エージェント配置問題における三角形分割を利用した近似モデル

Triangulation Based Approximation Model for Agent Positioning Problem

秋山 英久
Hidehisa Akiyama

産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
hidehisa.akiyama@aist.go.jp

野田 五十樹
Itsuki Noda

(同上)
I.Noda@aist.go.jp

keywords: multiagent systems, agent positioning problem, function approximation model

Summary

In this paper, we propose a triangulation based function approximation model for agent positioning problem in the dynamic environments. In many problems of the real-world multi-agent/robot domain, a position of each agent is an important factor to affect agents' performance. Because the real-world problem is generally dynamic, a suitable position for each agent should be determined according to the current status of the environment. First, we formalized this issue as a function approximation model that maps from state variables to a desirable position of each agent, and proposed a function approximation model using Delaunay triangulation. This method is simple, fast and accurate, so that it can be implemented for real-time and scalable problems. In our previous works, our model showed very high approximation accuracy and good generalization capability for two-dimensional input. However, two-dimensional input is insufficient for more generic problems. Therefore, we extend our previous model so that multi-dimensional input can be taken. The extended model forms tree structure that each node represents a local input space. This structure enables us to maintain the multi-dimensional input space flexibly. The previous model is directly used in each local input space. Therefore, each local input space keeps high accuracy and generalization capability. We implemented the extended model and performed the experiments to evaluate its performance. The result shows our extended model can take the multi-dimensional input adequately.

1. はじめに

連続空間におけるマルチエージェントやマルチロボットの問題では、各エージェントやロボットの空間的な位置がそのパフォーマンスに重大な影響を及ぼす。動的な環境下での協調的な振る舞いを要求されるタスクでは、各エージェントが各々の役割や義務を果たすために、環境の状態に応じて継続的に移動し続けなければならない。このような、環境の状態に応じたエージェントの適切な配置を決定する問題を、本稿では「エージェント配置問題」と呼ぶ。特に、本研究では連続空間内での複数のエージェントの戦略的な配置を扱う。

エージェント配置問題において、特に配置の定量的な評価が困難なタスクに対しては、人間の直感に基づいた訓練データを用いる教師あり学習が効果的であろう。しかし、エージェントが移動すべき位置をすべての状態に対して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求める何らかの補間法が必要となる。また、本研究では人間が訓練データを与えることを想定しているため、人間の作業コストを軽減するために、与えるべき訓練データの数をなるべく少なく抑えられるモデルであることが望ましい。これらの要求に対して、我々の以前の研究において、Delaunay 三角形分割と線形補間を組み合わせたモデルを提案している。このモデルは有望な性能を示したものの、扱える入力空間が二次元に限定されているために、もっとも重要な二つの入力変数をあらかじめ選定しておかなければならず、より多くの状態変数を考慮しなければならない問題への適用が難しい。

そこで、本稿では我々の提案したモデルを拡張し、多次元入力へと対応したモデルを提案する。拡張したモデルでは、従来の提案モデルを階層化することで多次元入力に対応する。更に、木構造によって入力空間を部分的に多次元化することで、与える訓練データの数を少なく抑

えることを試みた．拡張したモデルを複数のエージェントによる閉鎖空間内での警護タスクへ適用する実験を行い，我々の提案するモデルによって少ない訓練データからも適切にエージェントの配置を獲得できることを示す．

第 2 章で関連研究について述べる．第 3 章で我々の提案するモデルの基本的な説明を行う．第 4 章で拡張したモデルの説明を行う．第 5 章で提案モデルを警護タスクへ適用した実験を示す．最後に第 6 章でまとめとする．

2. 関連研究

エージェントやロボットの配置を扱う問題として，多角形内での可視性を考慮した最適配置や巡回経路探索を目的とした警備問題が計算幾何の領域で盛んに研究されている [Berg 00, 譚 01]．これらの研究では，閉じた多角形内での最適配置や最適経路を解析的に求めることを目的とし，多くの派生問題が提案され，さまざまなアルゴリズムや定理が発見されている．ロボットの経路計画に対する解析的なアプローチとしては，ポテンシャル法 [Rimon 92, Koren 91] に基づく研究も盛んに行われている．これらのアプローチは，ある評価関数や制約条件に基づいた最短経路や最適配置を解析的に求める目的に適している．しかし，実環境におけるタスク（例えばサッカー）では，評価を定量化しにくい高度な戦略的配置が求められることがあり，これらの解析的アプローチが適さない場合がある．エージェントの配置に高度な戦略が要求される問題では，エキスパートの経験や知識，直感に基づいて配置を調整していくアプローチが効果的であろう．本研究では，エージェントの戦略的配置を獲得するために訓練データを人間が作成することを前提とし，その訓練データに基づいて効率良く配置を獲得することを目的としている．

我々の以前の研究では，RoboCup サッカーシミュレーション環境 (RCSS) [NODA 96] においてエージェントの配置獲得を試みた．RCSS では，Situation Based Strategic Positioning (SBSP) [Reis 01] という，ボール位置座標を入力，エージェントの移動位置座標を出力とする関数近似としてエージェント配置をモデル化する手法がよく知られている．SBSP は単純で扱いやすいモデルであるものの，関数の設定パラメータ値を人間が直接編集するという調整方法を採用しており，更に，パラメータ数が多いだけでなく，個々のパラメータ調整の影響が広範囲に及ぶために直感的な調整が難しいものとなっていた．そこで，我々は，SBSP と同様にエージェントの配置を関数近似の問題として捉えた上で，人間が直感的に作成した訓練データを用意し，教師あり学習による関数近似によって配置の獲得を試みた [Akiyama 08, Akiyama 07]．関数近似モデルとして，ニューラルネットワーク (BPN)，Radial Basis Function Network (RBF) [Poggio 90]，Normalized Gaussian Network (NGnet) [Moody 89]，

そして，我々が提案した三角形分割と線形補間を組み合わせたモデルを用意し，性能比較実験を行った．その結果，BPN は高い汎化性能を示すものの，個々の訓練データに対する近似精度は不十分で，教示者の意図通りの配置を実現することが困難であった．一方，RBF や NGnet では訓練データ周辺での再現精度は非常に高いものの，訓練データの分布に偏りがある場合に望ましい出力を得られにくかった．RBF や NGnet では，対象となる状態空間全域に渡って一様に分布する訓練データを用意すれば高い性能を発揮することが期待できるものの，大量の訓練データの作成は人間の教示者にとって大きな負担となってしまう．本研究では訓練データが人手によって作成されることを前提とするため，より少数の訓練データからでも教示者の意図を反映しやすい関数近似モデルが望ましい．我々の提案したモデルは，訓練データの影響範囲を予測しやすく，人間にとって扱いやすいものとなっており，この目的に適している．以前の実験では，少数の，あるいは，偏りのある訓練データからでも提案モデルによって適切な出力が得られることが確認できた．

しかしながら，我々の提案モデルは入力空間が二次元に限定されるという欠点を持っており，より複雑な条件を考慮しなければならない問題への応用が難しい．そこで，本稿では，従来モデルを拡張することで多次元入力に対応したモデルを提案する．実験では，厳密な経路計画と戦略的な配置が要求されるタスクとして，複数のエージェントによる警護タスクを設定し，提案モデルの有効性を検証する．

3. Delaunay 三角形分割を利用したエージェント配置手法

本節では，我々が提案する三角形分割と線形補間を組み合わせた関数近似モデルについて説明する．このモデルは，与えられた訓練データに基づいて入力空間を分割し，各訓練データの影響範囲をその分割領域に限定する，という考えに基づいたものとなっており，各訓練データの影響範囲の把握が容易である．ただし，このモデルは入力空間が二次元に限定されるという欠点を有している．以降，本稿ではこの関数近似モデルを基本モデルと呼ぶ．

3.1 基本モデルにおける出力の導出

訓練データが複数与えられると，それらの入力変数を頂点として入力空間を複数の三角形に分割することができる．三角形の各頂点は訓練データそのものであり，エージェントが移動すべき位置座標を出力として保持している．ある入力ベクトルに対する出力を得るには，訓練データによって生成された三角形集合の中から，その内部に入力ベクトルの点を含むものが存在するかどうかをまず探索する．入力ベクトルを含む三角形が発見されれば，その三角形の頂点が有する三つの訓練データを使用して出

図 1 Delaunay 三角形分割 .

力を補間する．入力ベクトルがどの三角形にも含まれなかった場合は，対象領域外の入力が与えられたものとして 0 を出力するなどの特殊な対応を行う．

本稿では，訓練データはノイズを含まず，すべて正確であると想定する．すなわち，統計的な学習は行われず，すべての訓練データに対して入出力関係が必ず一対一で決定することになる．実環境への適用においてはセンサノイズへの対応が必要となるが，本稿で提案するモデルはノイズへの対応を考慮していない．ノイズ除去などの処置は提案モデルへの入力の前処理段階において別のモジュールに委譲することを想定している．あるいは，ノイズを考慮した訓練データが必要であるならば，その対応は教示者である人間の責任となる．

基本モデルでは，三角形分割の手法として Delaunay 三角形分割を，補間アルゴリズムとしてグローシェーディングアルゴリズムを用いる．それぞれの手法について以降の節で詳しく述べる．

3.2 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 [Berg 00]．図 1 に Delaunay 三角形分割の例を示す．図中にも示されているように，Delaunay 三角形分割は Voronoi 図と双対な関係にあるグラフであり，点集合に基づいた凸包を形成する．

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．数百個程度の点集合であれば数ミリ秒で三角形分割を導出可能であり，教示者の操作に対して実時間性を損なうことなく応答することができる．本研究では人間の教示者が訓練データを適宜修正していくことを想定しているため，訓練データの修

図 2 グローシェーディングアルゴリズムによる線形補間 .

正が高速に反映されることは大きな利点である．

3.3 線形補間アルゴリズム

基本モデルでは，線形補間手法として単純な 3 点の内挿法を用いる．これは，三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 [Gouraud 98] 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 を通る直線と線分 $P_b P_c$ との交点 I を求める．
- (2) $|\overrightarrow{P_b I}| = m_1$, $|\overrightarrow{P_c I}| = n_1$ 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) $|\overrightarrow{P_a B}| = m_2$, $|\overrightarrow{B I}| = n_2$ とすると，

$$O(B) = O(P_a) + (O(I) - O(P_a)) \frac{m_2}{m_2 + n_2}$$

3.4 基本モデルの長所と短所

基本モデルでは Delaunay 三角形分割によって空間分割が行われるので，与えられた訓練データセットに対して一意な三角形分割を高速に導出可能である．一意な三角形分割が求められるという特性から，与えられた訓練データセットがまったく同一であれば，訓練データを与える順序に関係なく，得られる近似関数に完全な再現性が保証される．また，入力点が三角形分割の凸包内部に含まれる限り，この関数は常に連続であり，更に各訓練データの影響範囲はその入力点が属する三角形内部に限定されるため，人間にも入出力関係の予測が容易である．これは，ある訓練データの影響範囲を三角形の集合として視覚的に把握できることを意味し，人間による訓練データ作成作業においては大きな利点となる．

一方、基本モデルの短所としては入力空間が二次元に限定されるという点が挙げられる。理論上は、入力空間が多次元化されても Delaunay 三角形分割の方法論で空間を分割することは可能であるが、空間の分割状態を人間が理解可能な形で可視化することが困難になり、教示者が訓練データの関係や影響範囲を直感的に把握しづらくなる。その結果、直感的な操作による訓練データの修正も困難なものになってしまう。また、訓練データの作成において、冗長なデータを減らすために重要ではない入力変数を訓練データから省略したい場合があるが、Delaunay 三角形分割の方法論のままではやはり対応が簡単ではない。本稿では、基本モデルの長所を残しつつ、これらの問題へ対応することを目的とする。

4. 拡張モデル

本節では、多次元入力へ対応するために基本モデルを拡張したモデルについて説明する。以降、本稿ではこれを拡張モデルと呼ぶ。

4.1 入力空間の多層化

拡張モデルでは、基本モデルを木構造状に多層化することで多次元入力空間に対応する。基本モデルの入出力関係は図 3(a) のようになる。図中の $interpolate(O_1, O_2, O_3)$ は 3.3 節のアルゴリズムを意味する。これに対して、入力空間の多層化を図 3(b) のように行う。図に示すような木構造内の一つのノードを、本稿では部分入力空間と呼ぶ。この部分入力空間を木構造状に連結させることで多層化し、多次元入力への対応を実現する。

まず、部分入力空間 A に対していくつかの訓練データが与えられ、三角形分割が行われているとする。図中の頂点 a_1, a_2, a_3 が、与えられた訓練データの位置を表している。更に、各部分入力空間には *dontcare* 点というものを持たせる。図中の $a_{dontcare}$ が部分入力空間 A の *dontcare* 点を意味する。*dontcare* 点はその空間上の三角形分割とは独立しており、三角形分割には影響を与えない。*dontcare* 点はその部分入力空間を考慮しない場合に使用される。例えば、拡張モデル全体に与えられた入力ベクトルに A の成分が含まれていない場合、A では $a_{dontcare}$ 点を使用される。

各部分入力空間の頂点は子ノードとなる部分入力空間を持つことができる。例えば、図 3(b) では、頂点 a_1 には部分入力空間 C が属しており、 a_2 には部分入力空間 B が属している。部分入力空間 B 内の b_1 と $b_{dontcare}$ には更に部分入力空間 C が属している。 $a_1, b_1, b_{dontcare}$ に属する 3 つの部分入力空間 C は互いに独立である。これらの接続関係は、訓練データを与える過程で人間の教示者が構成する。

このように、部分入力空間 A がルートとなった n 分木の木構造が形成され、必要なだけ部分入力空間のノード

を積み上げていく。逆に、追加の部分入力空間が必要無ければ、 a_3 や b_2 のようにそこで終端となる。全体の出力は、木構造を再帰的に走査することで得られる。出力の具体的な導出手順は 4.2 節で説明し、木構造の形成手順は 4.3 節で説明する。

木構造を用いることで入力空間を部分的に多次元化され、訓練データに含める入力変数の一部を省略することが可能となる。複数の部分入力空間を決定木のように積み重ねていくことで部分的な修正を可能とし、要求される訓練データの数を減少させることが拡張モデルの目的の一つである。

4.2 出力の導出アルゴリズム

木構造全体での出力の導出手順をアルゴリズム 4.1 に示す。

Algorithm

4.1: CALCULATE($tree, inputList$)

```

procedure OUTPUT( $space, inputList$ )
   $input \leftarrow FINDINPUT(inputList, space.name)$ 
  if  $input == NULL$ 
    then  $\begin{cases} space \leftarrow space.dontcare.childSpace \\ \textbf{return} (OUTPUT(space, inputList)) \end{cases}$ 
    else  $inputList$  から  $input$  を削除
    SPACEOUTPUT( $space, input$ )

procedure SPACEOUTPUT( $space, input$ )
   $tri \leftarrow FINDTRIANGLE(space, input)$ 
  if  $tri == NULL$ 
    then  $\begin{cases} \textbf{return} (0) \\ \begin{cases} out_1 \leftarrow tri.v_1.output \\ + OUTPUT(tri.v_1.childSpace, inputList) \\ out_2 \leftarrow tri.v_2.output \\ + OUTPUT(tri.v_2.childSpace, inputList) \\ out_3 \leftarrow tri.v_3.output \\ + OUTPUT(tri.v_3.childSpace, inputList) \end{cases} \end{cases}$ 
    else  $\begin{cases} \textbf{return} (0) \\ \begin{cases} out_1 \leftarrow tri.v_1.output \\ + OUTPUT(tri.v_1.childSpace, inputList) \\ out_2 \leftarrow tri.v_2.output \\ + OUTPUT(tri.v_2.childSpace, inputList) \\ out_3 \leftarrow tri.v_3.output \\ + OUTPUT(tri.v_3.childSpace, inputList) \end{cases} \end{cases}$ 
    INTERPOLATE( $out_1, out_2, out_3$ )

main
   $space \leftarrow tree.rootSpace$ 
  return (OUTPUT( $space, inputList$ ))

```

ルートノードの部分空間から再帰的に木構造を走査し、各ノードの出力値の合計が最終的な出力値となる。 $Output(space, inputList)$ は再帰関数で、入力ベクトル $inputList$ に含まれる入力情報を参照しながら木構造を走査していく。 $FindInput$ は、第一引数の入力リストから第二引数の名前を持つ変数を探して返す手続きである。 $inputList$ には、各部分空間の名前とその変数のペアがリ

図 3 入力空間の多層化 .

ストで格納されている．各部分空間内での計算は基本モデルのアルゴリズムに基づいて行われるが，頂点が子ノードを持つ場合は，頂点の最終的な出力は子ノードの計算結果を反映したものになる．アルゴリズム中の *SpaceOutput* が各空間の出力を求める手続きで，ここでは *input* を内部に含む三角形を求めた後に，各頂点の出力に対して各頂点の子ノードの出力を加えている．*FindTriangle* は，部分空間 *space* 内の三角形分割の中から *input* を含む三角形を探して返す手続きである，*FindTriangle* の返り値は変数 *tri* へ代入されるが，該当する三角形が見つからなかった場合 *tri* の値は *NULL* となり，*SpaceOutput* を終了する．*tri* は三つの頂点 v_1, v_2, v_3 を持っており，各頂点は子ノードの部分空間 *childSpace* を持つことがある．そのような場合，それらの子ノードは再帰的に走査される．*Interpolate* は，3.3 節の線形補間アルゴリズムを意味する．

図 3(b) を例として考えると，部分入力空間 B からの出力値を求める場合， b_1 からの出力値は， b_1 自身が持つ出力値に加えて子ノードの部分入力空間 C からの出力値を加算したものになる．そして，部分入力空間 B での出力値を補間によって求め，その値は a_2 へと渡される．

4.3 訓練データ追加アルゴリズム

拡張モデルで使用する訓練データは，入力変数のリストとそれに対応する出力の組で与えられる．入力変数リストは任意の長さを取ることができ，入力変数リストに含まれない部分入力空間は *dontcare* として扱われる．訓練データが新規に追加されると入力空間の木構造の再構成が行われ，訓練データの追加が木構造のルート空間から再帰的に実行される．訓練データの追加手順をアルゴリズム 4.2 に示す．

Algorithm

4.2: ADDNEWDATA(*tree*, *data*)

```

procedure ADDDATA(space, data)
  input  $\leftarrow$  FINDINPUT(data.inputList, space.name)
  if input == NULL
    then ADDDATA(space.dontcare.childSpace, data)
  else input を data.inputList から
    data.removedInput へ移動
  if inputList is empty
    then {
      vertex  $\leftarrow$  input と一致する頂点
      if vertex == NULL
        then vertex  $\leftarrow$  新しい頂点
      vertex.input = input
      vertex.output = data.output
      - CALCULATE(tree, data.removedInput)
    }
  else {
    vertex  $\leftarrow$  input と一致する頂点
    if vertex == NULL
      {
        vertex  $\leftarrow$  新しい頂点
        tri  $\leftarrow$  FINDTRIANGLE(space, input)
        out1  $\leftarrow$  tri.v1.output
        out2  $\leftarrow$  tri.v2.output
        out3  $\leftarrow$  tri.v3.output
        vertex.output
          = INTERPOLATE(out1, out2, out3)
        vertex.input = input
        ADDDATA(vertex.child, data)
      }
  }

```

main

```

space = tree.rootSpace
ADDDATA(space, data)

```

変数 *data* は訓練データそのものを表し, *data.inputList* は訓練データに含まれる入力変数のリストを表す. 走査中の部分入力空間 *space* に対応する入力変数が *data.inputList* に含まれない場合, 訓練データに対して *space* を *dontcare* とみなし, *space* 上の *dontcare* 点に属する子ノードへと移動する. *space* に属する *input* が見つかった場合, *input* は *data.inputList* から *data.removedInput* へ移動する. このとき, *data.inputList* が空であれば *space* は終端ノードとなる. *input* と一致する頂点が *space* 上に存在すれば, その頂点の出力値が上書きされ, 一致する頂点が存在しなければ新しい頂点が作成される. このとき, 出力値は親ノードまでの出力値との差分を取るため, 現在の空間を含まない出力計算を *Calculate* によって行う. *data.inputList* に入力変数が残っている場合は, *space* 上に *input* と一致する頂点が存在すれば, その頂点に属する子ノードへ移動する. 一致する頂点が存在しなければ新規に頂点を作成するが, このとき追加される頂点の出力値はその頂点を含む三角形から補間される.

4.4 定義域の決定

入力空間を多層化する際, 同時に各部分入力空間の定義域を決定する. この定義域とは, ある閉領域でその境界上での出力が必ず 0 になり, 閉領域より外の領域の出力も 0 となるものと定義する. 部分入力空間が二次元平面の場合は, 訓練データによる三角形分割を包含する任意の凸包多角形を定義域として適用できる. 訓練データに含まれる入力変数は与えられた定義域内に含まれなければならない, 定義域外の入力を与えられた場合は無効な訓練データとして破棄する.

三角形分割は与えられた定義域内で形成されるが, 定義域の各頂点も三角形分割の頂点として扱われる. ただし, 定義域の頂点には子ノードを持たせず, その出力は常に 0 とする. 定義域の頂点を仮想的な訓練データとすることで, 定義域内に一つ以上の訓練データが与えられれば必ず三角形分割が行われ, 出力の補間と連続性が保証されるようになる. さもないと, 部分入力空間ごとに必ず三つ以上の訓練データが要求されることになり, 訓練データの追加作業コストが極めて煩雑なものになってしまう.

定義域は部分入力空間の特性に応じて適切に決定しなければならない. 定義域の面積が大きくなれば, その部分入力空間が影響を及ぼす範囲も大きくなるため, 一つの訓練データが予想以上に広範囲に作用する可能性が高くなってしまふ. 定義域も訓練データと同様に随時修正可能であり, 必要に応じて教示者が修正していくことを想定しているが, 部分入力空間が影響を及ぼす範囲を見積り, 適切な定義域を与えることが肝要であるため, 扱う問題に対する経験や知識がある程度要求されることになるだろう.

4.5 訓練データの依存関係

各部分入力空間は木構造上に親子関係を持って連結されるため, 親のノードに対して依存関係を持つことになる. 訓練データを追加する際はこの依存関係を考慮する必要がある, より依存関係の大きい部分入力空間は木構造のルート近くに配置すべきであろう. 例えば, 移動物体を警護するタスクの場合であれば, 警護対象物体の位置座標をルートノードに設定するのが妥当である. 警護対象物体にもっとも近い味方エージェントの位置座標と二番目に近い味方エージェントの位置座標を用いるのであれば, 警護対象物体により近い方の味方エージェントを上位のノードにすべきであろう.

一方, 依存関係の小さい部分入力空間の扱いに関しては議論の余地が大きい. 警護対象物体にもっとも近い味方エージェントと警護対象物体にもっとも近い敵エージェントを考えると, どちらをより上位のノードに置くかを定量的に判断することは難しい. 拡張モデルでは, これらの接続関係をどのように決定するかは教示者の判断に委ねることを想定している.

5. 実験

前節で述べた拡張モデルを実装し, 複数のエージェントによる移動物体の警護タスクを想定した環境において, 実際にエージェントを配置する実験を行った.

5.1 警護タスク

警護タスクにおける各エージェントの目的は, 警護対象物体を常に視界に捉えられるように対象物体を追従することである. エージェントと警護対象物体の間に壁が存在した場合, エージェントは警護対象物体を視界に捉えることが出来ない. また, エージェントと警護対象物体のいずれも壁を通り抜けることはできず, エージェントが壁に近づきすぎるとエージェントと壁との衝突が発生する. エージェント同士が互いに近づきすぎた場合にも, エージェント同士の衝突が発生する. 実世界においては, 衝突は物理的な被害を発生させるため, 回避すべき現象である. よって, 今回の問題設定においても, エージェントに衝突が発生しないように経路選択を行わなければならないことにする. また, エージェントは閉鎖空間の外へ出られないので, 閉鎖空間の外への配置が要求されてはならない. 本稿では, 教示者の意図としてこれらの目的を達成する配置の獲得を行う. 警護タスクは計算幾何分野における警備問題の派生版と言えるが, 複数の自律移動エージェントの戦略的配置が要求される点で拡張されている.

図 4 に警護タスクの実例を示す. この図はいくつかの訓練データが既に与えられた状態であり, 図中の番号付きの×印は入力空間として警護対象物体の位置座標を用いた場合の訓練データの分布を表している. 図中の黒い

図 4 複数エージェントによる移動物体の警護タスク．

枠は壁を表しており，壁で囲まれた部屋のような閉鎖空間を想定している．この閉鎖空間の大きさは横が約 90m，縦が約 60m とし，空間内の壁の配置を変えたパターンをいくつか用意する．この座標系の原点は閉鎖空間の外側左上付近に位置し，閉鎖空間の中央の座標値は (57.5, 39.0) となる．図中の小さい円が警護対象物体で，この物体は閉鎖空間の内部を自由に移動する．番号を振られた円がエージェントを表しており，大きさは半径 1m とした．今回の実験では，警護対象物体の数を 1，エージェントの数を 2 とした．

5.2 各関数近似モデルの設定

まず，警護タスクに対して基本モデルを適用し，他の代表的な関数近似モデルと比較することでその性能を検証する．ここでは，入力として警護対象物体の絶対位置座標を使用する．比較する関数近似モデルとして，BPN，RBF，NGnet，k-nearest neighbor(k-NN)，タイルコーディング [Richard S. 00] を用いる．

§ 1 BPN

BPN は中間層を一つ持つ標準的な 3 層構造とし，中間層，および，出力層での各ユニットの発火関数としてシグモイド関数を用いる [熊澤 98]．結合加重は [-0.5, 0.5] でランダムに初期化し，誤差逆伝播法で学習する．入出力値は [0, 1] に正規化し，出力の誤差が閾値 (0.003) 以下になるか，5000 回ループするまで学習を実行する．結合荷重の学習率，修正モーメント法の係数，および中間層のユニット数は実験的に定め，それぞれ 0.3, 0.9, 20 とした．BPN に対しては，訓練データを一度に与えるバッチ学習 (BPN-batch) と，データの一つ与えるごとに学習を行う逐次的学習 (BPN-inc) の二種類の方法を用意した．逐次的学習を行う場合，訓練データは教示者が与えた順番で適用される．これは，訓練データの一つ与えるごとに現在の配置獲得状況を確認する，という手順を繰り返すことに相当する．

§ 2 RBF, NGnet

RBF および NGnet の基底関数としては，ガウス関数を用いる．NGnet は，RBF における各基底ユニットの

図 5 左上付近に分布が偏った訓練データセット．

出力を全基底ユニットの出力の和によって正規化する点のみが異なる．本稿では，訓練データがすべて正確であるという前提であるので，一つの訓練データに対して一つの基底ユニットを対応させる．すなわち，訓練データ数と基底ユニットの数は常に一致する．各基底の中心位置 c_j は訓練データが持つ入力位置として固定し，基底位置の学習は行わない．基底の分散についても学習は行わず，以下のヒューリスティックな値を用いる [Moody 89, 坂和 97]．

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

ここで， N は基底ユニットの数， c_i は c_j から最近傍の基底位置を意味する．

以上の設定より，求めるべきパラメータは基底関数の線形結合の係数のみとなるので，これは最小二乗法で求める．

§ 3 k-NN

頂点間の位置関係に基づいて使用する頂点の選択が行われるという点で，基本モデルは k-NN との関連が高い．本稿では目的関数が実数値を扱うので，距離重み付け k-NN を採用する [Mitchell 97]．入力点 x_q から距離が近い順に k 個の訓練データを選択し， x_q と各訓練データ x_i との距離の二乗の逆数 w_i を求める．

$$w_i = \frac{1}{d(x_q, x_i)^2} \quad (2)$$

ここで， $d(x, y)$ は x と y の間のユークリッド距離である．すると， x_q における出力 $\hat{f}(x_q)$ は，

$$\hat{f}(x_q) = \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i} \quad (3)$$

ここで， $f(x_i)$ は x_i の出力である．ただし， x_q が x_i のいずれかと一致している場合は， $f(x_i)$ を $\hat{f}(x_q)$ に一致させる．

予備実験として， k の値を 3 から 6 まで変化させて性能を測ったところ， $k = 3$ が平均的にもっとも性能が良かった．以降の実験では， $k = 3$ の設定を用いる．

(a) 警護対象物体を見失った割合 .

(b) 壁と衝突した割合 .

(c) 閉鎖空間の外へ飛び出した割合 .

(d) エージェント同士が衝突した割合 .

図 6 各関数近似モデルの評価 .

§ 4 タイルコーディング

基本モデルは状態の分割を人間の直感によって行い、その分割に基づいた補間を行う手法である。状態分割に基づいた関数近似モデルとしてはタイルコーディングがよく知られている。ここでは、最も単純な格子状のタイル分割方法を用いる。各タイリングの一回の試行におけるステップサイズパラメータは、タイリングの数が m の場合には $\frac{1}{10m}$ とする。タイリングの数と格子の大きさにいくつかのパターンを用意して予備実験を行い、平均的に最も性能の良かったものを使用する。今回は、1 タイリングあたりのタイル数を $5 \times 4 = 20$ 、タイリングの数を 44 とした。この設定で入力空間の解像度は約 880 となり、実験環境においては約 $9m^2$ の粒度となる。各タイルの更新は出力が収束するまで実行される。

5.3 使用する訓練データセット

訓練データセットとして以下の 7 パターンを用意した。

- 5m 間隔で訓練データを与えたもの。
- 10m 間隔で訓練データを与えたもの。
- 20m 間隔で訓練データを与えたもの。
- 30m 間隔で訓練データを与えたもの。
- 壁の頂点の位置にのみ訓練データを与えたもの。
- 閉鎖空間内の左上付近に集中的に訓練データを与えたもの。
- 閉鎖空間内の中央付近に集中的に訓練データを与えたもの。

訓練データが持つ入力警護対象物体の位置座標とし、図 4 の環境では、上から順に 234 個、74 個、35 個、20 個、16 個、48 個、58 個の訓練データが用意される。上

の4パターンでは訓練データの入力値は閉鎖空間内ではほぼ様に分布しているが、下の3パターンでは入力値の分布に偏りがある。図4は20m間隔で訓練データを与えた場合の入力値の分布を表しており、これに対して、図5は閉鎖空間内の左上付近に集中的に訓練データを与えた場合を表している。

これらの訓練データセットはすべて人手で作成する。その際、壁を回避するための配置を与えるために、すべての訓練データセットにおいて壁の頂点の位置に訓練データを用意する。また、壁の内部や閉鎖空間外部には訓練データを用意しない。このため、各訓練データ間の距離は一定ではなく、若干の歪みが存在する。

5.4 二次元入力空間での実験結果 1

まず、図4に示す形状の空間に対して、すべてのモデルを適用した結果を示す。警護対象物体を閉鎖空間内で0.5m間隔で移動させ、その位置座標を入力とした場合の各エージェントの移動位置座標を出力し、いくつかの評価基準に基づいて評価する。評価した入力位置の総数は、図4の空間内で合計18588点となった。警護タスクの目的が達成されたかどうかを示す評価基準として、警護対象物体を視界に捉えられなかった割合、エージェントが壁と衝突した割合、エージェントが閉鎖空間の外へ飛び出した割合、エージェント同士が衝突した割合を計測する。BPNに関しては、同一の訓練データセットに対して10回の平均を取った。図6にこれらの結果を示す。

各グラフは、縦軸の値が小さいほど警護タスクの目的に対して性能が良いことを示している。ほとんどのモデルで訓練データ数が少なくなるにつれて性能が低下しているのに対し、基本モデルはすべての場合において他のモデルよりも良い結果を示している。特に、訓練データに大きな偏りが無い場合には、基本モデルは完全にタスクの目的を達成することができた。一方で、意図的に偏りを大きくした訓練データセットの場合には、基本モデルでも完全な性能を得られていない。得られた結果を観察したところ、これは、基本モデルによって生成される三角形分割が閉鎖空間の形状と合致していないために起こる現象であることが確認できた。閉鎖空間の左上付近に偏って訓練データを与えた場合、基本モデルでは図7の状態の三角形分割が得られる。図7は閉鎖空間の左上付近を拡大表示したものであるが、この図より、壁にまたがった三角形が存在していることが分かる。このような三角形が存在すると、出力の導出において壁の反対側に存在する訓練データの影響を受ける場合があり、これが完全な性能を得られなかった原因となっている。そこで、壁をまたがった三角形が存在しないように訓練データを調整してみたところ、問題を修正できることが確認できた。この修正作業は、三角形分割の状態を観察することで直感的かつ容易に行うことができた。

他のモデルでは、k-NNとBPN-batchが比較的良好

図7 閉鎖空間の左上付近に偏った訓練データによる三角形分割。

性能を示しており、対象物体を見失った割合と壁との衝突の割合は常に10%以下に収まっている。NGnetとタイルコーディングも比較的安定した性能を示しているものの、訓練データ数の減少に伴う性能の劣化が若干大きくなっている。RBFについてはその傾向が更に顕著で、訓練データの分布に偏りがある場合には、エージェントが空間の外へ飛び出す割合が他よりも大きくなっている。これは、RBFの基底の分散の大きさによっては、基底の中間位置付近の入力に対してRBFの出力が0に近づきやすくなってしまうためだと考えられる。実際に、エージェントの移動範囲をプロットすると、頻繁に座標系の原点へ向かって移動しようとしている様子が伺えた。その一方で、5m間隔の訓練データを使用した場合、RBFは非常に高い性能を示しており、RBFの分散パラメータの調整が性能に大きく影響することが伺える。BPN-incは非常に不安定な結果を見せており、特に訓練データ数が多い場合は過学習が発生していたと予想される。

今回使用したk-NNではユークリッド距離を使用したのが、これをマンハッタン距離に置きかえるなど、壁を回避した経路での距離を考慮することで結果が改善される可能性がある。タイルコーディングにおいても、今回の実験ではタイルの分割方法が単純な格子状であったが、[Sherstov 05]や[Whiteson 07]のような、タイルの数の数やタイルの分割形状を自動的に獲得する手法を応用することで性能の改善が期待できる。これらは今後の検討課題である。

5.5 二次元入力空間での実験結果 2

続いて、基本モデルのスケラビリティを測るために、壁のパターンが異なる空間に対して同様の実験を行った。閉鎖空間の形状として図8の6パターンを用意した。空間全体の大きさと原点の位置は図4とほぼ同様である。これらの閉鎖空間パターンに対して、5.3節の訓練データセットを用意し、基本モデルに適用した。ただし、この実験では意図的に偏りを設定した訓練データセットは省略する。

実験の結果、基本モデルでは、対象物体の可視性や壁の回避は完全に達成できるものの、特定の形状を持つ環境においてエージェント同士の衝突が発生することが分かっ

図 8 閉鎖空間のパターン .

表 1 各閉鎖空間パターンにおけるエージェント同士の衝突回数 .

	訓練データのタイプ				
	5m	10m	20m	30m	Corner
図.8(a)	10	37	79	83	1029
図.8(b)	8	11	193	199	199
図.8(c)	0	0	0	0	0
図.8(d)	0	0	0	0	0
図.8(e)	7	33	94	150	150
図.8(f)	0	0	0	0	0

図 9 壁面に沿った三角形分割が行われなかった場合のエージェントの移動範囲 .

た．表 1 に各閉鎖空間パターンにおけるエージェント同士の衝突の発生回数を示す．図 8(a),8(b),8(e) では，5m 間隔の訓練データセットであっても，発生回数は減るものの衝突を完全には回避できていない．特に，図 8(a),8(b) のような，エージェントの移動経路が 8 の字にねじれてしまう環境においては，入力として警護対象物体の位置を考慮するのみではエージェント同士の衝突をスムーズに回避することは難しい．エージェント同士の衝突をスムーズに回避するには，移動位置の履歴や他のエージェントの位置などの，新しい入力変数の導入が必要である．

また，図 8(f) のパターンにおいて壁の頂点のみに訓練データを与えた場合に，警護対象物体を見失った割合が約 8.74%，壁との衝突が約 10.79% となり，他のパターンと比べて急激に性能が悪化した．図 9 は，この訓練データに基づいた三角形分割と，一方のエージェントの移動範囲の分布を示している．この図から，中央付近の壁の側面に三角形が重なっており，壁の壁面に沿った空間分割が行われておらず，壁の反対側の頂点からの影響を強く受けてしまい，移動位置の出力が壁の内部に侵入してしまったことが見て取れる．これは，前節の実験 1 で偏りの大きい訓練データを与えた場合と同じ現象である．これらの結果から，警護タスクに基本モデルを適用する際は，入力値が取り得ない値の範囲（今回の実験設定では，壁などの固定障害物）を避けて空間分割が行われるように，訓練データを工夫しなければならないことが分かる．ただし，基本モデルは人間にとって理解しやすい性質を持っているため，得られた結果から調整作業を行うことは非常に容易であることも確認できた．

5.6 拡張モデルによる配置獲得実験

前節の実験によって示した，エージェント同士の衝突の回避が困難な環境に対して，拡張モデルを適用することで問題の解決を試みる．拡張モデルに対する入力変数として以下の二つを設定した．

- 警護対象物体の絶対位置座標
- エージェントの互いの相対位置座標

このうち，警護対象物体の絶対位置座標は常に使用する入力とし，拡張モデルのルートノード空間として扱う．警護対象物体の定義域は閉鎖空間全域を覆う矩形，エージェントの互いの相対位置座標の定義域は各エージェントを中心とした 10m x 10m の矩形とした．エージェントの互いの相対位置座標がこの矩形の範囲外であれば，互いの位置関係が結果に及ぼす影響は 0 となる．

まず，前節の実験と同じ訓練データを与えてルートノード空間における三角形分割を形成する．ここでは，それぞれの閉鎖空間パターンにおいて最も衝突回数が多かった，壁の頂点に対してのみ訓練データを与えた訓練データセットを用いる．次に，エージェントの互いの相対位置座標を入力に含めた訓練データを与えることで，ルートノード空間の頂点に子ノードの部分入力空間を与える．今回は，ルートノード空間の各頂点に対して同一の子ノード空間を与える．子ノード空間へ与えた入力は， $(+2.12, 0.0)$ ， $(-2.12, 0.0)$ ， $(+0.1, 0.0)$ ， $(-0.1, 0.0)$ ， $(0.0, +2.12)$ ， $(0.0, -2.12)$ ， $(0.0, +0.1)$ ， $(0.0, -0.1)$ の 8 つで，それぞれにルートノードの出力値に対する差分値を与える．実際に子ノード空間を形成するための訓練データを与えると，図 10 のような状態になる．図中には，ルートノード空間の三角

図 10 子ノードの部分入力空間が与えられた状態。

形分割に加えて、中央付近に子ノード空間の定義域と三角形分割が三つ表示されている。子ノード空間はルートノード空間の頂点に属するため、図ではルートノード空間の頂点の位置に対応させて表示している。

ここで、子ノード空間への入力を加えた訓練データを、ルートノード空間のどの位置へ追加すべきかを決定しなければならない。その方法として以下の3つが考えられる。

- (1) ルートノード空間に存在するすべての頂点に対して与える。
- (2) ルートノード空間において衝突が発生した入力点を含む三角形を探し、その頂点に対して与える。
- (3) ルートノード空間において衝突が発生している領域の重心位置付近へ、ルートノード空間の新しい頂点とともに与える。

方法(1)は明らかに冗長である。方法(2)は、基本モデルの性質から、衝突が起こる入力に対して確実に修正が反映されることが期待できる。しかし、方法(2)では、もし衝突がルートノード空間の広い範囲で発生していれば多くの三角形が該当することになり、その結果、ほぼすべての頂点に対して追加の訓練データを与えることになりかねない。これに対して、方法(3)では、最小限の訓練データで衝突回避を実現できる可能性がある。ここでは、前節で衝突回避に失敗した閉鎖空間に対して、方法(2)と(3)に基づいて訓練データを与える。

5.7 拡張モデルの実験結果

図 11 に、図 8(a), 8(b), 8(e) の閉鎖空間における衝突発生位置を示す。図にはルートノード空間の訓練データに基づいた三角形分割も表示している。図中に+印で衝突発生位置を示しており、網掛け状になっている部分は衝突が連続的に発生した領域とみなせる。図 11(b), 11(c) では、いずれも閉鎖空間の中央付近で集中的に衝突が発生しているが、図 11(a) では4ヶ所に分散して衝突が発生していることが分かる。これらに対して上記の二通りの方法で子ノード空間のための追加訓練データを与える。

図 11(b), 11(c) では、衝突位置を含む三角形の数は2であり、合計の頂点の数はいずれも4である。これに対し

て、図 11(a) では、衝突位置を含む三角形の数は8で、合計の頂点の数は12になる。方法(2)ではこれらの頂点に追加訓練データを与える。方法(3)では、図 11(b), 11(c) に対しては閉鎖空間の中央付近に一つだけ追加訓練データを与え、図 11(a) に対しては衝突発生位置の分布の重心付近の合計4ヶ所に追加訓練データを与える。

以上の基準に基づいて、実際に追加の訓練データを与えて配置を獲得させ、5.4節の実験と同じ評価を行った。その結果、対象物体の可視性、壁との衝突回避、閉鎖空間の外への飛び出しについての性能を保ったまま、エージェント同士の衝突を完全に回避することができた。図 12 から図 14 に、一方のエージェントの移動範囲の変化の様子を示す。これらの図から、衝突の発生箇所周辺でエージェントの移動範囲の変化が観察できる。特に、図 14 では、衝突が発生していた領域を回避している様子が良く分かる。これらから、方法(3)によって方法(2)と同等の性能が得られていることが見て取れる。追加された訓練データの数を方法(2)と(3)で比較すると、図 8(b)と8(e)に対しては4倍、図 8(a)に対しては12倍もの差がある。警護タスクにおけるエージェント同士の衝突回避という目的においては、追加すべき訓練データの増加を抑えることに成功したと言えるだろう。

6. ま と め

本稿では、我々が提案した三角形分割と線形補間を組み合わせた関数近似モデルを拡張し、多次元入力への対応を可能にした。実験では、まず、より一般的なタスク設定において基本モデルと他の関数近似モデルとの比較を行い、訓練データの数が少ない場合や訓練データの分布に偏りがある場合にも、基本モデルは他のモデルよりも良い性能を示すことを確認した。更に、基本モデルを木構造上に多層化することで入力空間の部分的な多次元化が可能となり、警護タスクにおいて要求される訓練データ数を少なく抑えられることを実験的に示した。拡張モデルの汎用性を示す定量的な評価には至らなかったが、拡張モデルは警護タスクのような問題設定において教示者の意図する配置を実現するための有望な手法と言えるだろう。

しかしながら、基本モデルは訓練データを適切に反映することができたが、この性質は逆に欠点にも成り得る。基本モデルでは、訓練データが常に正しくノイズを含まないことが前提であるため、ノイズを含む訓練データを大量に与えられる場合には対応が困難となる。このような何らかの統計的学習が必要となる問題設定では、BPNやRBF、タイルコーディングなどの他の関数近似モデルが有利となる。ノイズを含む訓練データに対しても基本モデルの性質を生かす手法を検討することは今後の課題の一つである。

また、拡張モデルによって多次元入力への対応が可能

図 11 衝突発生位置の分布 .

図 12 拡張モデルの利用による移動範囲の変化 (図 6(a)) .

図 13 拡張モデルの利用による移動範囲の変化 (図 6(b)) .

図 14 拡張モデルの利用による移動範囲の変化 (図 6(e)) .

となったが、拡張モデルには訓練データの入力と管理にかかる作業が複雑なものになってしまうという問題も考えられる。これに対応するには、訓練データの作成を支援するGUIツールプログラムのユーザインタフェースを改善することで対応することを検討している。例えば、部分入力空間の依存関係をより直感的に把握し操作できるように、部分入力空間の接続関係を立体的に表示する機能や、木構造の構成を直接操作するためのインタフェースなども必要だろう。そして、要求される訓練データ数の増加を抑えられるとしても、やはりその総数は二次元入力の場合よりもはるかに増加するため、今後は訓練データの作成を支援する手法の確立も必要となるだろう。適切な空間分割の獲得に関しては、三角形分割との親和性が高い Growing Neural Gas [Fritzke 95] を応用することで、自動あるいは半自動で獲得することを検討している。

謝 辞

本研究の一部は、NEDO 戦略的先端ロボット要素技術開発プロジェクト 被災建造物内移動 RT システム（特殊環境用ロボット分野）閉鎖空間内高速走行探査群ロボットによる研究助成によって行われたことを付記いたします。

◇ 参 考 文 献 ◇

- [Akiyama 07] Akiyama, H., Katagami, D., and Nitta, K.: Training of Agent Positioning using Humans's Instruction, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 11, No. 8, pp. 998–1006 (2007)
- [Akiyama 08] Akiyama, H. and Noda, I.: Multi-Agent Positioning Mechanism in the Dynamic Environment, in *RoboCup 2007: Robot Soccer World Cup XI* (2008), to appear
- [Berg 00] Berg, Mark de M. v. K., Otfried Schwarzkopf and Overmars, M.: *Computational Geometry: Algorithms and Applications*, Springer Verlag, second edition (2000)
- [Fritzke 95] Fritzke, B.: A Growing Neural Gas Network Learns Topologies, in *Advances in Neural Information Processing System 7*, MIT Press (1995)
- [Gouraud 98] Gouraud, H.: Continuous shading of curved surfaces, in Wolfe, R. ed., *Seminal Graphics: Pioneering efforts that shaped the field*, ACM Press (1998)
- [Koren 91] Koren, Y. and Borenstein, J.: Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, in *Proc. of the IEEE int. Conf. on Robotics & Automation*, pp. 1398–1404 (1991)
- [Mitchell 97] Mitchell, T.: *Machine Learning*, McGraw-Hill (1997)
- [Moody 89] Moody, J. and Darken, C. J.: Fast learning in networks of locally-tuned processing units, *Neural Computation*, Vol. 1, No. 2, pp. 281–294 (1989)
- [NODA 96] NODA, I. and MATSUBARA, H.: Soccer Server and Researches on Multi-Agent Systems, in Kitano, H. ed., *Proceedings of IROS-96 Workshop on RoboCup*, pp. 1–7 (1996)
- [Poggio 90] Poggio, T. and Girosi, F.: Networks for approximation and learning, *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481–1497 (1990)
- [Reis 01] Reis, L. P., Lau, N., and Olivira, E.: Situation Based Strategic Positioning for Coordinating a Simulated RoboSoccer Team, *Balancing Reactivity and Social Deliberation in MAS*, pp. 175–197 (2001)

- [Richard S. 00] Sutton, R. S., Barto, A. G., 三上 貞芳・皆川 雅章 共訳：強化学習，森北出版 (2000)
- [Rimon 92] Rimon, E. and Koditschek, D. E.: Exact Robot Navigation using Artificial Potential Functions, *IEEE Transaction on Robotics and Automation*, Vol. 8, No. 5, pp. 501–518 (1992)
- [Sherstov 05] Sherstov, A. A. and Stone, P.: Function Approximation via Tile Coding: Automating Parameter Choice, in Zucker, J.-D. and Saitta, I. eds., *SARA 2005*, Vol. 3607 of *Lecture Notes in Artificial Intelligence*, pp. 194–205, Springer Verlag, Berlin (2005)
- [Whiteson 07] Whiteson, S., Taylor, M. E., and Stone, P.: Adaptive Tile Coding for Value Function Approximation, Technical Report AI-TR-07-339, University of Texas at Austin (2007)
- [熊澤 98] 熊澤 逸夫：学習とニューラルネットワーク，森北出版 (1998)
- [坂和 97] 坂和 正敏，田中 雅博：ニューロコンピューティング入門，森北出版 (1997)
- [譚 01] 譚 学厚，平田 富夫：計算幾何学入門 幾何アルゴリズムとその応用，森北出版 (2001)

〔担当委員：山口 智浩〕

2007 年 10 月 30 日 受理

著 者 紹 介

秋山 英久(正会員)

1999 年東京工業大学工学部情報工学科卒業。2001 年同大学院総合理工学研究科物理情報システム創造専攻博士前期課程修了。同年同研究科知能システム科学専攻博士後期課程に入学。2006 年，独立行政法人産業技術総合研究所情報技術研究部門テクニカルスタッフ。RoboCup サッカーシミュレーションを中心に，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。人工知能学会，情報処理学会会員。

野田 五十樹(正会員)

1992 年京都大学大学院工学研究科電気工学第 2 専攻博士課程修了。同年，通商産業省工業技術院 電子技術総合研究所。2002 年，改組により，独立行政法人産業技術総合研究所 サイバースタッフ研究センター。2004 年，同研究所情報技術研究部門主任研究員。1999–2000 年，スタンフォード大学言語情報研究センター滞在研究員。博士（工学）。神経回路網によるシンボル・パターン統合処理，機械学習，マルチエージェントシステム，社会シミュレーション，災害情報システムの研究に従事。人工知能学会，情報処理学会，日本神経回路学会員。RoboCup Federationa，ロボカップ日本委員会，防災推進機構理事。