかし，計算機能力の飛躍的な向上に伴い，エージェントの意思決定において，以前とは比べ物にならないほどの精緻で深い計算を実行できるようになった．その結果，探索型エージェントによるパフォーマンス改善の可能性が広がりつつある．ここでの探索型エージェントとは，行動の連鎖と状態予測をリアルタイムに生成・評価するエージェントを意味する．従来は人手で作り込んだルールベースエージェントが主流であったが，作り込んだエージェントは柔軟性に乏しい．これに対して，探索型エージェントは評価関数次第でさまざまな戦術を容易に実現できるだけでなく，状態表現の設計，探索空間の設計，評価関数の獲得など研究的にもさまざまな展開が期待できる．問題設定の整理がまだそれほど進んでおらず，将棋などの完全情報ゲームにおける方法論が活かせる基盤が整ったとは言えない状況であるが，今後の発展に期待したい．

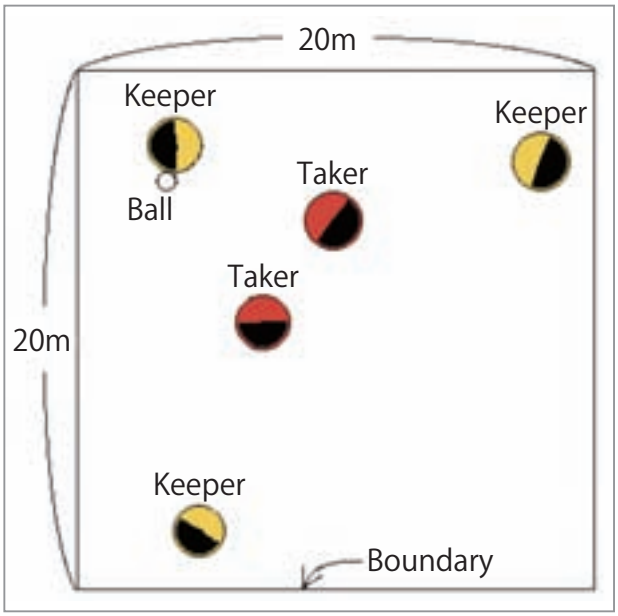

図-5　Keepaway モード

## テストベッドのためのサブタスク

RCSSには，Keepawayというサブタスクを実行するモードが用意されている．Keepawayモードでは，プレイヤはkeeperと呼ばれるチームとtakerと呼ばれるチームに別れ，keeperがある領域内でボールを保持し続けることを，takerはそのボールを奪うことを目的とする．Keepawayは通常の試合よりも少人数のプレイヤで実行することが想定されており，初期設定では20m × 20mの矩形領域内で3体のkeeperと2体のtakerがボールを奪いあう設定となっている（図-5）．Keepawayは通常のサッカーの試合では複雑すぎる環境を適度に簡単化しており，強化学習のテストベッドとして利用されている．強化学習以外にも，さまざまな手法の評価を行うための実験環境として利用価値は高いだろう．

## チーム開発

RCSS上で動作するエージェントの開発を一から始めるのは簡単ではない．ルールの複雑化や競技レベルの進歩に伴い，実装しなければならない項目は多岐にわたっており，最低限の動作をするエージェントを実装するだけでもソースコードが1万行以上に達するのが普通である．近年の上位チームのソースコードはC++言語で数十万行に達しており，競技会でパフォーマンスを発揮できるチームを作るためにはソフトウェア開発におけるプロジェクト管理能力も要求されている．このような状況で一から実装作業を始めていては，研究を始めるまでに数年を要してしまうだろう．これからサッカーシミュレーションリーグへ挑戦するのであれば，既存のベースチームを利用することを薦めたい．

本稿では，筆者が開発している以下のソフトウェアについて紹介する．

- librcsc：サッカーエージェント開発のための基本ライブラリ
- agent2d：librcscを用いたサンプルチームプログラム
- fedit2：agent2dのためのフォーメーションエディタ
- soccerwindow2：エージェント開発補助のためのビューワプログラム

これらのソフトウェアはLGPLまたはGPLに基づいて公開されており，自由に利用可能である☆4．2006年にライブラリの解説書が出版されたが[6]，2010年現在，絶版となっている．書籍とほぼ同内容のドラフト版PDFを同様に公開しているので，必要に応じてそちらを参照してもらいたい．

☆4 http://sourceforge.jp/projects/rctools/

### サッカーエージェント開発用ライブラリ：librcsc

librcscはRCSS上で動作するサッカーエージェントおよび関連ツールを開発するための基本ライブラリで，C++で実装されている．librcsc自体はライブラリ群であり，以下で紹介するagent2d，fedit2，soccerwindow2で使用される．librcscには，2次元の幾何演算，RCSSとの通信と同期，エージェントの内部モデル，サッカープレイヤとしての基本的な行動，ログ解析などのためのさまざまなクラスライブラリが含まれている． librcscは5年以上にわたって継続してメンテナンスされているため，安定性が高く，最近の開発環境でも問題なく動作する．Linux，FreeBSD，Max OS Xを公式にサポートし，Windows環境での動作報告もある．また，RoboCup 2010において優勝したHELIOS2010でもlibrcscを使用しており，性能面でも世界トップレベルのものが提供されている．

librcscのソースコードは16万行程度の規模となっており，個人で利用するライブラリとしてはやや大きいかもしれないが，チーム開発においてはライブラリのすべてを熟知する必要はない．幾何演算クラス，エージェントの内部モデルの参照方法，そして，サッカープレイヤとしての基本的な行動群についておおよその把握ができれば，独自のチームを作る準備としては十分である．APIリファレンスとしては，Doxygen形式のドキュメントがほぼ完全に整備されている．

librcscを利用するには，システム上にヘッダファイルとライブラリファイルのインストールが必要となる．ソースコードはGNU Autotoolsでパッケージングされており，インストール手順は以下のとおりである．configureに‘-- help’オプションをつけて実行すると，利用可能なオプションが表示される．たとえば，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インストール先を指定することができる．

```
$ tar xzvf librcsc-x.x.x.tar.gz
$ cd librcsc-x.x.x
$ ./configure
$ make && sudo make install
```

## ベースチーム：agent2d

agent2d は，librcsc を用いたベースチームである．初期状態でサッカーチームとしてある程度の性能を発揮する実装が施されており，チーム開発時のテンプレートとしての利用が想定されている．agent2d をテンプレートとすることで，rcssserver との通信と同期，センサ情報の解析や行動コマンドの生成といった低レベルな処理を意識せずに，エージェントの意思決定のみに注力することができる．2010 年現在，日本国内の 2D リーグ参加チームのほぼすべてが agent2d を用いており，海外からの競技参加者にも利用者は多い．

agent2d のプレイヤエージェントは，ゴールキーパに関しては専用の行動アルゴリズムが実装されているが，フィールドプレイヤは全員が同じアルゴリズムで動作する．各フィールドプレイヤは背番号に基づいてチームフォーメーション内の配置が割り当てられ，ボールを持っていない状況では組込みのフォーメーションシステムに基づいたポジショニング動作を実行する．自分がボールに最も早く到達できると判断した場合は，自動的にボールを追いかけ，ボールを持った場合は，パスやドリブルなどを簡易な評価関数に基づいて選択する．戦術的な作り込みはまったく施されていないので，チームとしては単純な動作しかできない．独自のチームを作る場合は，特にボールを持ったときの意思決定アルゴリズムに工夫を施すことが必要になる．また，ポジショニング動作についても，マンマークなどの状況に応じた特殊な行動は実装されていないので，各自で実装する必要がある．役割配分や配置の調整といったフォーメーションの最適化も必要である．プレイヤ間のコミュニケーションとして，ボール情報などを状況に応じて共有するルールが組み込まれているが，それら以外の情報共有が必要であれば独自の工夫を施さなければならない．

agent2d のコーチエージェントは，試合開始前に組込みのルールに基づいてヘテロジーニアスプレイヤの割り当てを行い，試合実行中は敵チームのヘテロジーニアスプレイヤの分析を自動的に実行する．ヘテロジーニアスプレイヤの割り当て方によってチームのパフォーマンスが大きく変動するため，役割配分タスクとして工夫を施す必要がある．librcsc ではコーチ言語をサポートしていないため，オンラインでのアドバイスはまだ実行できない．

agent2d を利用するには，システムに librcsc がインストールされていなければならない．ビルド手順は以下のとおりである．

```
$ tar xzvf agent2d-x.x.x.tar.gz
$ cd agent2d-x.x.x
$ ./configure
$ make
```

ビルドが成功すれば，パッケージ内に含まれるチーム起動スクリプトによってチームを実行することができる．チームの実行方法は，rcssserver と rcssmonitor を起動して，src ディレクトリ内の‘start.sh’を実行するだけである．成功すれば画面上にプレイヤが現れ，フィールド内に配置される．

## フォーメーションエディタ：fedit2

agent2d にはフォーメーションシステムがあらかじめ用意されている．各プレイヤエージェントはこのフォーメーションシステムに基づいてポジショニング動作を行うため，フォーメーション内の役割配分とプレイヤの配置によってチームのパフォーマンスが大きく変動する．チーム戦略の土台となるため，チーム開発においてフォーメーションの最適化は重要である．

実装されているフォーメーションシステムのアイディアは，ボールの位置座標に基づいて各プ

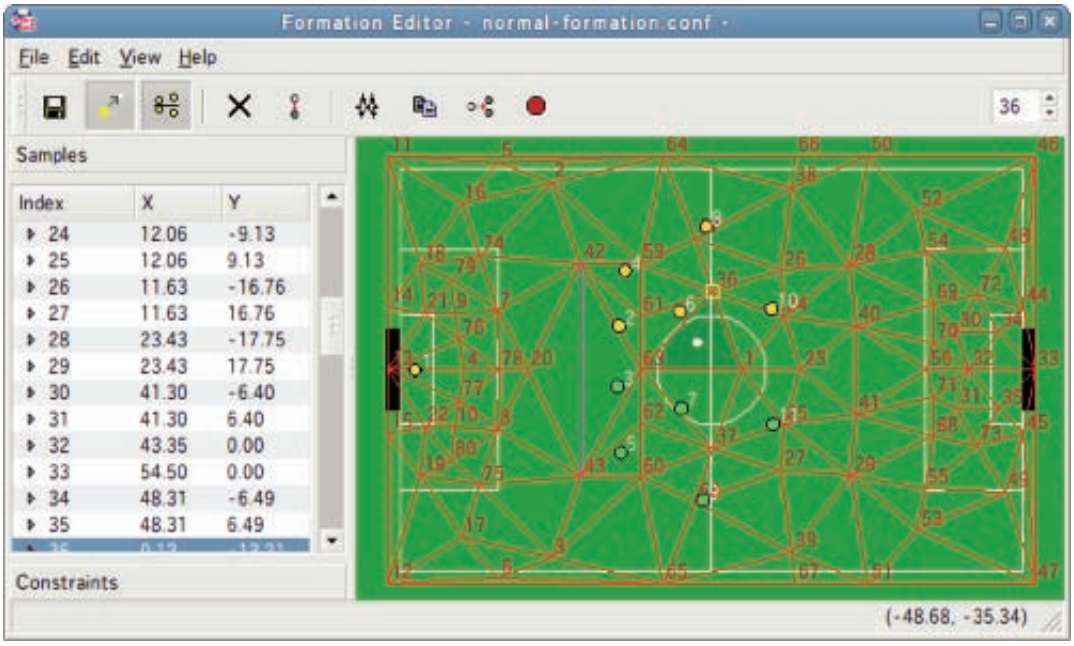


図-6　フォーメーションエディタの実行画面

レイヤの目標移動位置座標を決定する，というものである[5]．原理は単純だが，サッカーにおいてはボール位置が状況分類の重要な指標となるため，ボール位置に基づいて配置を決定する考え方は効果的である．

従来は人手によるパラメータチューニングでフォーメーションを修正することが多かったが，このパラメータチューニングを直感的に行える仕組みが利用可能である．図-6は，筆者らが開発してるフォーメーションエディタ，fedit2の実行画面である．チーム開発時，fedit2を用いてプレイヤの役割配分と状況に応じた配置をGUI上でデザインすることができる．

組込みのフォーメーションシステムでは，フォーメーションをボール位置座標からプレイヤの移動目標位置座標へ写像する関数とみなし，Delaunay三角形分割を利用した関数表現モデルを採用することで直感的かつ柔軟な仕組みを実現している．ユーザがフォーメーションをデザインする場合，fedit2上でボールを移動させ，その位置座標に応じてプレイヤを配置し，新しいサンプルとして登録する，または，不要なサンプルを削除する，という作業を繰り返すことになる．登録されたサンプルのボール位置座標に基づいてフィールド上に三角形分割が生成される．入力ボール位置が三角形分割の頂点以外の場合，既存の頂点が持つ情報から補間を行うことでプレイヤの移動目標位置が出力される．

fedit2を利用するには，システムにlibrcscとQt4がインストールされていなければならない．ビルド手順は以下のとおりである．

```
$ tar xzvf fedit2-x.x.x.tar.gz
$ cd fedit2-x.x.x
$ ./configure
$ make && sudo make install
```

ビルドおよびインストールが成功すれば，‘fedit2’コマンドによってエディタを起動できる．agent2dに含まれるフォーメーション設定ファイル（.confファイル）をエディタで開けば，既存のフォーメーションを修正することができる．新規にフォーメーションを作成することももちろん可能である．

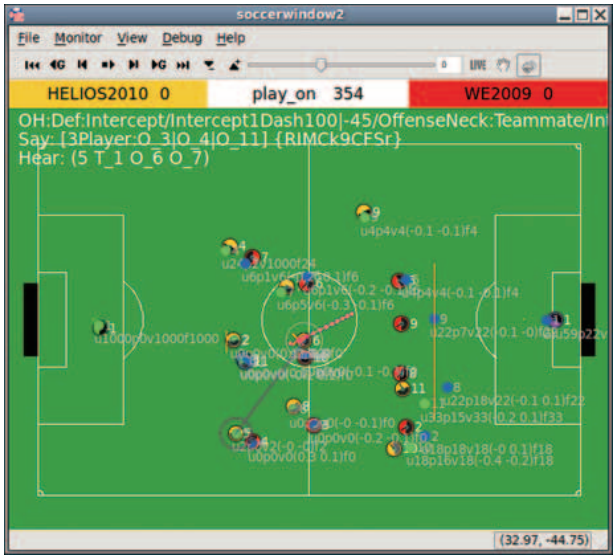


図 -7　エージェントの内部状態表示の様子

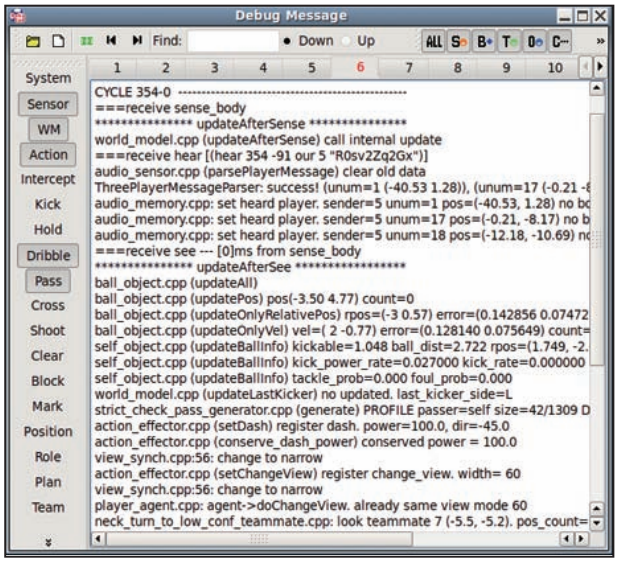


図 -8　エージェントのデバッグメッセージ表示の様子

## 多機能ビューワ：soccerwindow2

soccerwindow2 はプレイヤエージェントの開発補助を目的とした多機能ビューワプログラムである．rcssmonitor 互換のモニタクライアントプログラムとして動作するだけでなく，単体でログプレイヤとして動作可能である．タイムシフト再生機能も実装されており，試合実行中にも巻き戻して試合を再生することができる．さらに，プレイヤエージェントが出力するデバッグ情報を視覚的に表示するビジュアルデバッガとしても設計されている．図 -7 と図 -8 は，soccerwindows をビジュアルデバッガとして使用した際の実行画面のスナップショットである．

図 -7 では，rcssserver 上のフィールド状態に加えて，プレイヤエージェントの内部状態を重ねて表示している．内部状態を画面表示することで，エージェントの信念と実際の状況とのずれを確認しやすくなり，各エージェントの意思決定内容の確認も容易になる．この機能はデバッグサーバとして実装されているため，内部状態の表示は試合実行中にオンラインで実行可能であり，開発効率を著しく促進する手助けとなる．プレイヤエージェント側でこの機能を利用するには，エージェント起動時にデバッグクライアントモードでの起動を示すオプションを与える．デバッグクライアントモードで起動されたエージェントは soccerwindow2 へ自動的に内部状態の送信を行う．対応するオプションは agent2d に含まれる起動スクリプトに用意されている．

図 -8 は soccerwindow2 が持つ機能の 1 つで，テキストによるさらに詳細なデバッグメッセージ表示機能の様子である．エージェントが出力するデバッグメッセージには任意の文字列を使用でき，シミュレーションサイクルごとの表示切り替え，ログレベルによる表示内容の切り替えをサポートしている．ほぼ無制限にデバッグ情報を表示できるため，デバッグすべき項目が大量の情報を出力する場合に有効である．デバッグメッセージとして任意のフォーマットのテキスト情報を出力できる．ただし，メッセージ解釈は行指向で行われるため，複数行にわたるデバッグ情報を柔軟に扱うことはできない．エージェントをデバッグするためのプロトコルの規格化は今後の課題である．

soccerwindow2 を利用するには，システムに librcsc と Qt4 がインストールされていなければならない．ビルド手順は以下のとおりである．

```
$ tar xzvf soccerwindow2-x.x.x.tar.gz
```

```
$ cd soccerwindow2-x.x.x
$ ./configure
$ make && sudo make install
```

ビルドおよびインストールが成功すれば，‘soccerwindow2’コマンドによってビューワを起動できる．モニタクライアントとしてのrcssserverへの接続，ログの再生，さまざまな表示設定の変更はGUIのメニュー上で実行できる．

soccerwindow2はエージェント開発の大きな助けとなる．このようなビジュアルデバッガを利用しなければ，開発者の意図どおりのエージェントを開発することはきわめて難しい．サッカーシミュレーションに取り組む場合はぜひ利用してもらいたい．

## サッカーシミュレーションリーグの今後の展望

2Dリーグはいまだ発展途上であり，研究のためのテストベッドとして利用できる余地も多く残されている．RCSSはマルチエージェントシステム，チームワーク研究のプラットフォームとして今後も利用されていくだろう．3Dリーグはリアルロボットシミュレータとしての側面が強いが，3Dサッカーシミュレータの完成度が上がり，エージェントの行動制御ライブラリが充実して意思決定の抽象化が進めば，2Dリーグと3Dリーグの融合が進むかもしれない．将来的には，「自律二足歩行ロボットのチームで人間チームに勝利する」というロボカップの最終目標に向けて，2Dリーグ，3Dリーグ，実機リーグの枠を越えて，チームワーク研究の方法論を再利用するためのフレームワークを構築していくことが必要である．

**参考文献**


1) Gabel, T., Riedmiller, M. and Trost, F. : A Case Study on Improving Defense Behavior in Soccer Simulation 2d : The Neurohassle Approach, In Iocchi, L., Matsubara, H. and Zhou, C., Weitzen-feld, A. editors, RoboCup 2008 : Robot Soccer World Cup XII (2008).
2) Gabel, T. and Riedmillar, M. : On Progress in Robocup : The Simulation League Show-case (2010).
3) Noda, I. and Matsubara, H. : Soccer Server and Researches on Multi-agent Systems, In Kitano, H. editor, Proceedings of IROS-96 Workshop on RoboCup, pp.1-7 (1996).
4) Obst, O. and Rollmann, M. : SPARK - A Generic Simulator for Physical Multiagent Simulations, Computer Systems Science and Engineering, Vol.20, No.5, pp.347-356 (Sep. 2005).
5) Reis, L. P., Lau, N. and Oliv'eira, E. : Situation Based Strategic Positioning for Coordinating a Simulated Robosoccer Team, Balancing Reactivity and Social Deliberation in MAS, pp.175-197 (2001).
6) 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，秀和システム(2006)．


(平成22年8月23日受付)

■秋山英久（正会員）hidehisa.akiyama@aist.go.jp
2008年東京工業大学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．(独)産業技術総合研究所情報技術研究部門特別研究員．博士(工学)．マルチエージェントシステム，災害情報システムの研究に従事．