2007－GI－17 (2)
2007／3／5

RoboCupサッカーシミュレーションにおけるエージェント配置手法の提案

秋山 英久 †, 野田 五十樹 †

† 産業技術総合研究所

マルチエージェントシミュレーションにおけるさまざまな問題では，各エージェントの空間的な配置がシミュレーション結果に重要な影響を及ぼすことがある．特に，サッカーのように動的な環境では，環境の状態に応じて，プレイヤであるエージェントが移動する位置を変化させなければならないが，あらゆる状況に対して望ましい出力を事前に与えることは困難であるため，状態から移動位置座標への何らかの写像関数が必要となる．しかしながら，この目的に対して，既存の多くの関数近似モデルでは充分な精度が得られないことがある．そこで，本稿では，Delaunay 三角形分割によって空間を分割し，そこに線形補間を組み合わせることで局所的な近似を可能にするモデルを提案する．評価実験として、RoboCupサッカーシミュレータを用いたシミュレーション実験を行う．

Agent Positioning Mechanism in the RoboCup Soccer Simulation

Hidehisa AKIYAMA † Itsuki NODA †

† National Institute of Advanced Industrial Science and Technology

In this research, we propose a novel agent positioning mechanism for the dynamic environment. In many problems of the multi-agent domain, the position of each agent has significant effect to the result. If the environment is dynamic, the output position where agents should be moved must be changed according to the environment status. But, it is difficult to give the desired output to any state. So, it is necessary to acquire some map functions from the state to agents' positions. To acquire such map functions, it will be effective to adopt some function approximation methods and to use sample data input by expert as training data. However, such a function approximation method requires much computation resource and enough accuracy might not be acquired. To solve these problems, we propose a method that uses Delaunay Triangulation to split a environment space with sample data and output value is interpolated based on the shading algorithm in 3D computer graphic domain. This method is very simple, but runs fast and has high accuracy and scalability. In the experiment, we compared our method with other function approximation method. Normalized Gaussian Network is used as the compared method. To examine the performance of each method, we used the RoboCup.Soccer Simulator as an experiment environment.

1 はじめに

マルチエージェントシミュレーションにおけるさまざまな問題では，各エージェントの空間的な配置がシミュレーション結果に重要な影響を及ぼすことがある．例えば，predator-pray 問題のように，対象物体の移動に応じてエージェントの移動位置を変化させなければならないタスクでは，入力となる環境の状態と，出力となるエージェントの移動位置とをマッピングしなければならない．同様に，サッカーのように動的な環境でチーム対戦するゲームにおいては，プレイヤとなる各エージェントが現在の状況に応じて個々の判断で移動し続けなければ，チームのパフォーマンスは著しく低下してしまう．

このような問題に対して，エージェントが移動すべき位置を素早く獲得させるためには，人間の観察者からの教示情報を用いる教師あり学習が効果的である．しかしながら，任意の状態に対してエージェントが移動すべき位置を事前に与えておくことは困難であるため，与えられた有限のサンプルから出力値を補間する何らかの内挿関数が必

要となる．そこで，本稿では，環境の状態を入力とし，エージェントの移動位置座標を出力とする内挿関数の獲得に効果的なモデルとして，Delaunay 三角形分割と線形補間を組み合わせた手法を提案する．実験では，RoboCup サッカー 2D シミュレータ [4) 3)] を用いて，性能を評価する．

2 従来研究

2.1 美術館問題

計算幾何の領域では，監視システムとして戦略的にカメラを配置する美術館問題 (Art Gallery Problem)[2)] と呼ばれる問題が知られている．美術館問題の目的は，与えられた美術館の監視に必要な最少のカメラ台数とその配置を求めることであり，これは，三角形分割された多角形の 3 彩色問題として解くことができる．しかし，現実にはカメラが不足することがあるため，自律的に移動する監視員が必要となる．本稿で扱うのは，このようにリソースが不足している状況を想定した配置問題であり，更に，監視対象物体が空間内を移動する場合についても考慮する．

2.2 RoboCup サッカーシミュレーションにおけるエージェントの配置

サッカーのようなボールゲームにおいては，ボールが最も重要な注目対象であり，その位置を重要な状態変数とする考えは自然である．よって，ボールを美術館問題における監視対象と考えることで，美術館問題と同様の問題として捉えることができる．RoboCup サッカーシミュレーションにおいては，プレイヤである各エージェントの配置をボールの位置に応じて決定する手法として Situation Based Strategic Position[1)] が良く知られている．

2.2.1 Situation Based Strategic Position

Situation Based Strategic Position(SBSP) では，ボールの位置座標を入力とし，プレイヤの移動位置座標を出力する関数を用意する．更に，移動可能領域の制約条件を組み合わせることで，各々のプレイヤの最終的な移動位置座標を決定する．SBSP の基本的な移動位置決定アルゴリズムは以下のようになる．

```
GetSBSPPosition( Num )
 入力: プレイヤの役割番号 Number
 出力: プレイヤの移動位置
1. 基準位置 := BasePosition( Num );
2. 引力係数 := BallAttract( Num );
3. 移動位置 := 基準位置
     + ボール移動量ベクトル * 引力係数
4. 移動可能領域 := MovableRegion( Num );
5. if 移動位置が移動可能領域の範囲外
6.    移動位置を移動可能領域内に調整
7. return 移動位置
```

1 行目の基準位置とは，ボールがフィールド中央にある場合のプレイヤの移動位置を表す．2 行目の引力係数とは，ボールの移動に追従する割合を表す係数で，通常，$[0, 1]$ の実数である．そして，3 行目の計算によって，プレイヤの移動位置がほぼ決定する．ここでの計算は，基準位置に対して，ボールの移動量に引力係数を掛けたベクトルを足し合わせるという，単純な線形関数となっている．よって，引力係数の値が 1 に近付くほど，ボールの移動に対してプレイヤが追従することになる．最後に，プレイヤの移動可能領域を調べ，必要なら 3 行目で得られた移動位置を調整する

SBSP はかなり単純なモデルであり，他のプレイヤの存在を考慮していないが，全プレイヤがボールに注目しているという前提であれば，見かけ上チームを上手く機能させることができる．実装も極めて容易であることから，RoboCup サッカーシミュレーションリーグに参加するほぼ全てのチームが SBSP と同一か類似のモデルを採用している．実際の使用に際しては，パラメータを外部の設定ファイルに記述しておき，試合開始前に人手でパラメータ調整できるように実装される場合が多い．

2.2.2 SBSP の問題

SBSP には，移動位置決定アルゴリズムで使用される関数に出力値の特性が大きく依存してしまうという大きな問題がある．特に，前節に示しているアルゴリズムでは単純な線形関数が使用されており，単一のパラメータセットで実現できる移動特性も非常に単純なものになってしまう．そのため，移動特性を大幅に変更するためには，異なるパラメータセットを用意しておき，状況に応じて切替えて使用しなければならない．パラメータセット間の整合性の調整や，パラメータセットを切替える条件を管理しなければならなくなるなどの新たな問題が生じることとなり，利便性は低い．

2.2.3 既存の関数近似モデルの利用

我々の以前の研究において，SBSP において使用される関数を，線形関数から非線型関数へ変更し，既存の関数近似モデルによって獲得させることを試みた[7]．関数近似モデルとして，シグモイド関数を発火関数として持つ3層パーセプトロンを使用し，GUI のツールによって人間がサンプルを与える教師あり学習によって近似関数を獲得させた．結果として，ある程度実用に耐える近似関数を得られることが確認できた．しかしながら，サンプルの再現精度はあまり高くなく，教示者の思いどおりの配置を実現することは難しかった．また，3層パーセプトロンの過学習が頻繁に発生するという問題が生じた．ひとつのサンプルによる学習が全体に影響を及ぼしてしまうために，サンプルを与え続けるることで、初期に与えたサンプルの学習結果が損なわれてしまうという現象がしばしば発生した．この結果から，我々は，サッカーシミュレーションにおけるプレイヤの配置問題においては，局所的な学習が可能な関数近似モデルの採用が望ましいという結論に至った．

局所的な学習が可能な関数近似モデルの代表的なものとして，動径基底関数 (Radial Basis Funciton : RBF) ネットワーク[5] や正規化ガウス関数ネットワーク (Normalized Gaussian network : NGnet)[6] などがある．しかし，これらのモデルは学習させるパラメータの数が3層パーセプトロンよりも多く，サンプルの数が増えるに従って学習にかかるコストが大きくなるという欠点を持つ．また，3層パーセプトロンと同様に，教示者の意図どおりの学習結果が得られるとは限らず，ある程度の推定は可能であるものの，得られる近似関数がブラックボックス化してしまうという問題も残される．そのため，3層パーセプトロンと同様に，獲得された関数に対して微調整を施すことは依然として困難である．

3 Delaunay 三角形分割を利用したエージェント配置手法

教師データとして与えられた各サンプルが局所的に作用し，他のサンプルへの影響を最少限に抑えるには，与えられたサンプルに基づいて空間を分割した上で，各サンプルの影響範囲をその分割領域に限定する，という方法が考えられる．そこで，我々は，美術館問題と同様に対象となる空間を三角形分割し，三角形の各頂点で得られる出力値の線形補間を行う，という手法を提案する．本稿では，三角形分割の手法として Deluanay 三角形分割を，線形補間の手法として単純な内挿法を用いる．

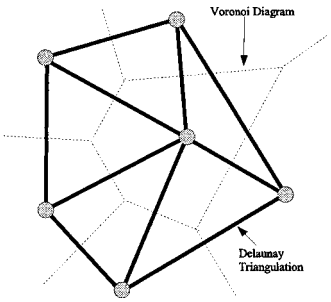

Fig. 1 Delaunay 三角形分割

3.1 Delaunay 三角形分割

Delaunay 三角形分割とは，" 平面上の点集合 P の三角形分割 T を構成した場合に，T に含まれる任意の三角形の外接円がその内部に P の点を含まない" 分割のことで，各三角形の最小の内角を最大にする（すなわち，三角形をなるべく細長くしない）という特徴を持つ．よって，Delaunay 三角形分割を求めることで，与えられた点集合に対して最も均質で安定な三角形分割を得ることができると言える．図 1 に Delaunay 三角形分割の例を示す．図中には，Delaunay 三角形分割と双対な関係にある Voronoi 図も描かれている．

与えられた点集合の要素数が3以上の場合，Delaunay 三角形分割はその点集合に対して一意に求めることができる．Delaunay 三角形分割を計算機上で求めるアルゴリズムはいくつか知られており，最も高速なアルゴリズムの計算量は $O(n \log n)$ である．よって，人間が把握できる程度の数の点集合であれば，リアルタイム性を損なうこと無く実時間での三角形分割の導出が可能である．本稿で用いるプログラムは，もっとも高速なアルゴリズムのひとつとして知られる確率的逐次添加法[2] を用いて実装した．

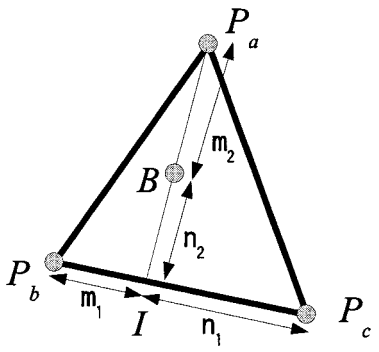

Fig. 2　グローシェーディングアルゴリズムによる線形補間

3.2　線形補間アルゴリズム

線形補間手法としては，実行速度を重視して，単純な 3 点の内挿法を用いる．これは，3 次元コンピュータグラフィックにおける陰影付け手法のひとつであるグローシェーディングアルゴリズム [8] と同じ計算方法である．

図 2 に，グローシェーディングアルゴリズムの計算過程を示す．三角形の頂点 P_a,P_b,P_c から得られる出力をそれぞれ $o(P_a)$，$o(P_b)$,$o(P_c)$ とすると，三角形の内部に含まれる点 B における出力 $o(B)$ は以下の手順で求められる．

1. P_a と B を通る直線と半直線 P_bP_c との交点 I を求める．
2. $|\overrightarrow{P_bI}| = m_1$，$|\overrightarrow{P_cI}| = n_1$ とすると，I における出力値 $o(I)$ は，
$$o(I) = o(P_b) + (o(P_c) - o(P_b))\frac{m_1}{m_1 + n_1}$$
3. $|\overrightarrow{P_aB}| = m_2$，$|\overrightarrow{BI}| = n_2$ とすると，
$$o(B) = o(P_a) + (o(I) - o(P_a))\frac{m_2}{m_2 + n_2}$$

3.3　サンプルからのエージェント移動位置の導出

本稿では，教示者が与えるサンプルのボール位置を Delaunay 三角形分割の頂点として扱う．各頂点には，そのボール位置に対してプレイヤが移動すべき位置座標が保持される．ボールがある三角形に含まれたとき，前節で述べた線形補間アルゴリズムによって，プレイヤが移動すべき位置座標が 3 頂点の出力値から補間されて出力される．

本稿では，速度を重視して単純な線形補間を用いるが，他の補間法を用いることでより複雑な配置も実現できる．例えば，階段関数を用いれば，相転移のような現象を起こすことも容易である．

3.4　提案手法の特徴

本稿で提案する手法は，以下のような利点を持つ

- 既存の非線形関数近似モデルと同等以上の近似精度を実現．与えたサンプルの入出力関係が完全に保証される．
- 非常にシンプルで高速に動作．与えるサンプルの数が数百個程度であれば，実時間での使用が可能．
- サンプルの追加や修正が行われても，影響が局所的である．追加・修正されたサンプルが属さない三角形領域には全く影響を及ぼさない．
- 獲得された性質を人間が視覚的に把握することが可能で，与えたサンプルが結果にどのような影響を与えているかを容易に推定することができる。
- 高い柔軟性．より滑らかな移動曲線や急激な勾配を持つ移動曲線が必要であれば，サンプルをより多く与えて移動位置を細かく指定するだけで良い．逆に，現在の配置が満足いくものであれば，サンプルの密度を高める必要は無い．
- 高いスケーラビリティ．対象となる空間が拡大・縮小しても，既存のデータに変更を加えることなく対応が可能．
- 完全な再現性．サンプルセットに含まれるデータが全て同一のものであれば，サンプルが与えられた順序に関係無く，完全に同一の結果を得られる．

特に，完全な再現性は非常に重要な利点である．サンプルの追加順序に制約が無くなることで，任意のサンプルを任意のタイミングで変更することが可能となる．これは，既に獲得された配置に対して，任意のタイミングで人間が介入できることを意味する．

4 シミュレーション実験

提案手法を用いてシミュレーションサッカーチームの配置を実際に形成し，サッカーシミュレータ上で試合を実行してパフォーマンスを評価する．比較対象として，NGnet による関数近似モデルを使用した同様の配置モデルを実装した．提案手法と NGnet には，それぞれ同一のサンプルが与えられ，得られた配置を用いて，同一の対戦相手と試合を実行する．

4.1 NGnet

NGnet は RBF ネットワークの派生手法であり，RBF ネットワークにおける各基底の出力を，全基底の出力の和によって正規化する点が異なる．正規化によって，任意の入力に対して 1 つ以上の中間ユニットの活性が保証される．RBF ネットワークでは，基底間の距離が大きい場合に基底の中間位置付近で出力が 0 に近付いてしまう問題が発生するが，NGnet では正規化によって基底間で必ず補間が行われる．エージェントの配置問題においては，出力が 0 になることは望ましくないため，本稿では RBF ネットワークではなく NGnet を採用した．

x を入力とすると，NGnet の出力層のユニット i からの出力は，以下の式で与えられる．

$$f_i(x,w) = \frac{\sum_{j=1}^{N} w_{ij}\phi_j(x)}{\sum_{j=1}^{N} \phi_j(x)} \quad (1)$$

w_{ij} は中間層から出力層への結合荷重である．$\phi_j(x)$ は中間層の各基底ユニットからの出力であり，以下の式で与えられる．

$$\phi_j(x) = \exp(-\frac{||x-c_j||^2}{2\sigma_j^2}) \quad (2)$$

ここで，c_j は各基底の中心位置，σ は基底の分散パラメータである．w_i,c_j,σ の 3 つのパラメータを獲得させなければならない．

結合荷重 w_i は，以下の更新式による再急降下法で学習させる．

$$w(t+1) = w(t) - \eta\frac{\partial \epsilon}{\partial w} + \alpha(w(t) - w(t-1)) \quad (3)$$

学習率 η を 0.1，修正モーメント法の係数 α を 0.5 とした．基底の中心位置 c_j には，サンプルとして与えたボールの位置座標をそのまま使用する．すなわち，サンプルと基底の数は同一となる．基底の分散 σ には，以下の式で得られるヒューリスティックな値を使用する．

$$\sigma = \frac{1}{N}\sum_{i=1}^{N} ||c_i - c_j|| \quad (4)$$

ここで，c_j は c_i の最近傍の基底を意味する．

4.2 FormationEditor

プレイヤの配置の形成においては，人間による俯瞰的な視点からの観察，そしてその観察に基づく直感的な調整が必要である．そこで，我々は，サンプルの編集作業を効率化可能にする GUI ツール，FormationEditor を実装した (図 3)．このツールを使うことで，配置の形成過程を視覚化しつつ，教師データとなるサンプルを編集することができる．

4.3 実験設定

サンプルセットととして 3 つのパターンを用意する．付録 A の図 4 から図 6 にサンプルの分布を示す．それぞれ，図中の×印の位置がサンプルのボール位置を表す．

まず，通常の試合で使用できる程度にまで作り込んだサンプルセットを用意した (図 4)．このサンプルセットでは，フィールド全域に渡ってサンプルが与えられており，両ペナルティエリア内の密度がやや高めになっている．

2 つ目のサンプルセットは，1 つ目のサンプルセットからペナルティエリア以外の領域に存在するサンプルを削除する (図 5)．ただし，フィールド中央とフィールドの四隅のサンプルは例外として残す．この偏りありのサンプルセットでは，両ペナルティエリア内に集中的にサンプルが与えられており，フィールド中盤との疎密の差が大きくなっている．

最後に，2 つ目のサンプルセットから更にサンプルを削除し，フィールド中央とフィールド四隅にのみサンプルを残した最少のサンプルセットを用意する．(図 6)．

これらを教師データとして用いて，提案手法と NGnet のそれぞれにプレイヤの配置を獲得させた．そして，獲得された配置を使って，特定の対戦相手との試合を実行する．実験用プレイヤプログラムとして，我々が開発した RoboCup サッカーシミュレーション用ベースチームである agent2d プログラ

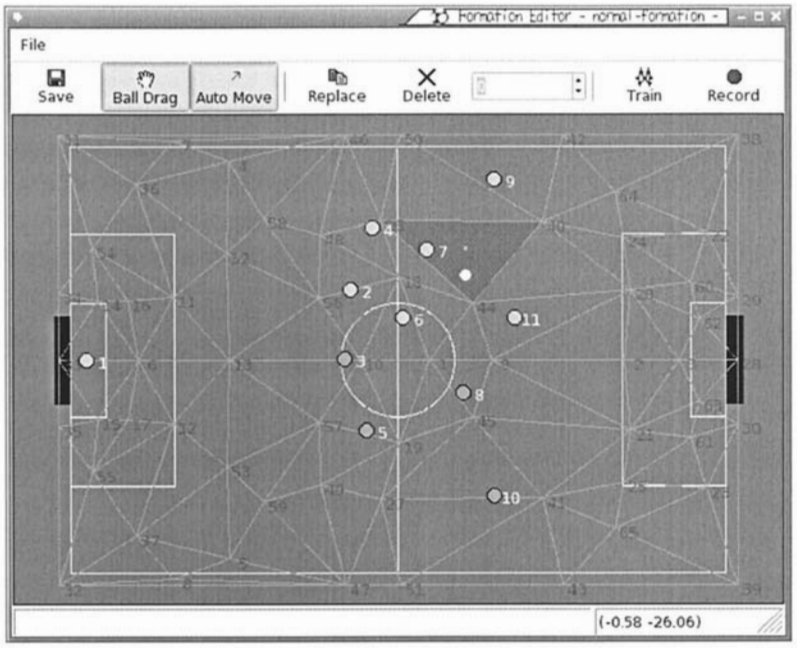

Fig. 3　FormationEditor の実行画面

ムを使用する[1]．対戦相手としては，RoboCup2005 に参加したチームのひとつである UvA Trilearn(アムステルダム大学) [2]を使用する．ランダムな要素を出来るだけ排除するために，プレイヤの能力は全て均一とする．

試合実行後，ログファイルから統計データを抽出する．抽出するデータは，プレイヤの配置の違いによる影響が出やすいであろう指標として，得失点，パスの成功回数，パスカットの回数とした．

4.4　実験結果

各配置データに対して，1 ハーフ 3000 サイクルの試合を 50 回ずつ行った，付録 B の表に，ログファイルから抽出したデータを示す．

表 2 の平均失点に注目すると，偏りのあるサンプルセットの場合に，提案手法は NGnet の 3 分の 1 近くまで失点が抑えらていることが分かる．逆に，得点は非常にわずかだが NGnet の方が多い．

次に，表 3 のパスの成功回数に注目すると，提案手法のほうがパス成功回数が多い．特に，偏りありのサンプルセットの NGnet の配置において，もっともパス成功回数が少なくなった．

4.5　考察

実験結果には，偏りのあるサンプルセットを用いて獲得した配置にもっとも大きな違いが現れた．これは，偏りのあるサンプルセットを用いて獲得した配置の特徴に原因があると推察できる．偏りのあるサンプルセットを用いた場合，NGnet で獲得される配置では，ボールがフィールド中盤にある状況では全プレイヤがほとんど静止している．ボールが両ゴールラインへ近付くと，全プレイヤが急激な移動を行い，ペナルティエリア周辺では提案手法とあまり差の無い配置となる．逆に，ゴールライン付近からフィールド中央へボールが戻る際にも，プレイヤの動き出しは鈍い．これに対して，全データの場合と最少データの場合とでは，いずれもフィールド全域においてボール位置に合わせたプレイヤの移動が確認できる．

NGnet での得点が僅かながらも多い理由は，一旦敵ペナルティエリア付近までボールを持ち込めれば，そのままプレイヤが前方に集まった状態が維持され，得点の機会が増えたためであろうと予想される．逆に，敵に攻め込まれた場合は，フィー

[1] http://sourceforge.jp/projects/rctools/ より入手可能.
[2] http://www.science.uva.nl/~jellekok/robocup/

ルド中盤から守備に戻る移動の開始が遅いために，守備が間に合わずに失点するのではないかと予想される．

パス成功回数についても，NGnet で回数が減った原因は，中盤でのプレイヤの移動量の減少が原因であろう．味方プレイヤがボールを持っていても，その動きに合わせて移動しようとしないために，パスコースが少なくなってしまったと予想される．

これらの結果から，NGnet では，対象領域内でのサンプルの密度の差が配置の特徴に大きく影響を与え，結果として，チームのパフォーマンスにも大きな影響を与えやすいことが予想される．これは，言い替えれば，各基底の分散のパラメータの調整が難しいことを意味する．すなわち，NGnet では，フィールドに対してなるべく一様にサンプルを用意しなければ，充分なパフォーマンスを得ることが難しいと言える．これに対して，提案手法では，サンプルの密度の差に関係無く配置の特徴が安定している．そのため，提案手法でサンプルを与える際に，その密度を考慮する必要は無い．

5 まとめ

実験で測った指標においてはあまり大きな差が見られなかったが，提案手法は多くの利点を持ち合わせており，非常に扱やすいモデルであるため，今後も有望な手法であると我々は考えている．しかしながら，提案手法には以下の欠点も存在する．

- メモリを大量に消費．3 層パーセプトロンや NGnet のような関数近似モデルであれば，ネットワークの結合荷重などのパラメータに教師データが吸収・圧縮されてしまうが，提案手法では教師データであるサンプルを全て保持しておかなければならない．
- サンプルの整合性の維持管理コストが高い．三角形分割が細かくなればなるほど滑らかな移動曲線を実現できるが，入力すべきサンプルの数が増えるだけでなく，それらの整合性を保つためのコストが高くなるという問題が発生する．また，サンプルの追加や修正による影響が非常に局所的であることは，逆にエージェントの移動曲線の滑らかさを損なう要因にもなりうる．場合によっては，周囲のサンプルも修正しなければならないことがある．

今後は，特にサンプルの維持管理コストを削減するために，サンプルの調整の自動化や，不整合なサンプルの検出を支援する手法の開発を試みる予定である．

また，他の課題として，多次元入出力への対応が重要である．現在は入出力がいずれも二次元であるが，実際には状態空間は無数に近い次元数を有している．出力に関しても，エージェントの移動以外の意思決定に関わる出力を得られるようになれば，より有益である．しかし，次元が増えると情報の可視化が困難になるため，上手く次元を圧縮したり，重ね合わせるなどの工夫が必要になるであろう．

参考文献

1) L. P. Reis et al. *Situation Based Strategic Positioning for Coordinating a Simulated RoboSoccer Team*, Balancing Reactivity and Social Deliberation in MAS, pp. 175–197, 2000.

2) M. de Berg et al. 浅野哲夫 訳, **コンピュータジオメトリ – 計算幾何学：アルゴリズムと応用**, 近代科学社, 2000.

3) The RoboCup Soccer Simulator, http://sserver.sourceforge.net/

4) I. Noda et al., *Soccer Server and Researches on Multi-Agent Systems*, Proceedings of IROS-96 Workshop on RoboCup, 1996.

5) T. Poggio et al., *Networks for approximation and learning*, Proceedings of the IEEE, 78, pp. 1481–1497, 1990.

6) J. Moody et al., *Fast learning in networks of locally-tunned processing units*, Neural Computation, 1, pp. 281–294, 1989.

7) H. Akiyama et al., *Team Formation Construction Using a GUI Tool in the RoboCup Soccer Simulation*, Proceedings of SCIS & ISIS 2006, 2006.

8) H. Gouraud, *Continuous shading of curved surfaces*, In Rosalee Wolfe(editor), Seminal Graphics: Pioneering efforts that shaped the field, ACM Press, 1998.

A　実験で使用したサンプルセット

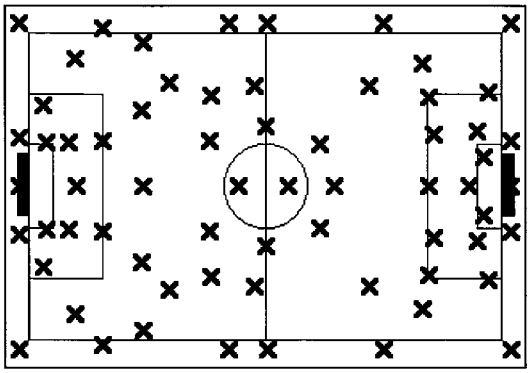

Fig. 4　サンプルセット（全データ）

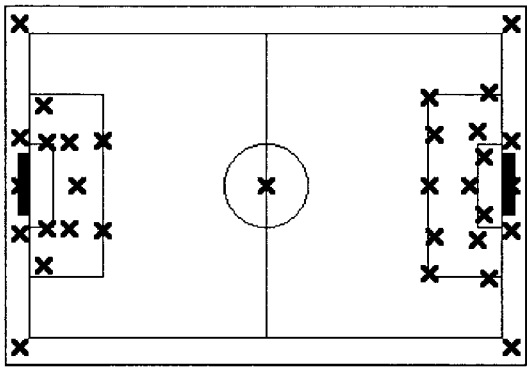

Fig. 5　サンプルセット（偏りあり）

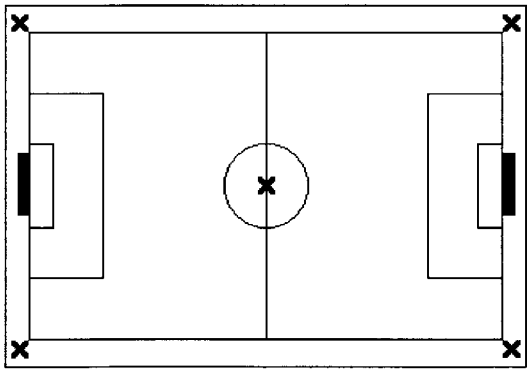

Fig. 6　サンプルセット（最少）

B　ログから抽出した統計データ

Table 1　各サンプルセットでの平均得点

	平均得点（標準偏差）	
	NGnet	提案手法
全データ	0.2(0.4)	0.08(0.27)
偏りあり	0.24(0.52)	0.06(0.22)
最少	0.02(0.14)	0.02(0.14)

Table 2　各サンプルセットでの平均失点

	平均失点（標準偏差）	
	NGnet	提案手法
全データ	0.98(0.89)	0.76(0.9)
偏りあり	1.76(1.24)	0.6(0.74)
最少	3.7(1.97)	3.16(1.67)

Table 3　各サンプルセットでの平均パス成功回数

	平均パス成功回数（標準偏差）	
	NGnet	提案手法
全データ	99.06(19.08)	110.6(18.79)
偏りあり	67.68(15.07)	91.44(21.35)
最少	83.86(19.1)	84.3(18.48)

Table 4　各サンプルセットでの平均パスカット回数

	平均パスカット回数（標準偏差）	
	NGnet	提案手法
全データ	20.2(5.06)	17.18(6.06)
偏りあり	18.8(5.52)	19.32(5.29)
最少	12.06(4.17)	14.5(5.21)