人工知能学会
第12回 SIG-Challenge 研究会

2001

2001年4月27日
福岡工業大学
（第4回ロボカップジャパンオープン 2001）
目次

1. 方策勾配法を用いた移動ロボットの行動計画法 - 問題の多様性への対応 -
   五十嵐 治一 (近畿大学) ........................................ 1

2. 対戦型マルチエージェントシステムにおけるチーム構成の戦略の獲得
   田村 隆, 高橋 泰岳, 浅田 稔 (大阪大学大学院) .................. 7

3. PVM を用いたロボットのタスクの分散化に関する検討
   中野 博史, 鈴木 英智, 長内 真人, 安井 憲太郎, 渡辺 一朗, 櫻尾 次郎 (三重大学) ............... 12

4. KU-Boxes2001 における高速色抽出処理
   田中 一基, 影山 茂, 五十嵐 治一, 黒瀬 能幸 (近畿大学) .......... 18

5. RoboCup シミュレーションリーグ人間参戦システム OZ-RP の提案
   秋田 純一 (はこだて未来大学), 西野 順二 (電気通信大学), 久保 長徳 (福井大学), 下羅 弘樹 (福井大学), 藤塚 健 (豊橋技科大学) .................. 23

6. 情報伝による移動ロボットの注視制御のためのセンサ空間構築
   光永 法明, 浅田 稔 (大阪大学大学院) ............................... 29

7. KU-Boxes2001 における走行加速度の向上と経路計画について
   早津 優子, 飯野 修一, 五十嵐 治一, 黒瀬 能幸 (近畿大学) .......... 35

8. ヘテロジニーニアスチーム OZ における協調的行動の分析
   伊藤 慶浩 (名古屋工業大学), 西野 順二 (電気通信大学), 森下 卓哉 (福井大学), 久保 長徳 (福井大学) .................. 40
方策勾配法を用いた移動ロボットの行動計画法
一問題の多様性への対応一

Motion Planning of Mobile Robots by a Policy Gradient Method
- Application to Various Problems-

五十嵐治一
Harukazu Igarashi
近畿大学工学部（広島県東広島市）
School of Engineering, Kinki University
igarashi@infohiro.kindai.ac.jp

Abstract
In a previous paper, we proposed a solution to motion planning of a mobile robot. That approach is a reinforcement learning approach based on a policy gradient method. In this paper, we consider possibility whether this approach can be applied to various motion planning problems of mobile robots under the environments of multiple robots, moving obstacles, and partially observed Markov decision processes.

1 はじめに

自律移動型ロボットの軌道・経路計画問題に関しては、様々な方法が提案されてきた[1]〜[10]。しかし、現実世界における軌道・経路計画問題は多目的多制約の計画問題となるのが一般的である。

そこで、従来、移動型ロボットの経路計画においては、ポテンシャル法、スケルトン法、空間分割法といった手法[7,10]が提案されてきた。これらの手法は主として路線の最短性及び障害物回避との観点を重視しており、対策とする問題の前提条件やタスクの追加・変更に対しては、アルゴリズムやプログラムの大幅な修正を余儀なくされる場合もししばしば見受けられた。

そこで、著者は、ロボットの軌道・経路計画問題の分野において、柔軟で一般性があり、かつ実現が容易な手法を確立することを目的として、軌道・経路計画問題を各時刻における敵状最適化問題に帰着させる方式を提案してきた[11]。さらに、その最適化問題で用いた目的関数の重み係数の値を、強化学習の一種である方策勾配法(policy gradient method)により自動学習させるという手法を提案した[12][13]。

ただし、これまでの応用例では、単数ロボットが静止
障害物の間を移動するという単純な場合のみを考えてきた。かつ、障害物の形状や配置によっては、局所的にトラップされてしまう場合も観測された。

そこで、本報告では、複数台ロボットの軌道・経路計画を立案する場合や、障害物が移動する場合、環境のように局所的にトラップされやすい環境の場合、行動決定が部分観測マルコフ決定過程となる場合への対応を考察する。なお、本研究では、計画立案者の存在と、軌道・経路に対する報酬を与えるユーザ、または、ユーザの報酬と等価な価値基準が存在することを前提としている。

2 行動計画の基本アルゴリズム

2.1 目的関数

前記でも述べたように、自律移動型ロボットの軌道・経路計画問題（以下では、両者を合わせて”行動計画問題”と称する）には、たとえ、既知環境下であっても、制約条件、前提条件、ユーザの要求する仕様[7][11]等による多様性が存在する。その多様性が、走行経路のアルゴリズムを複雑にしている要因と考えられる。本章では、すでに著者が提案している行動計画方式[12][13]を簡単に説明する。

まず、次の目的関数

$$E_r(v_r, r_r, r_{goal}) = b_r E_{goal} + b_r E_{cost} + b_r E_{con}$$  (1)

ここで，$$r_r$$ は離散時刻 t におけるロボットの位置，$$v_r$$ は時刻 t におけるロボットの速度ベクトル，$$r_{goal}$$ はゴール地点の位置である。この目的関数は、時刻 t において速度ベクトル $$v_r$$ の値を選択した場合の不適当な度合い（ベナルティの量）を表現している。

最初の項 $$E_{goal}(v_r, r_r, r_{goal})$$ はロボットがゴール地点の方向へ進むことを要求している引力項である。具体的には
次の関数を用いた。

\[ E_{goal}(v_i, r_i, r_{goal}) = \text{sgn}(G(v_i)) \cdot |G(v_i)| ^2 \]  \hspace{1cm} (2)

ただし，関数 \( G(v_i) \) は次式で定義されている。

\[ G(v_i) = \left\| r_{goal} - r_{goal}^i \right\| - \left\| r_i - r_i^i \right\| \]  \hspace{1cm} (3)

上式中，時刻 \( t \) におけるロボットの位置 \( r_i \) は，時刻 \( t \) におけるロボットの位置 \( r_{goal} \) に対する距離を示す。時間間隔 \( \Delta t \) により，以下のように定義されている。

\[ r_i^t = r_i + v_i \Delta t \]  \hspace{1cm} (4)

式(3)の第2項 \( E_{goal}(v_i, r_i) \) は滑らかであるため，次式で定義する。

\[ E_{smooth}(v_i; r_{goal}) = \left\| v_i - v_{goal} \right\|^2 \]  \hspace{1cm} (5)

ここで，\( v_{goal} \) はゴール速度のユークリッドノルムを表す。この項はロボットがゴールに至る間の速度の変化を滑らかにすることを示している。

式(1)の最後の項 \( E_{coll} \) は障害物との衝突を避ける力項であり，次式で定義する。

\[ E_{coll}(v_i, r_i) = \begin{cases} D_{col} & \text{if} \quad \text{Dist}(r_i^t) < 0 \\ -D_{col}^2 & \text{if} \quad 0 < \text{Dist}(r_i^t) < R \\ R^2 & \text{if} \quad \text{Dist}(r_i^t) > R, \end{cases} \]  \hspace{1cm} (6)

ここで，\( \text{Dist}(r) \) は，位置 \( r \) における障害物過度の距離を示した関数である。本手法では，障害物が障害物の距離，および位置を近似するため，環境を模擬的に表現されている。ただし，計算は障害物および位置を正確に模倣した環境を模擬的に表現されている。計算は障害物が障害物を避けるために，常に大きな値を設定する。式(6)の力項は，障害物が障害物を避けるため，距離が \( R \) 以内であれば，距離の2乗に比例した力が生じることを意味している。

2.1 探索空間

本章，目的関数 \( E \) は速度ベクトル \( v_i \) の連続関数であるが，ここで，速度ベクトルの探索空間を離散化し，原点を中心とする円内に探索範囲を限定する。離散ベクトルの空間を離散化するため，探索空間の分割数，探索精度と処理速度を平衡することができるからである。すなわち，探索精度の向上は，探索空間の分割数を細かくし，逆に，処理速度を優先したい場合には，分割を粗くすれば良い。探索空間の離散化の例を図1に示す。

2.3 期待報酬値と確率的政策

\( E_r \) を最小化して得られる軌道・移動は，式(1)中の重み係数 \( \{b_i(\alpha, \beta) \} \) に大きく依存する[11]。そこで最適な値を決定するために強化学習を用いた方法を考察した[12]。本節と次節とで簡単に述べる。

まず，学習の目的を期待報酬の最大化と定める。移動ロボットの行動計画の場合は，軌道 \( u \) に与えられた報酬 \( R(u) \) の期待値を最大化するように，各時刻 \( t \) における速度ベクトル \( v \) を選択する方策 \( \pi \) の中に含まれるパラメータを決定することが本方式における学習である。

ここでは，Suttonらと同様に，期待報酬値を \( \rho(\pi) \) で表し，以下の様に定義する[14]。

\[ \rho(\pi) = E[R(u)] \]  \hspace{1cm} (7)

\[ = \sum_u P(u) \cdot R(u) \]  \hspace{1cm} (8)

ここで，報酬 \( u \) は時刻 \( t \) におけるロボットの位置 \( r_i \) の時系列であり，\( P(u) \) はロボットの軌道が \( u \) となる確率である。また，式(7)の右辺の記号 \( E[\ldots] \) は期待値による期待値操作である。期待値操作を考えるのには，次に述べるように，必ずしも目的関数 \( E_r \) の最小値（実際はある点の近傍での極小値）を与える速度ベクトルの値を決定論的に選ぶのではなく，確率的に選択するからである。

ある時刻 \( t \) において適切と思われる速度ベクトルを選択することは“方策”と称したが，本方式では，方策 \( \pi \) として以下の Boltzmann 分布による確率方策 (probabilistic policy) を用いる。

\[ \pi(v_i, r_i, r_{goal}, \{b_i \}) = \frac{e^{-E_r(v_i)}}{\sum_v e^{-E_r(v_v)}} \]  \hspace{1cm} (9)

ただし，\( E_r(v_i) \) の第2項である滑らか項は，式(5)より1ステップ前の時刻の状態に依存するので，方策 \( \pi \) も1ス
テープ前の時刻の状態に依存している。これにより、式(9)の方策を用いた行動決定の過程は、非マルコフ的な行動決定過程となっている。従来より、強化学習の有力な方法としてQ学習が知られているが、Q学習の場合、行動決定の過程として非マルコフ過程を想定している。したがって、式(5)のような非マルコフ的な知識を利用した行動計画の場合、Q学習をそのまま適用することは難しく考えられる。

2.4 確率的勾配法による期待報酬値の最大化

以上の議論の下で、期待報酬の$\rho(\pi)$の勾配$\partial / \partial \theta \ln \pi(v; \theta)$を計算法し、確率的近似法(Robins-Monroアルゴリズム)の一種である確率的勾配法[15]を用いると、重み係数$\theta_k$に関する次の学習則を得る。

$$\Delta \theta_k = \epsilon \cdot R(u) \cdot \frac{1}{T} \sum_{t=1}^{T} \frac{\partial}{\partial \theta_k} \ln \pi(v_t; \theta_k)$$

$$= \frac{1}{T} \sum_{t=1}^{T} \frac{\partial R(u)}{\partial \theta_k} - \frac{\partial R(u)}{\partial \theta_k}$$

ここで、$\epsilon$は学習係数で、ときには正の定数である。右辺に出てくる微分係数の値は、目的関数$E_q(\theta)$の形なれば、特に容易に計算することが可能である。式(11)の中$X_{t}(\theta)$は各時刻における速度ベクトル$v_t$に関する平均操作

$$X_{t}(\theta) = \frac{1}{T} \sum_{t=1}^{T} X \cdot \frac{e^{-E_q(v_t; \theta, (v_{t-1}, (0)))}}{T}$$

を表している。この処理においても、図1のように速度ベクトルに関する探索空間を離散化し、ベクトルの大きさを限定しているので、計算量が抑えられている。

なお、式(10)の学習則は、Williamsのepisodic REINFORCEアルゴリズム[16]と同一である。

3 方策の空間依存性と時間依存性

本章では、政策$\pi$の空間依存性と時間依存性について考察する。

3.1 走行経験から得られる局所情報の利用

式(11)の目的関数を各時刻において決定論的に最小化しててしまうと、状況によってはローカルミニマムに陥り、局所的にトラップされてしまうことがある。図2のその例を示す。式(9)の確率的方策では、温度パラメータ$\Gamma$の値を調整することにより、この状況をある程度回避することは可能であるが、障害物の形状や配置によっては調節が難しい。

例題1: 平均の右側で表された3つの条件は、報酬の高い軌道・経路を生成するために時間が端的に与えた先見的知識を表現している。そこで、図2のようなローカルミニマムに陥りやすい場合や、迷路からの脱出問題のように、各地点においてどういう方向へ行けば目的を達成できるかという情報をそれまでの道路経験をもとに計算しておき、同じ地点に来たときにほどの値をもとに行動を決定する方法の方が役に立つ場合がある。これを一般的に言うと、エージェントが先に経験を通して得ることができた経験的知識を方策に反映させるということである。Q学習のQ値はこの経験的知識に相当すると考えられる。

このように局所に依存した経験的知識を表現するために、式(11)の目的関数$E_q(\theta)$に式(12)の関数$E_q(\theta)$を追加する。すなわち

$$E_q(v_t; \theta) = b_{E_1} E_{pos} + b_{E_2} E_{neg} + b_{E_{c1}} E_{c1} + b_{E_q} E_q$$

ただし、$E_q$は

$$E_q(v_t; \theta) = -\sum_{r} q(r,v) \cdot \delta_{v_t, r}$$

で定義する。ここで、位置座標$r$と時刻$t$におけるロボットの位置$r_t$とは何らかの方法で離散化されており、速度ベクトル$v_t$も図1のように離散化されているとする。$s$はクローンカーの$s$記号である。この項目の意味は、位置$r$において速度ベクトル$v$を選択したときの妥当性を、パラメータ$q(r,v)$で表現しており、方策$\pi$の学習に役立てるということである。

式(13)のパラメータ$b_{E_1}$と式(14)のパラメータ$q(r,v)$を、式(11)に示した学習則を用いて全く同様に学習することが可能である。パラメータ$q(r,v)$の値を格納する表のサイズが大きすぎてならならない。したがって、入力層$q(r,v)$を表すパーセプトロンに入力し、出力層$q(r,v)$の値を出力する多階層パーセプトロン型のニューラルネットワークモデルで計算$q(r,v)$を近似する方法が考えられる。

また、時間の与えた先見的知識が空間依存性を持つこ
とがわっている場合には、式(1)や式(13)で示した重み係数 \( b_k \) (k=1,2,3) の空間的なパラメータ \( b_k(t) \) (t=1,2,3) を用いて計算した。式(11)の学習則に含まれる重み係数の違いは、姫の場面数において、主に障害物に衝突する危険性を表す重み係数 \( b_{kr} \) の値の相違である。逆に、障害物の多い場面においては、この逆の状況が望ましいと言ったような場合である。

### 3.2 時間変動する環境下での方策学習

前節では、行動決定を行う際に、場面に依存した知識または主に時刻数の中に含まれるパラメータの空間依存性により表現した。しかし、環境が変動するように、環境が時間変動する場合には、この知識もしくは方策学習を提案する。そこで、元の知識は、式(1)中のパラメータを \( b_k(t) \) (t=1,2,3) の時刻変動するパラメータと考えた場合の方策学習を考察する。

式(11)で示された両方の問題 \( E_{(v)} \) のかわりに、離散時刻 \( t \) における方策 \( E_{(v)} \) を考え、

\[
E_{(v)}(v; t) = b_1(t)E_{(x)} + b_2(t)E_{(y)} + b_3(t)E_{(z)}
\]

ただし、重み係数 \( b_k(t) \) (t=1,2,3) は時間の関数であり、目的の位置 \( r_{opt} \) が障害物の位置を時間の関数とする。

このとき、式(11)の学習則は、

\[
\Delta b_k(t) = -\frac{\epsilon}{T} \left( R(t) \cdot \left[ \frac{\partial E_{(v)}}{\partial b_k(t)} \left( \frac{\partial E_{(v)}}{\partial b_k(t)} \right)^{r_{kr}} \right] \right)
\]

となる。式(11)との違いは、式(11)では生成された軌道上の各点に沿って、\( \Delta b_k(t) \) は以下の計算を示し、その合計を取るようである。式(16)では相対的に変化すること、これは、時間依存性が強いため重み係数の値は、軌道上のすべての点における行動決定に影響を与えるが、重み係数の時間依存性がある場合には、異なる時刻の重み係数は互いに独立の変数であり、他の時刻の行動決定に直接には関与しないからである。

### 4 マルチロボットシステムにおける行動計画

#### 4.1 集中的な計画方式と自律分散的な計画方式

前節までは、ロボット１台の行動計画を対象としてきたが、本稿では複数台のロボットシステム（マルチロボットシステム）の行動計画を立下ることを考える。マルチロボットシステムの行動計画を立下する際には、すべてのロボットの行動をひとまとめにして取り扱う集中的な計画方式（集中方式）と、個々のロボットの行動をそれぞれ個別に決定する自律分散的な計画方式（自律分散方式）とに大別される。

行動計画問題を効率最適化問題に帰着させるという立場からみると、前項の方式では、解の精度は良いが、重み係数が非対称で解がポジティブルを持つ数がロボット台数の数の増加に従って増加することを示唆する。また、ロボット同士の衝突が必ずしも遺伝子に制限される場合で、お互いの行動計画に影響を及ぼすできないのがロボット同士が接近し合った場合に限られる。したがって、行動計画を立下する方法は、段階的な、全ロボットの行動の組み合わせを考える集中方式では、決定の効率が良いとは言えない。そこで、本稿では、後者の自律分散方式に基づいて、方策学習の適用を検討することにした。

### 4.2 ロボット同士の衝突回避と優先順位

複数台のロボットが協力するシステムにおいて、多くの場合、ロボット同士の衝突を回避する軌道・経路を計画することがしばしば要求される。この目的を実現するための一つの方法として、ロボット間の近距離衝突の影響を持つ行動に有効であると考えられる。

ここで、ロボットごとに次のように行動計画を考える。

\[
E_{(v)}(v; t) = b_1 E_{(x)} + b_2 E_{(y)} + b_3 E_{(z)} + b_{mn} E_{(mn)}
\]

これで、第4項の \( E_{mn} \) は、他のロボットと同時に働く近距離衝突力の影響を考慮した行動計画のための行動計画である。
ような場合を想定していた。
本章では、ロボット自身が、未知環境下で自身の行動判定方法（方策π）を自らの走行経験により学習することを考える。この場合、ロボットは自らの位置xを直接に知ることはできず、自らを含む環境をサンプルにより得られる観測値xだけでなく行動判定を行う必要がある。通常、センサの観測能力は完全ではないが、かつ、確実でないのでも、現在制のロボットの位置を同定することは難しくロボットの行動判定過程をマルコフ決定過程で厳密に定式化することは難しい。観測が不完全・不確実のために、自らの状態を含めて環境に不確実性がある場合のエージェントの行動判定過程は、一般に、部分観測マルコフ決定過程（PMODP: Partially Observable Markov Decision Process）と呼ばれている[18]。PMODPにおいても、本アプローチが適用できることを以下に示す。

2章で述べた行動計画法を、未知環境下での各時刻における行動判定方法として考え、式(1)の目的関数を計算するための必要情報は、時刻tにおけるロボットの位置xを、これは観測Xにより推定された位置xを代用する。正確な位置推定を行うには、何らかの位置推定（周辺化、ローカライゼーション）の操作を行う必要がある[11,13,17]。一方時刻における予測位置xもxはt=Dx大气で代用する。次に、状態方程式の計算で使用する障害物までの最短距離值Dist(t)は、距離センサにより得られた情報から推定する。さらに、各時刻における目的地点までの相対距離や方向が何らかの方法によりわかる場合には、引力方程式に用いている関数G(a)の中に反映させることも可能である。

上記の方法における目的関数の重み係数の学習則は、式(11)に示したもと形式上は全く同じものになる。この学習則は、PMODPにおける強化学習法として、木村らが提案している確率的傾斜法の文脈[18]中の定理2で報酬の割引率γの値を1とした場合の学習則になっている。ただし、本方式の場合、方策πや学習則は現時刻における状態xの関数ではなく、センサ値から推定された状態xの推定値の関数xは観測xを用いており、センサの観測値から現時刻におけるロボットの状態（位置と向き）を推定する必要がある。

また、行動判定において、本方式のように目的関数の最も大小という定式化を用いて、「ニューラルネットワークモデルのような関数近似法を用いて、観測Xと方策πや学習則を直接関連付けることも可能である[16][18]」方策πとして階層型のニューラルネットワークモデルを用いる方法は、すでにWilliamsにより提案されている[16]

6 おわりに

本報告では、先に提案した方策学習を用いた移動型ロボットの軌道・経路計画問題の解法を、いくつかの場合に拡張することを試みた。
まず、行動計画に関する先見的な知識だけではローカルミニマムに陥りやすい場合や、先見的知識以外の経験的な知識の表現方法として、空間に依存したパラメータの導入により、方策に空間依存性を持たせることを提案した。次に、移動障害物が存在する場合のように環境が時間変動する場合には、パラメータに時間依存性を持たせることにより対応する方法を考察した。
さらに、マルチロボットシステムにおいて、個々のロボットの行動計画を自律分散的に行う方針の一端として、方策学習による本計画方式の適用可能性を示した。最後に、環境に不確実性のある部分観測マルコフ決定過程への適用について考察した。今後は、計算機シミュレーション等で本論文において提案したアルゴリズムの検証を進めて行く予定である。

謝辞
本研究に関してご討論いただいた。本学附属ウッド助教授に感謝の意を表します。なお、本研究は、日本学術振興会より科学研究費補助金（C2）23240480の助成を受けた。

参考文献

[8] 小方博之, 新井英夫, 太田利, ”時変環境でユーザ仕様を考慮した移動ロボットの軌道計画法, ” 日本ロボ
[13] 五十嵐治一," 離散最適化問題としての走行誘導・経路計画と強化学習によるパラメータ決定法", 人工知能学会第 6 回SIG-Challenge研究会資料, pp.7-12(00, 6 月, 函館)
対戦型マルチエージェントシステムにおける
チーム構成の戦略の獲得

Strategy Learning For A Team In Adversary Environments

田村 隆 高橋 泰岳 浅田 稔
Takashi TAMURA, Yasutake TAKAHASHI, and Minoru ASADA
大阪大学大学院 工学研究科 知能・機能創成工学専攻
Dept, of Adaptive Machine Systems, Graduate School of Engineering Osaka University
{tamtam,yasutake,asada}@er.ams.eng.osaka-u.ac.jp

Abstract

Team strategy acquisition is one of the most important issues of multi-agent systems, especially in an adversary environment. RoboCup has been providing such an environment for AI and robotics researchers. A deliberative approach to the team strategy acquisition seems useless in such a dynamic and hostile environment. This paper presents a learning method to acquire team strategy from a viewpoint of coach who can change a combination of players each of which has a fixed policy. Assuming that the opponent has the same choice for the team strategy but keeps the fixed strategy during one match, the system estimates the opponent team strategy (player’s combination) based on game progress (obtained and lost goals) and notification of the opponent strategy just after each match. The trade-off between exploration and exploitation is handled by considering how correct the expectation is. A case of 2 to 2 game was simulated and the final result (a class of the strongest combinations) was applied to RoboCup-2000 competition.

1 Introduction

チーム戦略の獲得はマルチエージェントシステムにおいて、特に敵対するエージェントが環境に存在するとき非常に重要な問題の一つである。ロボカップではAIとロボット研究者のためにこのような環境を提供してきた。この環境では従来から研究されてきたマルチエージェント環境と比べ環境の変動が速く実時間処理が必要とされ、さらに敵の存在のためモデルや計画を立てることを困難にしている。そのため環境の情報を直接使ったフィードバックを使った制御が有効である[1]。

シミュレーションリーグではさまざまなチーム戦略の獲得の手法が提案されて来た。一方で実機リーグではハードウェアやシステム全体の構成の方に重点があかれ、それほど多くの研究がチーム戦略の獲得に関して行われて来たわけではない。Castelpietra et al. [2]は無線を通じて情報や役割を交換することにより協調動作を創発する手法を提案した。Uchibe et al. [3]はタスクの達成度を共有メモリに書き込むことで動的に役割分担させる枠組を提案した。しかしこれらの手法はあらかじめチーム戦略を固定していると考えられる。

本論文ではエージェントがそれぞれの個性を持った固定政策をもち、エージェントの組合せを決定できるコーチの視点からチーム戦略を獲得する学習方法を提案する。敵チームも同種類のチーム戦略の選択肢があり、ゲーム中は戦略を固定している。学習システムはゲーム中敵チームの戦略を予測し最適なチーム戦略を選択しながら戦い、戦後に敵チームの戦略を教えてもらうとする。学習中の探索するための戦略が最適な戦略をとるかのトレードオフは試合結果が予測とあってはならないかにより自動的に決定する。2対2の試合におけるシミュレーションを通じて本手法の有効性を示す。

2 問題設定

環境がより複雑になるにつれ、システムはそれに対応するためにより複雑になっていく傾向にある。特にマルチエージェントシステムでは協調動作を実現するためにコミュニケーションは有効なようにみえる。しかしながら環境が非常に動的ので、それゆえに環境の変化の予測が難しくなるような環境では役に立たない。行為に基づく手法はこのような動的に変化する環境に対応できる一つの方法である。しかし対戦型マルチエージェント環境においてそれぞれのロボットを設計することは相手の戦略を把握して
知ることができないので困難である。
そこで我々は以下の条件のもとでチーム戦略を獲得する学習手法を提案する。

- それぞれのチームは多様性を持った選手を擁し、それぞれの選手が互いにコミュニケーションせず、ビークルベースドアプローチによって設計されている。

- チーム戦略は選手の組合せによって定義される。

- 監督は選手を交替させることでチーム戦略を変える。

- 相手チームも同じ能力を持った選手を選び、同じ戦略をとることができる。

ゲームの進行状況から監督の視点で相手のチームに勝てる戦略を見つけすることが目的となる。したがって監督は相手チームの戦略を試合経過を通して推定し、自分のチーム戦略との関係を獲得していく必要がある。そこで以下の仮定をおく。

- 相手チームは毎試合ごとに戦略をランダムに変更するが、試合中は変更しない。一方自チームの監督は試合中に何度か戦略を変更することができる。

これ以降、監督とは学習するチームの監督を指すものとする。

3 チーム戦略学習

まず学習するチームの監督にとって二つの政策を用意する。

チーム探索政策　得失点表を作成するための探索行動

最適チーム推定政策　相手のチームを推定し、勝てるチームを算出

チームの選択政策はチーム探索政策と最適チーム推定政策の重みつきの足し合わせで決定する。その重み \( w_e \) はチーム探索政策と最適チーム推定政策の比率を決めており、最適チーム推定政策の推定結果が適切になるにつれてチーム探索政策から最適チーム推定政策へ移行していくように重み \( w_e \) を更新する。j番目の戦略をとる確率 \( P(j) \) を以下のように定義する。

\[
P(j) = (1 - w_e)P_r(j) + w_eP_e(j) \quad (1)
\]

ここで \( P_r(j) \) と \( P_e(j) \) はそれぞれチーム探索政策と最適チーム推定政策により決定されるj番目の戦略をとる確率である。

3.1 チーム探索政策

チーム探索政策は非常に単純で、最も経験の少ない戦略を選択する政策である。監督は一試合の中で何回か戦略を変更する機会を持つ。ここでは戦略を変更するまでの期間をビリオドと呼び、システムは \( n \) 個ある戦略のうちj番目の戦略をとった回数を保存した合計回数表をもつ。チーム探索政策により次のビリオドでj番目の戦略を選択する確率 \( p_r(j) \) を以下のように定義する。

\[
p_r(j) = \frac{p_r(j)}{\sum_{l=1}^{n} p_r(l)} \quad (2)
\]

ただし

\[
p_r(j) = \begin{cases} 
0 & (j \text{ 番目の戦略を既に行行った}) \\
1 & (T_r(j) = 0) \\
\frac{1}{T_r(j)} & (それ以外)
\end{cases} \quad (3)
\]

Figure 1 に式 (2) と (3) の考え方を示す。監督は第一ビリオドでは試合回数表から経験の少ない戦略を選択する。第二ビリオド以降では既にとった戦略以外の中から経験の少ない戦略を推定する。

3.2 最適チーム推定政策

最適チーム推定政策は相手チームの戦略を推定する手続きと推定した相手に勝てる戦略を推定する手続きからなる。j 番目の戦略を相手がとるj 番目の戦略を自チームがとったときの得失点差を記録する得失点表 \( T_e(k,j) \) をシステムが保持している。自チームが第iビリオドにj 番目の戦略をとった得失点 \( a_i^j \) を得たとき、システムは相手チームの戦略を以下のように推定する。

\[
P_e^*(k) = \frac{p_e^*(k)}{\sum_{l=0}^{n} p_e^*(l)} \quad (4)
\]

\[
p_e^*(k) = \max_i |a_i^j - T_e(l,j)| - |a_i^j - T_e(k,j)| \quad (5)
\]
(5) 式右边の第一項は推定の最大値を仮定、第二項は相手の戦略が$k$であると仮定したときの誤差であり、推定が正しければ小さな値を返す。したがって$P_s^i(k)$は相手の戦略が$k$番目であれば大きな値を返す関数になるので、これを式(4)で正規化することで相手の戦略が$k$番目である確率的な値から仮定を算出する。あるゲーム中に$i$ビリオド進過した場合、相手チームのとっている戦略は次式で推定する。

$$
P_s(k) = \frac{1}{m} \sum_{i=0}^{m} P_s^i(k)$$

次に監督が$k$番目の戦略をとったときに得られる得失点差の推定値$x_s(j)$を以下で計算する。

$$x_s(j) = \sum_{k=1}^{n} P_s(k)T_{i}(k,j)$$

これから最適チーム推定政策として$k$番目の戦略をとる確率を次のように定義する。

$$P_s(j) = \frac{P_s(j)}{\sum_l P_s(l)}$$

ここで

$$P_s(j) = \begin{cases} 0 & \text{(if } x_s(j) \leq 0) \\ x_s(j) & \text{(else)} \end{cases}$$

Figure 2に最適チーム推定政策の基本的な考え方を示す。チームの監督はまず得点差表から平均的に強いチームを割り出し、第一ビリオドに臨む。ここでは最大の平均値5を取っれたB戦略を取る。次にビリオドが来たら得点差表から相手チームの戦略を割りだす。直前のビリオドに取った戦略Bと得点差5から相手チームはA戦略を取っていると予想する。その戦略に勝てる戦略を推定し実行に移す。相手チームがA戦略をとっているものと仮定してB戦略をとる。

3.3 重み$w_{er}$の更新

一試合は$n_p$ビリオドにわかれており、第$i$ビリオドに$a_i$番目の戦略をとった時、チーム探索政策と最適チーム推定政策の割合を決定する重み$w_{er}$を以下のように更新する。

$$w_{er} \leftarrow w_{er} + \Delta w_{er}$$

$$\Delta w_{er} = \alpha \sum_{i=2}^{n_p} s^i \frac{i}{n_p} x_s(a_i)$$

ここで$\alpha$は更新率である。もし重み$w_{er}$が1より大きくなった場合は1に、0より小さくなった場合は0に変更する。この式の意図は重み$w_{er}$を学習初期はチーム探索政策を優先し、推定がうまくいくようなら最適チーム推定政策を優先させるように自動的に更新させることにある。式(11)は推定が正しければ$w_{er}$を増やす形になっている。$\frac{i}{n_p}$はデータが少ないときの予想よりも、数ビリオド戦った後のデータが豊富な時の予想を大きく考慮させるための項である。

4 実験
4.1 設定
ここでは以下の仮定をおく。

- 選手とフィールドはRoboCup2000の規定に則している。
- 各チームは2選手を揃える。
- 選手はそれぞれ個性を持った固定政策で動く。
- 各選手はコミュニケーション能力を持たない。
- 一試合は500試行である。
- 100試行を1ビリオドとする。つまり一試合中5回の戦略変更が可能である。
- 選手の初期配置は自陣の中でランダムであり、ボールは中央にかれる。

4.2 チーム戦略
ここではチーム戦略は監督の立場で出場させる選手の組合せを決することとする。4種類の選手を用意した。まずボールに対してブローチする制御器を二種類用意する。

ROUGH ボールへのブローチが速く、ハンドリングは荒い。

CAREFUL ボールへのブローチHmmというハンドリングともに慎重。

つきに見方を認識したときの振舞いとして次の二種類を用意する。

SELFISH 敵も味方の障害物としてのみ認識する。

SOCIAL 味方を識別し、味方に道を譲る。

この二つを組み合わせることで4種類の制御器がある。チーム戦略としては二選手の組合せで10種類ある。

4.3 実験結果

Figure 3: The change of $w_{er}$

—9—
Figure 4: Difference of obtained and lost goals

Table 1: The result of difference of obtained and lost goals by learning

<table>
<thead>
<tr>
<th>opponent</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.3</td>
<td>3.5</td>
<td>5.5</td>
<td>4.5</td>
<td>3.0</td>
<td>5.0</td>
<td>7.0</td>
<td>6.0</td>
<td>5.0</td>
<td>4.0</td>
<td>4.3</td>
</tr>
<tr>
<td>2</td>
<td>3.0</td>
<td>4.5</td>
<td>5.0</td>
<td>3.5</td>
<td>2.0</td>
<td>3.0</td>
<td>5.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.3</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>4</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>5</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>6</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>7</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>8</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>9</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>10</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
<tr>
<td>avg</td>
<td>3.0</td>
<td>4.0</td>
<td>5.0</td>
<td>3.0</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Table 1は学習が終わったと見なせる時点(50試合)で5ビリオド(5試合間)以上の試合が行われた組合せの結果だけを選んだ対戦成績である。ただし表中の値は1ビリオドでの得失点の平均である。探索範囲が上位チームの中に探索されている。このことにより絶対に強い領域を決定的に探索していることが分かる。絶対に強い領域を約1/3の時間でした。

Figure 5はTable1の結果を用いて作成したチーム間の相位の図である。サイクル状の関係に成っていることから、一番強いチームを選んで来るとそれを実行しきれれば良いことになることがわかる。下位のチームの相関関係が分かりなくても試合に勝てる上位のチームを獲得しているため、試合に勝つことができたアプローチは、また試合が長くても少ない情報(試合数)で敵チームに勝
Figure 6: A sequence of a failure recovery behavior among two robots

5 Conclusions

本論文ではエージェントがそれぞれの個性を持った固定政策をもち、エージェントの組合せを決定できるコーチの視点からチーム戦略を獲得する学習方法を提案した。2対2の試合におけるシミュレーションを通じて本手法の有効性を示した。

参考文献

PVM を用いたロボットのタスクの分散化に関する検討
A Study on Distribution of Tasks of Robots utilizing PVM

中野博史，鈴木秀智，長内真人，安井慎太郎，渡辺一郎，樋口次男
Hiroshi NAKANO, Hidetomo SUZUKI, Masahito OSANAI,
Shintaro YASUI, Ichiro Watanabe, Jiro KASHIO
三重大学工学部
Faculty of Engineering, Mie University
{nakano,suzuki,osanai,yasui,wata,kashio}@kashio.info.mie-u.ac.jp

Abstract

This paper describes the system configuration of our robots, and a method of distribution of tasks on robots with PVM. PVM is a tool to integrate many computers connected through network as a distributed parallel computer and execute parallel processing. PVM can handle the task running on robots with the same manner by use of the message passing mechanism. This makes easy to develop and maintain the robots’ programs. An experiment shows that the latency of multi-thread programming and multi-process programming is nearly negligible for controlling a robot.

1 はじめに

人工知能と知能ロボットに関する研究のひとつの標準問題として、ロボットによるサッカー競技 RoboCup が提唱され、種々の技術の統合が必要な複合的問題として注目されている。著者らの研究室では、これまで画像処理、ネットワーク通信技術およびその応用であるマルチメディア処理に関する研究を行ってきたが、これらの研究成果の応用およびその制御処理への展開のための題材として RoboCup を採用した。主な研究対象は、自律移動ロボットの視覚情報処理の精度向上、情報の共有による情報補完、行動計画の高度化、協調動作の実現であるので中型リーグを模倣した。これらの問題については既に多くの研究グループによって成果が出されているが、解決すべき問題はまだ多く残されていると考える。

本報告では、上記の問題を扱うために製作したロボットシステムの概要を紹介し、PVM (Parallel Virtual Machine) [Geist, 1994] を利用したロボットシステムの制御プログラムの開発について述べる。さらに、PVM を利用した場合の応答遅延は、ある一定条件下では大きな問題にならないことを実験により示す。

2 ハードウェア構成

ロボットは、大別すると制御部と移動機構部からなる (Fig. 1)。おおまかな構成は大阪大学 Trackies の自律移動ロボットを参考にさせていただいた。

2.1 制御部

ロボットの中枢であり、IBM AT 互換の CPU カード (PCI-371,JDS) を 1 個搭載している。CPU は Intel 社の Celeron(433MHz) であり、64MB のメモリを搭載している。2 個のシリアルポートは、ビデオカメラの制御および移動機構部との通信に使われている。パラレルポートは手動操作時の指示に用いている。Ethemet ポートを有し、無線 LAN 装置を介して複数ロボットで構成される LAN に接続されている (LAN Anywhere, Callus。or WLI-PCI-L11, Melco)。補助記憶装置として IDE 接続のハードディスク (16GB) を用いている。

最も負荷の重い画像処理は下記の画像処理ボードに担当させる、移動に関する詳細な制御は移動機構部に任せていている。これにより、CPU カードの負荷を軽減し浮動計算資源を環境の認識、他のロボットとの情報交換、およびそれらに基づく行動計画と実行 (移動機構部への指令を含む) に割り振ることが可能になった。

ビデオカメラ (EVI-D30, Sony) は、テレビ会議用の高機能のものであり、首振り機構、自動焦点機能、対象物追跡機能などが可能です。ただし、本研究では、対象物追跡機能は使用せず、独自の対象物探索および追跡機能を開発した。

画像処理ボード (IP5005BD, 日立) は、ビデオカメラから送られてくる映像を高速に処理するための専用ブロッサを搭載したものであり、映像から各種の対象物 (ボール、ゴール、ロボットなど) を認識するために情報処理を
いた画像処理手法を実行する。現在、1秒間で15フレーム程度の処理が可能である。
電源は密封型鉛蓄電池を用い、交直両用の電源装置 (NSP2-180-H2X, 日本プロテクタ) を介して必要な直流電圧を発生させている。

2.2 移動機構

ロボットの運動を担当する部分であり、既製の移動用台車 (TRIPTERS-mini, JSD) を用いている。2輪駆動1操舵輪方式の台車であり、両駆動輪および操舵輪にロータリーベンチレーションが付いていて、駆動輪の移動距離および操舵輪の回転角を知ることができると。これに加えて swing して、 northeast が搭載されているので、この台車のみで障害物を検知し、それに応じて行動することも可能である。また、RS-232C インターフェースをもっており、これを介して、ホスト側とプログラムとデータを送受信したり、搭載された OS への操作指示を受け取ったりすることができる。

本研究では、上記の仕様を活かして、移動に必要な制御機能 (基本動作、動作の達成度の検査、動作の中断、再開など) を移動機構部で実現し、制御部の移動制御を単純化した。なお、超音波センサの距離情報は不可視部分の認識や衝突回避に利用されている。

3 ソフトウェア構成

3.1 制御部

制御部で使用している主なソフトウェアを以下に示す。

- OS: Linux, Version 2.0.36 (Vine Linux 1.1)
- プログラミング言語: C++ (g++)
- その他の開発環境: GNU ソフトウェア (emacs, vi, gdb, etc.)
- 分散並列処理環境: PVM

制御部では以下のタスクを実時間応答が可能な速度で実行しなければならない (Fig. 2)。なお、ここでは、PVM の用語法に基づき、「タスク」は仮想並列コンピュータ上で動作しているプログラム単位 (プロセス) を表すものとする。

(1) LAN: 無線 LAN を介して外部と通信する。
カーサーバがCOGに対応し、エージェントがPLANに対応する考え、PLANのインターフェースをエージェントのそれに合わせることにより、シミュレーションシリーズのエージェントをロボットに容易に取り込むことができると。もちろん、エージェントが獲得した戦略などの知識をそのまま本ロボットに適用することは難しいので、この点に関する詳細な検討が必要である。

3.2 移動機構

移動用台車はMS-DOSの縮小版に相当するJS-DOSと呼ばれるOSが搭載されている。したがって、ソフトウェア開発に必要な、MS-DOSを利用したものとほぼ同じである。具体的には、MS-DOSの実行形式プログラムを作成し、それを台車のCPUボードに搭載されたEEPROMにロードして実行する。

本研究で作成した制御プログラムは、制御部のTRANSFERタスクから受け取った行動指令の解読と実行、および、超音波センサの距離情報の取得とTRANSFERタスクへの通知を行う。TRANSFERタスクから受け取る行動指令は、距離や速度の指定が可能な前進後進や旋回に関する指令である。それらの指令を実行するときは、移動距離や速度の検出、それにに基づく目標達成のための制御、モータ負荷時の対応など複雑な処理が必要である。このような処理を行うために、台車に搭載されていた制御用ライブラリを本目的に合うように変更し、それを利用した。

4 PVMを用いたタスクの分散化

4.1 PVM（Parallel Virtual Machine）

PVMは、メッセージパスシング機構を用いた移植性の高いプログラミングシステムであり、これを用いると複数のコンピュータをひとつの仮想並列コンピュータとして扱うことができる。この仮想並列コンピュータを構成するコンピュータは互いに離れたところに設置された異種コンピュータであってもよい。PVMアプリケーションを構成する各タスクは、C、C++、Fortranなどで書かれる。現在、PC、ワークステーション、マルチプロセッサ、スーパーコンピュータなど数多くのコンピュータで利用できる。PVMはいくつかの機関で共同で研究開発が進められていて、GPLライセンスに基づくフリーソフトとして公開されている。たとえば、Netlib [Netlib, http]からもダウンロードできる (日本国内にミラーサイトあり)。

ネットワーク接続されたコンピュータを用いて分散並列処理をするときは、ソケットライブラリによるネットワーク通信を行う必要があるが、接続するマシンの管理、負荷の分散、通信制御などネットワークを意識した処理が必要になり、これが分散並列処理プログラムを作成するときの障害になっている。PVMは、ネットワーク通信およびselectシステムコールによる非同期通信（通信の多重化）の複雑な処理を受け持ってくれるので、PVMを利用した分散並列処理プログラムの作成は容易である。

PVMの構成をFig.4に示す。仮想コンピュータを構成するホスト（コンピュータ、プロセッサ）にpvmドアーモンが起動され、これらが相互通信することによって仮想並列コンピュータが構築される。タスク（並列動作するプログラム）間の通信はpvmドアーモンを通じてTCPソケットを通じて行われる。pvmドアーモンの通信はUDPソケットによって行われる。

ユーザは、まず、マスタープログラムを起動し、並列動作させるプログラムをスレーブプログラムとして仮想並列コンピュータ上で複数起動することによって並列処理を実行する。タスク間の通信は、タスク固有のIDおよびメッセージIDをメッセージに添付して行う。各ホストの物理的な配置および通信方法はPVMによって行われるので、ユーザは各ホストについて考慮する必要はない。もちろん、ホストを指定した配置も可能です。通信方法には、同期（ブロック）方式と非同期（ノンブロック）方式があり、必要に応じて使い分けることができる。

PVMはネットワーク接続された異種コンピュータ間のデータ交換を扱うので、ネットワークを介したデータ交換の問題は、データ表現の変換の問題など余分な処理時間を要する。したがって、単一プロセッサ内で実行したり、同・アーキテクチャのコンピュータ間で通信したりするのに比べて実行速度が低下するという欠点がある。PVMにはこの問題を軽減するためにメッセージパッファによ
る通信制御が採用されている。この種の流れがロボットの制御にどの程度影響するか重要な問題であるので、後節で検討する。

4.2 他の並列処理用ソフトウェアとの比較

並列処理を実現するソフトウェアは多く存在し、専用言語およびライブラリに大別される。

4.2.1 専用言語

専用言語としては、Concurrent {C, C++, Pascal, Euclid}、Modula-{2,3}、Adaなどのように、並列処理に必要な基本機能であるプロセスの生成や削除などを文法仕様としてもつものがある。プロセスのプロッサへの配分などは言語処理系が行うので、ユーザはこの作業をしなくてもよいが、処理の分割や並列化については指定する必要がある。

スーパーコンピュータなどでよく利用されるHPF (High Performance Fortran) [HPF, http] はFortranをベースにして並列化のための指示行を追加したものである (HPC++) といわれる C++をベースにしたものもある [HPFPC, http]。この言語処理系は MPIなどを利用した SPMD (Single Program Multiple Data Stream) 形式のプログラムに変換するものが多い。ユーザはプログラム内で並列化したい部分を指示行で指定するだけでもよく、プログラムのSPMD 化、通信管理などは処理系によって実現されるという特長をもつ。また、共有メモリモデル用として標準化されたコンパイラ指示行ベースのAPIとしてOpenMP [OpenMP, http] がある。

4.2.2 ライブラリ

MPI (Message-Passing Interface) はMPIフォーラム [MPI, http] という任意参加の会議で策定された仕様であり、実際のソフトウェアを指すものではないが、この仕様に準拠したソフトウェアやライブラリを指すのに使われることが多い。MPIの実現 [MPI, http] は数多くあり、並列マシンマイナーが提供するものや、MPICH、MP-MPI、LAM/MPIなどのフリーのものがある。MPIはPVMより後に仕様策定されたものであり、PVMなどの主要なメッセージパッキングライブラリの開発者や並列計 算機ベンダーがこのフォーラムに参加しているので事実上の標準になっている。しかし、元々は、MPP (Massively Parallel Processor) に装着されていたため、同種プロセッサ向けであり、プログラムの実行に使用するプロセッサ数やプロセッサ配置が実行開始時に決めなければならず、SPMD形式しか使えない、という制約があった。現在の仕様であるMPI-2では、これらの制限は緩和されているが、実際のMPIソフトウェアが完全に対応しているとは言えない。MPIが標準仕様になりつつあるので、実装の力が安定すればPVMから移行することを考えている。基本的には同じ方式の並列化ライブラリなので移行はそれほど難しくない。

マルチスレッドとは、ひとつのプロセス内に複数の処理の流れ（スレッド）が生成され実行されているものである。この方式では、各スレッドはデータを共有し、コンテクストスイッチにおけるオーバヘッドが非常に少ない、などの特徴をもつので、計算機資源にあまり負荷をかけずに並列処理を実現できる。ただし、共有データの排他制御はユーザの責任となる。この方式を実現するライブラリとしては、POSIX (Portable Operating System Interface) に準拠したpthread、LinuxThread [LinuxThread, http] などがある。

その他、ソケットライブラリ、selectなどのI/O同期ライブラリ、共有メモリライブラリを用いた従来の並列処理プログラム作成法がある。周知のように、この方法では、並列処理に関する複雑な処理（プログラムの分割、通信管理など）をユーザが行わなければならない、ソフトウェア開発の効率は低いとは言えない。

上記の言語およびライブラリはそれぞれに特性があるので、いずれがロボット制御プログラムの分散並列化に向いているかは一概に言えない。ただ、ソフトウェア開発の効率化、拡張性、などを考慮すると、柔軟な対応や拡張が可能なライブラリの方が好ましいと考えられる。これは、本研究でPVMを使用した理由のひとつである。

4.3 応答の遅延

ロボットを制御するプログラムを作成する上で最も重要な問題は、環境の認識結果に対する行動の遅延である。この遅延時間は前節で紹介したプログラミング手法に大きく依存する。ロボットを制御するプログラムを並列動作可能な部分に分割して、それぞれを並列動作できるようにプログラムする場合、(a) OSによるプロセスまたはタスクのスケジューリング、(b) データ通信の速度、(c) プロセスまたはタスクのコンテクストスイッチのオーバヘッドなどが実時間応答性に影響すると考えられる。(a)については、リアルタイムOSを使用するのが確実であるが、ひとつのプロセッサの負荷を抑えることで実用範囲に収めることが可能である。(b)にはプロセス間通信とネットワーク通信があるが、前者はOSに依存し、後者はネットワークの通信速度や込み具合に依存するので、ユーザ側で対処することは難しい。(c)についてはマルチスレッドプログラミングを用いるのが好ましい。

森ら [森, 2000] の研究など、最近のロボット制御プログラムの多くはマルチスレッドプログラミングを用いている。確かに、マルチスレッドプログラミングは軽量な並列処理プログラムを作成するのに向いている。しかし、プログラムが大規模になったときの共有データの排他制御の問題、および、外部プロセスとの通信のためのプログラ
ラミングの煩雑さが、制御プログラムの拡張や保守において問題となる。

共有メモリなどを用いたマルチプロセスプログラミングでは、コンテクストスイッチのオーバーヘッド、プロセス間通信制御のためのプログラミングの煩雑さなどが問題になる。

一方、PVM (または MPI) を用いた並列処理プログラマーの場合は、コンテクストスイッチやライブラリ独自の処理によるオーバーヘッドが大きくなるが、プログラミングの効率は高くなると考えられる。なぜなら、プロセス（タスク）の生成と削除、プロセス間通信、各種併列制御について、ユーザ側では最低限の指示ですむように設計されていて、プロセッサ内とプロセッサ間の通信を同じ形式で行えるからである。このことは、ロボット制御プログラムの開発の効率を高めるのに役立つ。

以上の考察において、(c) がプログラミング手法によってどの程度影響するのかを実験で確かめる。

自律移動ロボットを用いて、Fig. 5 に示すタスクを以下のような方法で実行したときの処理時間を計測した。なお、今回は、プロセッサ間の通信を考慮した実験を行っていない。これは、プロセッサ間の通信による遅延はプログラミング手法による遅延に影響しないと考えたためである。

(1) 1 個のプロセッサ内、マルチスレッド処理
(2) 1 個のプロセッサ内、共有メモリとシグナルを用いたマルチプロセス処理
(3) 1 個のプロセッサ内、PVM を用いたマルチプロセス処理

Fig. 5 において、IP は IP5005BD を用いて 3 種類のカラー画像のしきい値処理を行い、その結果を、(1) の場合
は共通変数に、(2) の場合は共有メモリ SHM に、(3) の場合は IP の私的変数に書き込むという処理を繰り返す。

COG は IP の画像処理の終了をシグナルによって知って、認識結果（実際には適当な値）を共有メモリに書き込む。次の画像処理終了のシグナルを待つという処理を繰り返す。PLAN は COG の認識結果更新をシグナルによって知って計画および実行指針を行う（実際には適当な値を共有メモリに書き込む）、という処理を繰り返す。すなわち、IP が画像処理を反復し、その終了通知によって COG と PLAN が逐次的に動作しているという非常によい単純なマルチプロセス環境である。

上記のプロセス構成で認識・計画・実行というループを 10000 回繰り返し、その平均時間を求める結果、いずれの場合もほぼ 67ms となった。これは、COG および PLAN の処理時間が IP での処理時間を約 66ms に比べて非常に短いためである。また、平均時間が IP の処理時間におお
等しかったのは、画像処理の時間に比べてコンテクストスイッチなどのオーバーヘッドが無視できる程度短かったことを表している。今回の実験では各プロセスのメモリ所要量が
小さないのでスワップアウトが起きなかったと考えられる。

当然ながら、これが起きるような状況ではコンテクストスイッチによるオーバーヘッドの影響は大きくなる。

このことから、今回の実験の環境では、マルチスレッドプログラミングとマルチプロセスプログラミングと、遅延の問題に有意な差がないことが知られた。ただし、プロセス数が多くなった場合、および、各プロセスのメモリ使用量や計算量が大きくなったときの状況をこの実験結果のみから予測することはできないので、さらなる検討が必要である。

なお、(1) から (3) のいずれの場合でも反復の初回はプロセスの起動などのために 2 秒間の時間を要した。マルチスレッドプログラミングの方が 1 割程度時間が短かったが、これは初回のみであるので、ロボット制御においては実質的な問題ではない。

5 まとめ

本報告では、著者らの研究室で製作した自律移動ロボットの構成を紹介し、PVM を用いた分散並列動作の可能な制御用プログラムの作成について述べた。PVM を用いたプログラムはマルチスレッドプログラミングによるものに比べると、コンピュータへの負荷が大きく、応答時間の遅れが懸念されたが、今回の実験では有意な差は出なかった。PVM を用いると、メッセージングを含むの

オブジェクト指向的プログラミング手法を利用して、ソフトウェアの開発効率を高めることができる。したがって、自律移動ロボットの制御プログラムを作成するツールとして PVM は非常に有望であると考える。

この研究により、自律移動ロボットの協調動作を含め
参考文献


[HPCPP, http] High-Performance C++ : http://www.extreme.indiana.edu/hpc++/


KU-Boxes2001における高速色抽出処理

Fast Color Image Segmentation in Team KU-Boxes2001

田中一基、影山 茂、五十嵐治一、黒瀬能幸
Kazumoto Tanaka, Shigeru Kageyama, Harukazu Igarashi, Yoshinobu Kurose
近畿大学工学部（広島県東広島市）
School of Engineering, Kinki University, Higashi-Hiroshima

Abstract
It is important to process color image in real time for robot systems such as soccer robot systems for RoboCup. In this paper, we describe a method for fast color image segmentation used in our robot system. Furthermore, we propose a color extraction method where color-matching regions on HS-space are approximated by rectangle pieces on UV-space. Using the method, it is possible fast and robust color extraction without transforming UV to HS.

1 はじめに
自律移動型ロボットシステムの評価の場として、ロボットサッカーの競技会 RoboCup[1]があり、実機小型部門において、我々は共通プラットフォーム開発のプロジェクト JPS-II を進めてきた[2]。しかし、これまでの評価の結果、我々のシステムは、性能面におけるいくつかの問題が明らかになってい
[2]。本稿では、その中でも、画像処理速度の問題を取り上げる。

本システム（グローバルビジョン方式）の画像処理は、ロボットに取付けられたカラーマークやカラーボール等の色抽出処理が主である[3]。この処理に対して、我々は、1 コマのカラー画像あたり、33ms（ビデオレート）以下で処理することを目指している。しかし、これまでの処理方式では、等色領域ごとに抽出処理を繰り返していたため、1 コマのカラー画像の処理に、約 100ms の時間を要していた[4]

高速な色抽出処理の方法としては、これまでに、色の属性値を添字とする配列データの各ピットを、等色領域を示すフラグに見立てることによって、複数の等色領域との照合を同定に行う方式が、カーネギーメロン大学のチームにより提案されている[5]。そこで、まずこの方法を、現行のロボットシステム KU-Boxes2001 の画像処理システムで用いている画像処理ボード（日立 IP505）上で、実現することを試みた。2 で、この実装方法について、概略を述べる。

次に、我々は前述の配列を 2 次元に拡張し、YUV 画像において、高速かつ傾斜な色抽出を可能とする処理方式を考察した。KU-Boxes2001 の画像処理では、YUV の各値処理を用いて色抽出処理を行っているが、それよりも傾斜な色抽出方法として、色相、彩度、輝度に関する属性を用いる方法がある[6]。照明環境の変動等の問題に対しては、これらの属性を利用することが望ましい。しかし、属性 U, V から、色相、彩度に関する属性への変換処理が必要であるため、高速色抽出に対しては処理時間の点で問題がある。そこで、UV を変換することなく、色相、彩度、輝度で設定される等色領域を、近似的に設定する方法を提案する。3 で、この方式の詳細と、実験結果とを示す。

2 現行の色認識処理

KU-Boxes2001における色認識処理について、処理手順(1)~(4)に従って概略を述べる。

(1) YUV 濃度変換
画像処理ボードに取り込んだ YUV の各値を、6 種類の等色領域を表現可能な 8 ピットデータに変換する。濃度変換には、YUV の各値から設定される変換テーブルを用いる。この変換テーブルについて、以下に述べる。

等色領域を設定する際の範囲内の YUV 値について、各色に対応するビットに 1 を置き、その他のビットには 0 を置く。図 1 に変換テーブルの例を示す。同図では、変換後の濃度データの下位ビットから順に、色 A~H を対応させている。たとえば、色 A の Y, U, V の各値が、

(Ya_min, Ya_max) = (1, 2)
\[(U_{a\ min}, \ U_{a\ max}) = (2, 4)\]
\[(V_{a\ min}, \ V_{a\ max}) = (3, 4)\]
の場合、図で示す位置に 1 を置く。

<table>
<thead>
<tr>
<th>変換前</th>
<th>変換後</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Y 変換テーブル</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1  ⇒ 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>2  ⇒ 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>3  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>4  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>255 ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U 変換テーブル</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1  ⇒ 0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>2  ⇒ 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>3  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>4  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>255 ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V 変換テーブル</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1  ⇒ 0 0 0 0 0 0 1 0</td>
</tr>
<tr>
<td>2  ⇒ 0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>3  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>4  ⇒ 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>255 ⇒ 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

図 1 YUV 濃度変換テーブルの例

(2) 画像間 AND 处理
(1) で変動変換された Y, U, V の各々に対し、画像間の AND 处理を行う。各画素において、この処理を行うことにより、等色領域を示す数値、あるいは 0 からなる画像を得る。1 画素あたりで必要な AND 处理は 2 回である。処理後の画像の例を図 2 に示す。同図において 1, 2, 4, 8 の各値は、図 1 の変換テーブルの設定から、それぞれ、色 A, B, C, D を表している。

図 1 に示した濃度変換テーブルの例では、ある画素の濃度が Y=1, U=2, V=3 であった場合、それぞれ 3, 13, 17 に変換され、AND 处理によって 1, すなわち色 A を示す数値を得られる。

図 2 AND 处理後の画像の例

(3) 2 値画像処理
AND 处理後の画像を閾値 1 で 2 値化し、ラビリングおよび 2 値連結成分の属性値（重心座標等）の計算を行う。

(4) 色情報取得
連結成分の色情報を、AND 处理後の画像から得る。このため、(3) で導出された重心座標の各々を中心とした小ウィンドウ内（たとえば 3×3）の最大濃度値を抽出する。この処理は、対象物の像が凸集合であるため有効である。

以上の色認識処理は、IPS005 に用意されているライブラリ関数を、そのまま利用することができる。現在、カラー画像 1 コマ (Y: 512×512, U: 256×512, V: 256×512) あたり、ロボットの ID 識別等の後処理を含め、約 30ms で行うことが可能であった。したがって、KU-Boxes2001 では、ビデオレートによる画像処理が可能である。

3 2 次元配列要素の参照による色抽出方式
照明環境の変動等に対して順健な色抽出のために、図 3 に示すような、色相、彩度、輝度に関する関数を用いる方法が知られている。これらの関数処理において、2 で述べた方法を適用する場合には、属性 UV を色相、彩度に関する属性に変換した上で、これらの濃度変換テーブルを利用する方法が考えられる。しかし、IPS005 の場合、このような文字列変換には、U および V の 1 面 (512×512) あたり約 9ms の処理時間が必要である[6]。したがって、2 で述べた色認識処理時間を加えると、1 コマあたり 39ms の処理時間が必要となり、ビデオレートで処理するには、処理時間の点で問題が生じる。

そこで、属性 U, V を変換することなく、色相、彩度に関する関数処理方式と同程度に順健な抽出方式を考案した。この方式は、U および V 値
と、等色領域をUV平面へ投影した領域（以後、UV等色領域）との、対応情報を持つ配列UVと、UV等色領域およびY値と、等色領域との、対応情報を持つ配列UYとの、2つの2次元配列を用いる。配列UVにより、UV平面上の任意の座標を、UV等色領域として指定することができる。すなわち、図4に示すように、色相彩度に関して設定される領域は、正方形ブロックの集合で近似される。したがって、UV平面で、UVの最大/最小値による矩形領域や、色相彩度の推定値による矩形領域といった、比較的単純な等色領域の設定だけではなく、複雑な形状の領域設定も可能である。

また、配列要素の参照により、AND処理を行うことなく、複数の等色領域との同時照合が可能である。このため、2で述べた方式よりも高速な処理が期待できる。

図3 色相彩度輝度で設定される等色領域の例

図4 本方式で設定可能な等色領域の例

配列UY、UYを考える。UYの添字に、UVの値をそれぞれ対応させる。UV等色領域の各々に番号を設定し、UVの対応する要素に、この番号を持たせる。ここで、Y値の違いにより、複数の等色領域が共通のUV等色領域を持つ場合には、共通領域に対して、新たに番号を設定する。

次にUYの添字に、UV等色領域の番号、Y値をそれぞれ対応させる。等色領域の各々に番号を設定し、UYの対応する要素に、この番号を持たせる。

これらの配列を用いれば、色抽出は、UY(UV値、Y値)を参照することで可能となる。たとえば、ある画素のXYUVの濃度値がそれぞれ1,2,3であった場合（図5）、UY(UV(2,3,1))を参照することにより、対応する等色領域を示す番号1が得られる。

図5 配列要素参照による色抽出の例

3.1 本方式の詳細

まず、図5に例示するような、2種類の2次元

3.2 色抽出の実験

本方式と従来のYUV関値処理との、色抽出の
3.2.2 結果

閾値設定時よりも照明を暗くした場合と、明るくした場合の実験結果を、それぞれ図8、図9に示す。なお、抽出処理後の画像は、観察のため2値化している。

図8、図9のいずれも、従来のY.U.V閾値処理(b)では、カラーボールと背景が重複して抽出され、カラーボールの切り出しは困難である。これは、青紙の等色領域と、Y.U.V閾値処理で用いた閾値の範囲とが一部、重なっているためと考えられる(図6)。これに対し、本方式による処理(c)では、いずれの照明状態においても、青紙の影響を抑え、比較的安定してカラーボールの切り出すが可能であることがわかる。

なお、本方式の処理を実行可能なライブラリ関数がIPS005には用意されていないため、IPS005で取得した画像をホストコンピュータ（PentiumⅢ600MHz）のメモリに転送し、ホストCPUで本方式の処理を行った。Y.U.V値から等色領域番号への変換処理時間は、1コマのカラー画像あたり約5msであり（メモリへの転送時間は含まれず）、色抽出においても、十分に高速であると考えている。

図6 実験対象の計測値とUV等色領域

図7 実験対象の模式図

図8 閾値設定時よりも暗くした場合の抽出結果
色認識処理の概要を述べた後、頑健な抽出を行うための高速色抽出方式を提案した。本方法は、UV等々領域あるいは等色領域を表現する色抽出の抽出を特徴とする。本方式は、色相、彩度などの調整を行うことなく、頑健な抽出が可能となる。また、実験により、UVの栃木の最大値を用いた従来のY.U.V閾値処理と比べ、明暗変動に対する頑健性を確認した。また、ホストコンピュータ上で処理した場合、1コマのカラー画像あたり約5msの処理時間で色抽出を行うことを確認できた。

今後は、本方式を画像処理ボード上に実装した上で、安全な画像処理システムを検討した後、色抽出処理の安全性を含めた処理時間の評価を行う予定である。

なお、本研究の一部は、日本学術振興会より科学研究費補助金（C2、課題番号11680405）の助成を受けた。

参考文献
[2] 五十嵐治一、小末将吾、田中一基、黒瀬篤史、五百井正：“IPS-IIプロジェクトの総括と今後の展開”、人工知能学会研究会資料 SIG-HOT RoboCup Spring Camp 2000, pp.1-5, 2000
[3] 田中一基、朝倉啓、小末将吾、五十嵐治一、黒瀬篤史：“RoboCup小型部門用共通ロボットシステムにおける画像処理サーバ・IPS-IIプロジェクト”、人工知能学会研究会資料 SIG-Challenge-9804, pp.1-4, 1999
RoboCupシミュレーションリーグ人間参戦システム
OZ-RP の提案

OZ-RP system: RoboCup Simulation League Human Interface to beat replicants

秋田純一, 西野順二, 久保長徳, 下緒弘樹, 藤塚到
Junichi Akita, Junji Nishino, Takenori Kubo, Hiroki Shimora, Itaru Fujitsuka

OZ-RP Project
akita@fun.ac.jp, nishino@se.uec.ac.jp

abstract

This paper introduces the OZ-RP system architecture, which allows human players to join the RoboCup simulation league matches. Final goal of the RoboCup research project is to construct an artificial team that can beat a human soccer team in fair-play status. OZ-RP system is toward this goal in the simulation league. We show and discuss constraints and system configuration of regulation legal human interface systems in the league.

1 はじめに

本稿の目的は、ロボカップシミュレーションリーグにおいて人間11人がリーグの制約に合致しつつ参加するためのモデルとして、OZ-RPシステムを提案することである。

ロボカップシミュレーションリーグは、人間と対等にサッカーすることのできることを目的にした人工知能とロボティクスの研究課題RoboCupのうち、サッカーサーバシミュレーションの上で行なわれるものである。協調エージェントとしてのプレイヤープログラムで作成し、シミュレーションして対戦する。

シミュレーションリーグは他の実際リーグに比べて戦術、戦略面での高度化が進み、2000年現在で上位チームの協調的行動のレベルは、人間チームが実際のフィールドで行なっているゲームに匹敵している。これはシミュレーションリーグがハードウェアトラブルなどを回避することができ、また、過去の資産が比較的容易に継承できることに起因している。

ロボカッププロジェクトの最終目的は、人間チームと人工チームとの対戦という安全性なども考慮しなければならないきびしい制約下で、人工チームが勝利できるための原理と技術の開発にある[3]。そのための基礎技術の育成のためのシミュレーションリーグの環境でも、人間と人工プレイヤーとがサッカーを対戦することは大きな意義がある。

OZ-RP (OZ by Real Players) チーム[6]は、人間11人による実プレイチームである。このチームと人工プレイチームの対戦を行なう。このとき、人間のチームも人工プレイチームと全く同じ制約にのり、そのうえで勝つべく全力を持って戦うことが、人間の持ち協調能力を計測し、他の人工チームとの比較を行う上で重要である。

このチームで人間プレイヤとシミュレーションフィールドを繋ぐためにOZ-RPシステムを提案する。基本機能は、サッカーサーバからの情報を人間プレイヤ操作者（以下パイロットと呼ぶ）に提示し、パイロットの決定した行動をサーバーサーバで規定されたプロトコルに乗せ、送信するというものである。さらに、パイロットの操作は製造の判断を活かし、可能な限り強いチームとなるようなユーザインタフェースの工夫が必要である。提示する情報を分かりやすくマップしなおして情報補完を行う、状況に応じた
視点位置変更などの必要もある。またパイロットの戦略的な意図を投入として、それを低レベルのプログラムまで自動的に変換するアシストシステムも必要となる。

本稿では、本プロジェクトとシステムの持つ意義と目的を述べ、システムに求められる制約を明らかにした上で、システム構成の提案と実現化の検討を行う。

2 OZ-RP プロジェクト

OZ-RP (Oz by Real Players) チームは、11人の人間が操縦するインタラクティブプレイヤによるチームである。OZ-RP プロジェクトとしてシステム開発とチーム編成を行っている。

2.1 プロジェクトの意義と目的

OZ-RP プロジェクトの目的は、人間 11 人が操作するプレイヤによってシミュレーションリーグに参戦し、他の人工チームと対戦することにある。

これは RoboCup が掲げている、人間チームとロボットチームの戦いでロボットが勝つ、という最終目標にむけての一つのアプローチである。これまで、シミュレーションリーグでは、人工チームだけが開発、実験されてきた。しかしながら人間と人工チームとの対戦を最終目標とするならば、シミュレーションリーグにおいても人間との対戦を行うことが有意義であり、かつ必要である。

実機各部門と比較すると、現在のシミュレーションリーグの人工チームの開発では、基本的なスキルの実験はほぼ完成しており、チームとしての協調的行動の実現が主テーマとして扱われるようになっている。試合でのプレーも人間のチームのように高度な協調が発現しており、またその完成度がチームの強度と密接に結びついている。これらのチームと人間チームが対戦することで、人間の持つ柔軟な判断力およびプレイや実験で協調するための見識を鍛え、それぞれの知的能力を発揮できるかを調べることができる。

そこで人工プレイヤと同じ条件を人間同僚パイロットに対しても課し、上手に協調できるかどうか、その原理はなにかを探る。

このため、使いやすいユーザインタフェースと、人間の高度な判断を活かすための情報統合や行動生成などの、半自動プレイサポートシステムを構築することが目標である。

2.2 課題と方法

処理の面での課題は、プレイの視点に立ったときの情報補完と、人間の高度な戦略判断を活かす行動の生成である。国際大会などでは、他の人工チームにできるだけ近い制約を負うことが必要であり、これらの制約を負っていることを示すことができるシステムとなければならない。また大会では、フェアプレーの観点から他の人工チームと比較して、極端に多くのホストを要求したり、特別なホストを多数接続したりすることもできない。

複数台 1 台の入出力あるいは出力装置を共有する、つまり一つのディスプレイを二人以上で共有するような形態では、他のプレイヤの視覚情報や行動などが間接的に得られることになり、人工プレイヤには実現不可能な情報共有であり、不公平であると同時に本来の主旨から離れててしまう。このため、11人のプレイヤをプレイヤとしてサッカーリーグに接続するには、11 台の出力装置を必要とする。

11 台の出力装置を、より少ないホストを介して接続するため、アプリケーションリレーラのマルチプレイヤが必要であり、その構築は不可欠である。これにより 1 台が複数のホストを介してマルチプレイヤであり、なおかつ複数の入出力装置のデータを区別できるようなサバシステムも必要である。

2.3 現状

プロジェクトは公募型で行なわれ、システムを作るエンジニアと、プレイを行うパイロットとを合わせ 2001 年 3 月末現在で 15 名が参加している。パイロットだけの応募も認められた背景では、システムを作成作業に因わせ、プレイヤとして繰り返し練習をし、最大の能力を発揮してもらうことを期待している。メンバーは日本各地に分散しており、集まって密な相談のうえでシステム構築をすることが必要である。
OZ-RP としてのプロトタイプシステム開発は4〜6名が主として行なっており、スライドリストと専用のWWWページを介して相互に情報交換を行なっている。パイロットや直接にプロトタイプを開発していないエンジニアも、各自のアイディアを提案し検討している。

3 OZ-RP テクノロジーの提案

3.1 概念設計としての OZ-RP システム

OZ-RP システムとは、1つの実体としてのシステムを指すものではなく、ロボカップシミュレーションの制約に合致し、人間の能力を引き出すためのユーザーインターフェース構築の枠組である。

パイロットは、各自の好みを代表とする個々の特性を持っている。どこに直接の表示および入力のインターフェースでは、全員が同一のシステムを使用するので、個々の好みに合わせて選択できるようにした方が、習熟度も上がりやすい。

このため、システムとして1つの実体は作らず、人間参加における制約を満たすための概念設計のみとした。

3.2 システムの目的

OZRP システムは、OZRP プロジェクトのためのユーザインターフェースの概念設計である。このため、分散的に開発されるサブシステム群が統合システム事例を、以下の目的のために構築し示すことがシステムの目的である。

まず、パイロットが各自の好みや特性に応じて出力、出力、出力の各サブシステムの種類を選んで、自由に組み合わせることを目指している。さらに、システム設計上の制約として、ハードウェア接続の複合化（マルチプレキシング）が必要であり、これに対応できるような構造を示すことも目的となっている。

3.3 システム設計上の制約

シミュレーショングリーフレギュレーションに従うことが、そのままシステム設計上の制約となる。

項目としてまとめると以下のようになる。

1. プレイヤ間のオフライン情報交換の禁止
2. 可搬性とセットアップ所要時間の最小化
3. ホストの特権ユーザーや管理上の特別の措置の最小化
4. サッカーサーバプロトコル規格による仮想プレイ

項目1は、レギュレーション上重要な制約である。人工プレイはサーバを必要としないプロセス間での通信が禁止されている。人間の場合は音声やジェスチャーを含め多様なモードでのコミュニケーションができるため、システムとしてできるかぎりアンフェアな情報の共有・入手を排除する必要がある。たとえば、ディスプレイを共有することとは、本来他のプレイ者の視点でしか見えない情報を目にすることができることになり好ましくない。このため、パイロット毎の情報表示スクリーンが必要となる。

項目2と3は、大会の運営上の制約である。さらに、ロボカップの趣味である制約された計算能力での可用システムの実装から、他の人工チームと同じホストのみで処理する必要がある。これにより、パイロット11人の操作端末のために、11台のホストを要求しないことを前提にシステム設計する必要がある。

項目4は、サーバ側での変更・調整などを一切必要としないという意味である。これにより他の人工チーム同様サーバでの試合ログの保存などが可能となる。

3.4 OZRP システムの概要

以上の目的と制約のもとで、図1に示す2次元のサブシステムによる構造を提案する。このシステム構成により、サブシステムの連絡部で多重化（マルチプレキシング）や分散化ができる。また、表示システムと入力システムのインタフェース規格をそれに定義することで、パイロットの好みに適したものを選ぶことが出来るように配慮している。同様にアシスタントシステムもさまざまな交換可能とした。

とくに多重化できる部分を増やしたことで、使用する実デバイスやネットワーク構成によって、11台

分の多重化の実現方法も増加している。

3.5 SSCP

サッカーサーバに対しプレイヤとして直接接続する部分が、SSCP(サッカーサーバコミュニケーションプロセッサ)である。サッカーサーバプロトコル(SSPプレイヤ)では一つのUDP双向向ソケットで、情報の送信と行動命令送信を行なう。いっぱいOZRPシステム内では、情報表示と行動操作が自然的で分離されているため、SSCPがこれらの分離、統合を行なう。さらに内部にはタイマを持ち、サーバの100msec刻みに周期して送信制御を行なう。

3.6 プレイヤループ・アシストシステム

プレイヤループは通常の人工プレイヤと同様構造を持っている。すなわち、ノイズの含まれたサーバからの視覚などの情報統合、ワールドモデルの保持と更新、ボール追尾やドリブル、目的地への移動などの自動動作を行なう。バイロットにたいしては、統合した情報を提示し、バイロットからの戦略的なレベルでの行動命令を解釈し、なおかつ自動で動作を行なう。

最低限のシステムとしては、SEEメッセージで受け取った情報をそのまま提示し、DASHなどのSSプレイヤでのコマンド発行を人間が行なうことも含まれている。しかしながら、直接発行では操作が悪く人工プレイヤに匹敵するプレイは不可能である。

個々の人間の情報把握能力や行動速度は、人工プレイのそれらとは比較にならないほど速い。いっぱい人間は、十分な時間があれば有限された情報から協調のために必要な情報を取り出し、高度な戦略を立て動作決定することができる。人工プレイに匹敵した試合を行うためには、この人間の高度な判断能力の有効性を引き出すプレイループが必要である。

3.7 バッド

ユーザが触れる入力装置をバッドと呼ぶことにする。バッドの接続としては、キーボード、マウス、各種ゲーム機器バッドなどが挙げられる。

キーボードとマウスは各ホストに1台ずつ標準で装備されているものであり、単体プレイに対しては特別のハードウェアを必要とせず理想的である。しかし、11人分を用意するにはホストを11台占有する必要がある。

各種ゲームバッドは、一般に操作性に優れ使用感覚の評価も高い。ホストとの接続にはなんらかの特別なハードウェアが必要となる。このとき、接続ハードウェアにマルチプレイ機能を同時に組み込みれば、1台のホストに対し複数のバッドを接続できるようになる。

3.8 スクリーン

バイロットへの情報提示出力機器をスクリーンと呼ぶことにする。接続として、コンピュータディスプレイ(CRT、液晶、プロジェクターを含む)、携帯ゲーム機の液晶ディスプレイが挙げられる。

コンピュータディスプレイは、通常ホスト1台につき1台の接続となるため、共有をしないという制約から従うとホストが1台必要となる。

液晶ディスプレイを持つ携帯ゲーム機では、11台
用意できれば11面のスクリーンを準備することになる。この場合、11台接続のためのマルチプレクサと、ホストとの物理的な接続に特別なハードウェアが必要となる。この接続ハードウェアにマルチプレクサ機能を同時に組み込めば、1台のホストだけでも複数のスクリーンを接続できるようになる。

3.9 Visual control バイパス

図1のシステムで、SSUIからSSDWへのバイパスを設けている。これは、カーソルポイントイングや視点変更などを実現するときに、パッドからのスクリーンの制御を行うためのものである。このバイパスを通じてユーザインタフェース部での独立したループを構成している。

カーソルポイントイングなどは、アシストおよび情報統合操作とは無関係である。システムのパッドとスクリーンはプレイヤループ中のワールドモデルを介して情報伝達できるように作られているが、ついでにこのループを介することはシステムの純粋な負担となるため、バイパスを設けて効率を向上している。

4.1 OZRPC

OZRPC (OZ-RP Controller) [1] は、家庭用ゲーム専用機の入力機器(通称コントローラ)をリアルタイムで接続するシステムであり、はこだて未来大学の秋田氏によって開発されている。

11台のコントローラを統合し、ホストコンピュータの1つのシリアル端子に接続できるように工夫している。このため専用のシリアルレベルインタフェースに接続マルチプレクサをPICマイコンを用いて実現している。

4.2 JINN

JINNシステム [2] はゲーム機のコントローラ接続システムで、通常のサッカーベースをベースとしたプレイヤループに、プレイヤビューを表示するサッカーモニタを統合した、単体プレイ操作システムである。豊橋技術科学大学の藤塚氏と渡内氏によって開発がすすめられている。

ディスプレイと接続ホストを1プレイヤにつき1セットずつ必要とする。

4.3 OZip

OZIP (OZ Interactive Player) [7] は高機能サッカーモニタ (SoccerViewer) と、独立してベースイベントSavior [8] によるプレイヤループをベースにしたgtk+ (X11) による、統合型の単体プレイ操作システムである。福井大学の下戸氏によって開発が進められている。

情報統合補完とボールキャッチアップ、目標位置移動がアシストシステムとして実装され、マウスで動作命令を与える。

ディスプレイと接続ホストを1プレイヤにつき1セットずつ必要とする。

4.4 OZRPPWWS

OZRPPWWS (OZRP Wonder Witch System) [4] は、液晶ディスプレイを搭載した携帯ゲーム機を出力力両用の装置とするインタフェースシステムである。

システム構築上の制約である11台のディスプレイ
の準備が容易であると期待される。
ホストとシリアル通信によって、操作入力と情報
表示出力を交換する。１１台を一台のホストと接続
するため、専用のシリアルマルチプレクサを開発中
である。

4.5 検討中のシステムと方式

11台のOZRP端末を素早く簡単に、システムに
負担をかけずに接続する方法は、現在のところ無い
と言える。
問題提起のため、可能な手法とその得失をここに
まとめておく。

- 通信接続形態：シリアル、パラレル、USB、PS/2
  マウス、イーサネット
  USBは、ハブを介した11台のマルチロップ
  接続が可能であるが、ロボカップシミュレーション
  で使われるUNIXサーバで利用できるかどうか
  が不確実である。シリアルはハードウェアとし
  て一般的であり、もっとも有効性が高いが、専
  用のアプリケーションレベルマルチプレクサを
  準備する必要がある。

- スクリーン（表示）形態：各種ディスプレイ、携
  帯ゲーム機、HMD
  互いに干渉しない情報提示としては、HMD（ヘッ
  ドマウントディスプレイ）が最適であるが、可
  用性やコストおよび接続のために専用のハード
  ウェアが必要であるといった問題がある。

- パット（入力）形態：ゲームコントローラ、キー
  ボード、マウス、プロッタ、モーションキャプ
  チャ
  モーションキャプチャによる入力装置は操作感
  覚としては最適である。しかしながら、コストや
  大会会場での準備の面では利用是不可能に近い。

5 まとめ

OZRPシステムとして人間がロボカップシミュレーシ
ョンゲームに参戦するためのフレームワークを提
案した。

このフレームワークの設計とプロトタイプの実装
を通じて、ロボカップの目標である人間対人工シス

参考文献
情報量による移動ロボットの注視制御のためのセンサ空間構成

Sensor Space Segmentation for Visual Attention Control of a Mobile Robot

based on Information Criterion

光永 法明 浅田 稀
Noriaki MITSUNAGA Minoru ASADA
大阪大学大学院工学研究科
Graduate School of Engineering, Osaka University
{mitchy, asada}@er.ams.eng.osaka-u.ac.jp

Abstract

Since the vision sensors bring a huge amount of data, visual attention is one of the most important issues for a mobile robot to accomplish a given task in complicated environments. This paper proposes a method of sensor space segmentation for visual attention control that enables efficient observation taking the time needed for observation into account. The efficiency is considered from a viewpoint of not geometrical reconstruction but unique action selection based on information criterion regardless of localization uncertainty. The method is applied to four legged robot that tries to shoot a ball into the goal.

1 はじめに

移動ロボットには視覚センサが搭載されることが多いが、視覚センサを効率よく使うには適切な注視選択が重要となる。我々は、自己位置の同定ではなく行動決定のための、効率的な観測を行う行動決定法を提案している[1]。この手法ではセンサ値が前もって離散化されており、注視行動数がかかる時間は一定であると仮定していた。しかし、さらに効率的な観測を行うには、注視制御のための自律的センサ空間分割が必要となる。

強化学習の分野では、学習時間が状態空間の大きさに対応して指数関数的に大きくなる[2]ことから、自律的センサ空間の分割による状態空間構成法が提案されている[3][4][5][6]。しかし、観測にかかる時間や、能動的視覚システムは考慮されていない。Kamihara et al.[5]は、注視の粗密制御が結果として得られることが示しているが、用いたセンサは周辺の観測に能動制御を必要としない全方位視覚センサであった。

そこで我々は、注視制御のためのセンサ空間の分割と行動決定法の生成法を提案する。観測にかかる時間を考慮した、自己位置の同定のための観測ではなく、行動決定のための効率的な観測を実現する。

2 情報量に基づく効率的観測と行動決定

ロボットや環境、与えられるデータ等に関して以下のように設定する。1) ロボットは視覚センサを持つが視野角が限られており、受動的な観測のみでは行動決定に必要な情報が得られるとは限らない。2) 環境中にランドマークが配置されており、視覚センサの観測方向を変えることにより、視野を拡大し行動決定に十分な情報が得られる。観測方向の候補は挙げられている。3) 教示などにより、視野を拡大した際に得られる視覚センサの値と、その際にとるべき行動および、その行動後の拡大視野での視覚センサの値（これらを各のモーションデータとする）が与えられる。また行動決定に必要な情報は一定ではなく、状況に応じて変化する。

情報量によりコンパクトな分類を生成する手法としてID3, C4.5 [7]がある。分類木の生成には、トレーニングデータセットが必要であり、各データは分類先のクラスと分類のための属性の値からなる。分類木が行動決定木の場合にはクラスは行動に、属性はセンサに相当する。ID3の場合には属性值として離散値のみを扱う。センサごとについて知ったときの行動に関する情報量I_nをすべてのセンサについて計算し、最も情報量の大きいセンサの値によってトレーニングデータセットを分割し、全てのセンサについて情報を0になる（データセット中の行動が1種類となる）まで、データセットの分割を繰り返す。行動決定木は、ノードがデータセットを分割するセンサ、枝がセンサの値によって伸縮した形となり、葉で行動を表す。C4.5では離散値属性についてはID3と同じであるが、連続値をとる属性の場合には、センサ値と間値の大小関係でデータセットを2分割した際に、情報量が最大に近くなる型
値を求め、関従でデータセットを2分割することで、連続
な属性値を扱う。

本研究では、C4.5を元にロボットのセンサの特性を考
慮した拡張を行った行動決定木生成手法を用いる。視野
の限られた視覚センサを用いて、ランドマークの方位と
関従の大小関係を知るために、ランドマークが観測さ
れている場合を除き、視覚センサの方向を変え観測する
必要がある。これに対し、ランドマークがある範囲の方位
に観測されるか否かを知るためには、その方向の一度の
観測で済む。そこで、データセットの分割をランドマーク
がある範囲に観測されるか否かで行い、分割後の情報量
を比較する。

2.1 観測による情報量
行動の種類を$r$、トレーニングデータの数を$n$とする。ト
レーニングデータ中の行動を$t$と定めた回数を$n_{t}$とすると、
各行動$t=1,...,r$の生起確率$p_{t}$は、$p_{t} = n_{t} / n$である。
このときのpのエントロピー$H_{0}$は、次のようにになる。

$$H_{0} = - \sum_{t=1}^{r} p_{t} \log_2 p_{t}$$ (1)

ランドマーク$i$が[$\theta_{Lk}, \theta_{Uk}$]の範囲に観測されるか否か
が分かったときの事後生起確率を求める。ランダム化$i$が
[$\theta_{Lk}, \theta_{Uk}$]の範囲に観測された回数を$n_{ijk}$、観測されたとき
に行動$j$をとった回数を$n_{ij}$、$n_{ik} = \sum_{j=1}^{r} n_{ijk}$とすると、範
囲内に観測される場合の事後生起確率は、$p_{ijk} = n_{ijk} / n_{ik}$
となる。同様に、観測されなかった場合の行動$j$をとった
回数を$n_{0ijk}$、$n_{0ik} = \sum_{j=1}^{r} n_{0ijk}$とし、観測されない場合の事
後生起確率$p_{0ijk}$を求める。$n_{ik} = n_{ijk} + n_{0ijk}$として、この
ときのエントロピーを計算すると,

$$H_{ik} = - \sum_{x \in \{1,0\}} n_{xik} \sum_{j=1}^{r} \left( p_{x,ijk} \log_2 p_{x,ijk} \right)$$ (2)

となり、それぞれの観測による情報量は、$I_{ik} = H_{0} - H_{ik}$
である。情報量が大きいランドマーク$i$、観測範囲$k$ほど、
行動に関する暖昧さが減少する。観測範囲の上下限$\theta_{Lk}$、
$\theta_{Uk}$は、各ランドマーク$i$についてトレーニングデータ
中に含まれる方位の中心とする。

2.2 観測時間の考慮
観測に要する時間が観測対象(ランドマークとその観測さ
れる範囲)によらず一定の場合には、情報量を最も大きい
観測対象により、行動決定木を生成する。この決定木はコ
ンペクトから、木のノードを繊り返することで最短
観測時間で行動を決定できる。しかし、観測時間が観測対
象により異なる場合には、最短観測時間となるとは限ら
ない。また観測対象決定時に、得られる情報量と観測時間
のトレードオフを計算するのは、観測時の計算コストが
高く、決定木を使うメリットが減少する。

そこで決定木の生成の際に用いる指標を観測時間を考
慮したものとする。前回観測対象の次にその観測対象を
観測するのにかかる時間を$T$とし、単位時間辺りに得
られる情報量$i_{ik}$を次のように求める。

$$i_{ik} = \frac{I_{ik}}{T + a}$$ (3)

ここで $a$ は0で割らないための正の定数である。直前の
観測対象がいない場合には、行動決定時の一時的な状態か
らの観測時間$T$として適用する。すでに視覚センサが観
測した方向である場合には$T = 0$とする。

2.3 行動決定木の生成
$i_{ik}$を最大値とする$i,k$の組により、トレーニングデー
タを、ランドマーク$i$が[$\theta_{Lk}, \theta_{Uk}$]の範囲に観測された
場合と、観測されなかった場合に分け、行動が決定する
まで分割を繊り返す。この分割が木の枝分かれとなら、分
割を繊り返しても、行動が確定しない場合には、各行動
の頻度確率を記しておく。

たとえば、Table 1のトレーニングデータが与えられた
とする。表中の数字は各ランドマークが観測された方向
として視野が限られており視覚センサによる観測可能な方
向が [0,15], [15,30], [30,45]の3つであり、視覚センサは観
測開始時に [15,30]を向いており、$a = 1$観測方向を
変えるのに1時刻必要であるものとする。まず、$H_{0}$を計
算すると、$p_{x} = 2/4, p_{y} = 1/4, p_{z} = 1/4$から、$H_{0} = 1.5$
となる。Landmark A, Bがある範囲に観測されるか否か
が分かったときの、情報量$I_{ik}$と単位時間辺りに得られる
情報量$i_{ik}$を計算するとTable 2となる。そこで、最も単
位時間辺りに得られる情報量が大きい、Landmark Aが
[27,30]に観測されるか否かを確かめる。観測された場合
の、トレーニングデータはデータ番号3のみで行動は$y$
が決定できる。観測されない場合には、トレーニングデー
タ1,2,4が含まれ、行動は決定できない。このトレーニ
ングデータのサンプルセットで、Landmark A, Bがある範
囲に観測されるか否かが分かったときの単位時間辺りの情
報量は、Landmark Bが[0,15]に観測されるか否かが分
かったとき、Landmark Aが[30,40]に観測されるか
否かが分かったときが最も大きくなり、共に0.05である。左
側を優先して観測することにすると行動決定木は、Fig.1
となる。

<table>
<thead>
<tr>
<th>Data #</th>
<th>Landmark A</th>
<th>Landmark B</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>15</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>10</td>
<td>y</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>30</td>
<td>z</td>
</tr>
</tbody>
</table>

—30—
Table 2: Calculated information and information per time of the example training data (Lm means landmark)

<table>
<thead>
<tr>
<th>Observation</th>
<th>$I_{ik}$</th>
<th>Info. per time $i_{ik}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \leq (Lm_A) &lt; 15$</td>
<td>.31</td>
<td>.15</td>
</tr>
<tr>
<td>$15 \leq (Lm_A) &lt; 27$</td>
<td>.31</td>
<td>.31</td>
</tr>
<tr>
<td>$15 \leq (Lm_A) &lt; 30$</td>
<td>.50</td>
<td>.50</td>
</tr>
<tr>
<td>$27 \leq (Lm_A) &lt; 30$</td>
<td>1.4</td>
<td>1.4</td>
</tr>
<tr>
<td>$30 \leq (Lm_A) &lt; 45$</td>
<td>1.4</td>
<td>.70</td>
</tr>
<tr>
<td>$0 \leq (Lm_B) &lt; 7$</td>
<td>.31</td>
<td>.15</td>
</tr>
<tr>
<td>$0 \leq (Lm_B) &lt; 12$</td>
<td>.5</td>
<td>.25</td>
</tr>
<tr>
<td>$0 \leq (Lm_B) &lt; 15$</td>
<td>1.4</td>
<td>.70</td>
</tr>
<tr>
<td>$7 \leq (Lm_B) &lt; 12$</td>
<td>1.4</td>
<td>.70</td>
</tr>
<tr>
<td>$7 \leq (Lm_B) &lt; 15$</td>
<td>.5</td>
<td>.25</td>
</tr>
<tr>
<td>$30 \leq (Lm_B) &lt; 40$</td>
<td>1.4</td>
<td>.70</td>
</tr>
</tbody>
</table>

Observe $(15, 30)$, if $27 \leq (Lm_A) < 30$ then take action $y$
else Observe $(0, 15)$ and if $0 \leq (Lm_B) < 15$ then take action $x$
else take action $z$

Figure 1: Action decision tree of the example data

2.4 観測予測モデルの生成

観測を効率化するためには観測予測を行う。観測予測に用いるモデルは問わないが、一時的前にランダムマークの方位 $x(t-1)$ の多項式で、行動 $a$ を与えた場合の、次時刻のランダムマークの方位 $x_{ia}(t)$ が表されることとする。

$$x_{ia}(t) = \sum_i A_{ai} x^{i}(t) + b_a$$

(4)

ここで、$A_{ai}$, $b_a$ はロボットを環境中で示した際に観測したランダムマークの方位から最小自乗推定を用いて求めた。$A_{ai}$, $b_a$ には推定誤差が含まれるため、$x(t)$ は確率分布（一様分布）として表現する。

2.5 行動決定

行動決定は次のように行う。まず、一時前の $x(t-1)$ の確率分布から、予測モデルを用い近時刻での $x(t)$ の確率分布を計算する。次に、現時刻で視覚センサが向いてる方向について、確率分布を修正する。この確率分布を用いて行動決定書の各葉へ到達確率を計算する。同じ行動を示す複数の葉への到達確率の和を、その行動をとるべき確率とする。特定の行動が選ばれなければ、その行動をとる。そこでなければ、木の根に近い観測箇所を含む段階観測をしていない方向から順に観測を行い、確率分布を更新する。更新した確率分布により、特定の行動をとるべき確率が選ばれるまで、観測と確率分布の更新を繰り返す。

3 実験

ロボットとしては、RoboCup SONY 腿式ロボットリーグのロボット（図2）を用いた。カメラの画角は横 53 度、縦 41 度、画素数はそれぞれ 88, 59 である。脚は 3 自由度、首は 3 自由度（バン、チルト、ロール）ある。ランドマークを観測する際には、脚のロールを固定し、バン、チルト軸のみを利用した。バン軸はロボット正面に対し、-88 度から 88 度、チルト軸は、-80 度から 43 度が可動範囲である。そこで能動的に観測する方向としては、バン軸を 44 度毎の 5 方向、チルト軸を 40 度毎の 4 方向の 20 方向に分けた。バン軸の最大角度速度 5.9[rad/s] は、チルト軸の最大角度速度は 5.9[rad/s] である。また軸が目標値に達した後で画像を安定するまで 0.16[ms] 待ってから処理をしている。カメラの方向を変えて観測する場合に少なくとも 0.29[s]（バン軸の 44 度の回転に相当）かかることから、$a = 0.29[s]$とした。

Figure 2: The SONY legged robot for RoboCup SONY legged robot league.

実験環境を Fig.3 に示す。RoboCup SONY 腿式ロボットリーグのフィールドである。ランドマークは 6、ゴールが 2 あり、ポールが一つある。それぞれ、敵ゴール（TG）、自陣ゴール（OG）、北西ボール（NW）、北東ボール（NE）、中央西ボール（CW）、中央東ボール（CE）、南西ボール（SW）、南東ボール（SE）とする。すべてのランドマークとボールは色により識別される。ロボットがボールを TG に入れることをタスクとする。これを実現するためには場所に応じたボールへの回応込み、ポールの探索などが必要となる。

視覚センサとしては、各ランドマークとボールの図示座標、両ゴール（TG, OG）の画像上の $x/y$ 座標が最小/最大となる座標 4 つを用いた。座標としては画像上の値を直接用いるのではなく、対象が画像中心に観測されるときのバン、チルト軸の角度を用いた。またバン軸
図3：実験フィールド（同様にRoboCup SONY legged robot league）。

(x) 座標あるいはチルト軸 (y) 座標のいずれかの範囲ではなく、x,y の範囲の組（長方形領域）に入ると否かでトレーニングデータを分割するものとした。これは、例えば x 座標の範囲が -10 度から 10 度にあるか否かで分割するとき、y 座標に関しては指定されていないので y 軸についてカメラを回転して何度か観測を繰り返す必要があるためである。

4 実験結果

フィールド中央3点のいずれかから行動を開始して、ゴール正面のボールをシュートすることをタスクとした。行動としては、ボール接近、前進、左 (右) 前進回転、左 (右) その場回転、左 (右) 横移動回転の7つを用意した。各点からシュートするまでの行動を5回ずつ数多した。行動ステップ数は99回となった。提案手法により得られた行動決定木の一部を Fig.4 に示す。図中の文字の単位は角度 (degree) である。観測予測モデルを用いない場合には、一つ目のattention window は、(-19, -8) から (-8, 18) で、その中にボールの囲い内の観測されるかどうかである (Fig.5(a))。このときの観測方向は (3,2) となる。バンの方向 X は左から 1,...,5 で 3 が中央、チルト Y の方向は下から 1,2,3,4 で 3 が正面三者観測方向 (X, Y) で表示。行動決定開始時にはカメラは正面中央 (バン3, チルト2) を向いているものとされている。正面中央 (バン3, チルト2) に観測されるボールやランドマークが判断の基準として優先的に利用されている。ボールの回転がこの window に観測されなかった、次の window は、(-19, -10) から (-18, -13) となり、TG の y が最小となるという観測状態である。観測方向は同じ (3,2) である (Fig.5(b))。対象が window に観測されなかった場合にはボール接近の、それにない場合には前進行動をとる。ボール回転が window 内に観測されなかった場合には、次の window は、(-19, -18) から (-18, 18) となり、TG の x が最小となる観測状態が観測状態である。観測方向は (3,2) である (Fig.5(c))。観測対象が window 内に観測された場合

Figure 3: Experimental field (same as the one for RoboCup SONY legged robot league).

Figure 4: Part of action decision tree generated by proposed method.
Figure 5: Attention windows

センサ値の散乱化が木の大きさ、観測方向数、観測時間を小さくすることに有効であることが分かる。また観測時間を考慮することにより、木は少し小さくなるが、観測時間は半分以下に改善されている。

Table 3 に、3つの手法の実機実験による観測時間をの違いを示す。期待されたよりは長くなっているが、提案手法による観測時間は他の手法の半分以下となっている。観測予測を使うことでさらに観測時間が短縮されると期待される。

Figure 6: Created attention windows by pre-quantization method without time consideration.

Table 3: Comparison of sizes of the tree, expected number of observing directions, and expected time for observation

<table>
<thead>
<tr>
<th></th>
<th># of nodes</th>
<th>depth</th>
<th># of leaves</th>
<th>time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pre-quant.</td>
<td>61</td>
<td>18</td>
<td>31</td>
<td>7.6</td>
</tr>
<tr>
<td>segment. only</td>
<td>39</td>
<td>11</td>
<td>20</td>
<td>5.5</td>
</tr>
<tr>
<td>proposed</td>
<td>59</td>
<td>15</td>
<td>30</td>
<td>3.4</td>
</tr>
</tbody>
</table>

Figure 7: Created attention windows by proposed segmentation without time consideration.

Figure 8: Created attention windows by proposed method.

Table 4: Comparison of mean time for observation

<table>
<thead>
<tr>
<th></th>
<th>pre-quant.</th>
<th>quant. only</th>
<th>proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>time [s]</td>
<td>6.6 [s]</td>
<td>5.2 [s]</td>
<td>2.5 [s]</td>
</tr>
</tbody>
</table>

5 まとめ

自己位置の同定ではなく行動決定のための、視覚センサと観測時間を考慮しつつ、効率的な観測を行う行動決定法を提案した。実験により提案手法が、適切な離散化による行動決定木の圧縮と、観測時間を短縮に有効であることを

確認した。

参考文献


KU-Boxes2001における走行加速度の向上と経路計画について

Improvements in Moving Acceleration and Path Planning of Robots
in Team KU-Boxes 2001

早津啓道, 飯土井修一, 五十嵐治一, 黒瀬能幸
Hiromichi Hayatsu, Shuichi Iido, Harukazu Igarashi, Yoshinobu Kurose

近畿大学工学部（広島県広島市）
School of Engineering, Kinki University, Higashi-Hiroshima

Abstract
This paper describes improvements to our robot system
developed under the JPS-II Project. We changed a
constant-voltage system of a motor controlling circuit to a
constant-current system. That change provided our robot with
accelerating ability about two to three times larger than before.
Moreover, we proposed a roadmap method for planning a
path of a robot. We took account of time necessary for a robot
to turn its body.

1 はじめに
近年, 人工知能とロボット工学の分野において,
動的環境下でのマルチエージェントシステムの標
準問題としてロボットによるサッカーゲームが取
り上げられており, RoboCup(The Robot World Cup
Soccer Games and Conferences)という国際競技会が
1997年から,毎年,開催されている。我々は,1997
年9月から実機小型部門用の共通プラットホーム
作成を目的とする JPS-II プロジェクトを立ち上げ
ている[1-5]. このプロジェクトについては,す
でに, 2000年3月に東京で開催されたRoboCup
Spring Camp Symposium 2000において,一応の総
括を行った[6].

その後,本プロジェクトで開発したロボットシ
ステムの性能評価のために,実際に,2000年6
月に函館で開催された国内公式大会,RoboCup
Japan Open 2000実機小型部門に,KU-Boxes2000
というチーム名で参加し,実際にサッカーの試合
を行わせた[7]. そこで試合の様子を観察して,
①走行時の加速度,速度が不足している,②ロボ
ットモータの消費電力が大きい,③障害物回避と
経路計画が不十分である,④グローバルビジョン
の画像処理速度が不十分,⑤協調プレーの欠如,
などの問題点があることがわかった。本報告では,
上記の問題点のうち,①〜③に関する改良方法に
ついて述べる。

2 走行加速度の向上
2000年3月の段階で,本システムのロボットの
走行性能について評価実験を行った際には,アッ
ドレコニングによるロボットの直進時の距離誤差
は約3％(移動速度150mm/s, 移動距離500mm),
片輪旋回時の角度誤差は約5％(移動速度50mm/s,
旋回角度360°)であり,低速時の走行誤差は少
なく,走行精度についてはまずまずの結果であっ
た。しかし,当時はモータ電圧が3.6Vと低かった
こともあり,最高走行速度はわずか180mm/sに
留まっていた[8].

そこで,今回は,モータドライバを定電圧方式
から定電流方式に変更することにより,高速時の
トルクの確保とあわせて,モータ消費電力の削減
を試みた。

2.1 ステッピングモータの回転数とトルク
本ロボットでは駆動用にステッピングモータを
使用している。一般に,ステッピングモータの回
転数はモータに送るパルス信号Vpの周波数によ
って決まる。回転速度を上げるには入力パルスの
周波数をあげればよいが,モータ電流の立ち上がりの遅れにより,十分大きな電流が流れず(図
1),トルクが減少する。モータを高速で回転させ
るためには,電流を多く流し,高速時のトルクを
高めてやればよい。

従来,本ロボットのモータドライバには定電圧
ドライバを使用していた。定電圧ドライバにおい
ては,高電圧をかけて電流の立ち上がりを早くす
ることにより,モータ電流を多く流し,ある程度
2.3.1 実験手順

速度が上がるときトルクが低下するため、高速度を出すには低速から徐々に加速していく必要がある。そこで、走行速度を一定の値で予め定めた目標速度Vに達するまで加速することにした。図1は、目標速度Vを600mm/sから200mm/s間隔で設定している。各速度での移動をしない最大の加速速度aを調べ、上限速度10ptをつけて、最低速度0ptに10ptずつ速度を上げていくこととした。

2.3.2 実験結果

最大加速速度aの測定結果を表1に示す。定電流ドライバでは1100mm/sまで加速度できたが、定電圧ドライバでは900mm/sまでしか加速度できなかった。表1からわかるように、従来の定電流ドライバでも、今回のように、3.6Vから7.2Vに電圧を上げ、徐々に加速度することにより、最高走行速度を180mm/sから900mm/sへ向上させることができた。さらに定電流ドライバでは1100mm/sの最高速度を設定し、加速度aを定電圧ドライバの約2-3倍にしても脱調せずに走行できた。したがって目標速度に達するまでの時間は1/2-1/3に短縮される。なお、最高速度1100mm/sには1秒で到達することができる。

<table>
<thead>
<tr>
<th>表1 各ドライバでの最大加速速度a</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 定電流ドライバ</td>
</tr>
<tr>
<td>目標速度V</td>
</tr>
<tr>
<td>(mm/s)</td>
</tr>
<tr>
<td>600</td>
</tr>
<tr>
<td>800</td>
</tr>
<tr>
<td>900</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

2.3.3 動制トルクの抑制

走行実験では、両ドライバとも同じ電圧（7.2V）をかけたにもかかわらず、定電流ドライバを用いた場合には、最高速度と加速度が向上した。その理由は次のとおりである。パルスがOFFになった後もモータコイルにはしばらく電流が流れ続ける。特にパルス周波数が高くなった場合(高速走行時)、電流の減衰時間を占める割合が多くなり(図3(a))、コイルに制動トルクが発生し、モータが回転しない。一方、定電流ドライバでは減衰時間がカットされるように設計されている。
3 経路計画

1 章でも述べたように、昨年(00)までの本ロボット(KU-Boxes2000)の障害物回避は不十分であり、
試合中にも他のロボットと衝突することが見られた。また、的確な経路計画アルゴリズムによって
行動決定を行っておらず、目的地へ向けて必ずしも最短経路を走行しているとは限らなかった、
そこで今回は、ロードマップ法に基づく、障害物回避と経路計画の方法について考察した。

3.1 仮想ロートマップ上での最短経路探索

移動ロボットの経路計画問題[1][2]においては、
最短経路の計画をすることがしばしば要求される。その方法の1つにロードマップ法（また
はスケルトン法）がある[11]。これは、2次元の
走行空間上に仮想的なノード・アーケグラフ（ロードマップ）を設定し、ロボットの経路計画を
そのグラフ上で立案すると言う、空間のスケルトン化の一手法である。ロードマップの一例を図6に
示す。簡単のために、以下ではこの例を用いて、
我々の経路計画法を説明する。

図6では、格子状の道路を考え、各格子点をノードとし、ノードを結んだアーカをロボットが通行する
可能な経路とする。図6において、小さな○印
がノード、それら2つを結ぶ線分がアーカを表す。
ここで水平・垂直方向のアーカコスト（走行距離
や走行時間に相当）を1、斜め方向のアーカコス
トを\sqrt{2}とする。また障害物上にあるノードは使
用不可可能なノードとして設定し、それらを始点・
終点とするアーカのコストを無限大とする。図6
において、大きな○印は障害物を表し、小さな○
印のノードは通行不可能なノードを示している。
なお、通行不可能なノードを障害物の大きさより
も若干大きめにっているのは、ロボット自体の
大きさを考慮したためである。このように、一旦、
グラフのモデルができあがれば、グラフ上におけ
る任意の2点間の最短経路は、ダイクトラ法[12]
によって容易に求めることができる。

3.2 ノードコストの表現

ダイクトラ法によって求められた2点間の最
短経路は、現実のロボットが移動する際の最短時
間の経路であるとは必ずしも限らない。なぜなら
ば、現実のロボットが移動する経路では、グラフ上のノードにおける回転コスト（ノードコスト）を考える必要があるからである。我々は、こうしたノードコストを具体的に表現する方式を提案している[13]。ここでは、単純のために4差路の交差点を例に、ノードコストを表現する方式について述べる。

通常行われている4差路の交差点の表現を図4に示す。本方式では、まず、交差点のノードOを各方向に分解し、各方向に1個ずつノードを交差点の手前に新設する（図5のノードN₁～N₄）。次に、新設したノードから各方向へ向かってアークを張る。図5では、ノードO₁から各方向へ向かって張られたアークのみを記しているが、他の新設ノードN₂、N₃、N₄についても同様である。次に新設されたノードから発するアークのコストにノードコストを加えて、それらのアークのコストとする。カーナビゲーションなどの応用で、実際の道路において、車両の右折や左折を禁止したい場合には、無限大のコスト値を加えればよい。

3.3 ノードコストを考慮した最適経路の例

この方法を用いてノードコストを考慮した経路計画の一例を示したのが図6である。図6(a)は、ノードコストを考慮しない場合に得られたStartノードからGoalノードまでの最短経路αである。経路αは、太字で示されて、総コストは26.97である。

次に、ノードコストを考慮した場合の最短経路βを図6(b)に示す。ロボットが交差点で方向を転換するコストをノードコストとする。図6は、回転角に比例した大きさのノードコストを表2のように設定した。

<table>
<thead>
<tr>
<th>回転角度</th>
<th>0°</th>
<th>45°</th>
<th>90°</th>
</tr>
</thead>
<tbody>
<tr>
<td>ノードコスト</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

表3は経路αと経路βについて、ノードコストを考慮する場合と考慮しない場合の総コストを示している。ノードコストを考慮しない場合の総コストは、経路α、βともに差はないが、ノードコストを考慮した場合の総コストは、経路βの方が小さくなっている。これは、ノードコストを考慮した経路βの方が、経路αよりも総回転角度が少ないからである。

<table>
<thead>
<tr>
<th>各経路の総コスト</th>
<th>ノードコストを考慮しない場合</th>
<th>ノードコストを考慮した場合</th>
</tr>
</thead>
<tbody>
<tr>
<td>経路α</td>
<td>26.97</td>
<td>33.97</td>
</tr>
<tr>
<td>経路β</td>
<td>26.97</td>
<td>30.97</td>
</tr>
</tbody>
</table>

4 おわりに

今回、ロボットのモータドライバの定電流方式から定電流方式へ改めることにより、高速走行時のトルクを高めることができた。これにより、従来のKU-Boxesロボットの2倍以上の加速度度で、ロボットを加速することが可能となった。現在のところ、1秒で最高走行速度1100mm/sに到達する。

なお、駆動モータの消費電力の削減に関しては、まだ、電流測定や耐久走行実験などを行っている。
ので正確な結論を出すことはできないが、体感として1/2以下には削減できたという感触を得ている。
また、本論文では、移動ロボットの走行制御に不可欠な経路計画問題に対しては、仮想のロードマップをフィールド上に設定することにより、障害物回避と回転コストを考慮した最短経路探索を行い経路計画法を提案した。この経路計画法は、まだアイディア段階ではあるが、今後は競技用ロボットへの実装や、サッカー競技用以外の応用、例えば、無人搬送車の走行制御への応用を行っていく予定である。

参考文献

[1] 小末将吾，五十嵐治一，黒瀬能流，"RoboCup小型部門用ロボットシステム—JPS-IIグループの現状—" (98.3 ホットトピックスと並列人工知能研究会資料 SIG-HOT/PPAI-9702, pp.1-3)
[2] 小末将吾，五十嵐治一，黒瀬能流，"RoboCup小型部門用ロボットシステムの開発—JPS-IIグループ—" (98.4 第3回JSMEロボメカ・シンポジア論文集, pp.21-24)
[8] 小末将吾，“グローバルビジョンを用いた分散協調型移動ロボットシステム”, "99年度近畿大学大学院工学研究科修士論文".

ヘテロジーニアスチーム OZ における協調的行動の分析

Analysis on cooperations in the Heterogeneous RoboCup Simulation team Open Zeng

伊藤暢浩、西野順二、森下卓哉、久保長徳
Nobuhiro Ito, Junji Nishino, Takuya Morishita, Takenori Kubo

Open Zeng Project
bobson@phaser.elcom.nitech.ac.jp, nishino@se.uec.ac.jp

abstract
In this paper we show an analysis of cooperative teamwork of the heterogeneous RoboCup simulation team Open Zeng (OZ). The team OZ employs eleven different client program, for considering several realistic constraints such as difference and ill communication between clients. As a result, the team makes good match results in both JapanOpen 2000 and RoboCup 2000 Melbourne. Covering area analysis shows the reason of strength of OZ.

1 はじめに
本稿は、公募型ヘテロジーニアスクライアントチーム Open Zeng について、そのプレイの協調性を実際の試合結果に基づいて分析することを目的とする。また、こうした多様で多変数なマルチエージェントシステムの分析指針についても検討する。
チーム Open Zeng (OZ) は、2000 年から RoboCup シミュレーションリーグに参戦した。公募型の異種クライアントによる協調チームである [6, 3]。
通常シミュレーションリーグにおいては、単一アルゴリズムのコピープレイヤによるチームになりがちである。OZ では作業的に、11 体のクライアントを互いに異なる原理とアーキテクチャで作成し、チームとして編成した。このようなヘテロ構成のチームが協調的かつ効果的にシステムとして機能するための要件について検討することを目的としている。
2000 年には日本国内大会のジャパンオープン会館

2 メルボルンでのチーム OZ
2.1 チーム編成

OZ は分散開発によるチーム [4] であり、非集中型の開発形態を取っている。11 体のプレイプログラムは、それぞれ別個の 11 人開発者によって作られている。いっぽうで、各クライアントプログラムがどのようなパフォーマンスを持つか、また持つべきかといったチームとしての指針をあえて作らず、それぞれの開発者に完全に委ねている。このため各クライアントプログラムの詳細な仕様は、開発者以外のメンバーに取っては分からないものであった。他者の開発したクライアントに関する情報の入手は、実際にそのプログラムを起動して、その行動を各自で観察することが基本となっている。
チームとしての各クライアントの最小限の仕様は、それぞれのポジションとして与えられている。これにより、各クライアントのロールと、最低限のチームとしてのまとまりを確保した。
3 作業空間での協調行動の分析

チーム全体としての協調プレイの様相を分析するために、サッカーフィールド全体を作業空間とみなし、プレイ中の行動領域を調べた。

予選で互角に戦ったチーム Virtual Werder との試合結果について、11体のプレイが個々の行動を図 4～図 14 に示した。ゲーム全時間 6000 時間の移動軌跡を表示している。これによって、各プレイの行動域、静的なポジションと、基本的なアルゴリズムが分かる。

3.1 カバリング領域による協調の分析

以下で分析を進めるために、カバリング領域を定義する。カバリング領域とは、各プレイの行動軌跡を含む領域と、それらの和で表される領域である。これが 6000 ステップの全時間を表す静的な行動領域の指標となる。

全てのプレイのカバリング領域として、軌跡全体の和を図 2 に示す。これは図 3 に示した、ボールが移動した領域全体をカバーしている。ここでも、もしカバーされていない部分があるとするとき、チームとしてその領域でのボールコントロールを空けることができないことを示すことができる。

今回の結果では、各プレイの移動領域によるカバーが行われ、チーム協調を発現する必要条件を満たしていると言えることができる。ただしこれは、時間制限に静的な状態での分析であり、時間軸なども考慮したコンフィギュレーション空間で位相的に到達できないボールとプレイ配置が発生しないことを示すものではない。

3.2 各プレイの分析

各プレイの行動アルゴリズムについては、各チーム個々の報告 [2, 1, 5] に掲載されている。ここ
表 1: a Formation of Open Zeng; abbreviations are FW:=Forward player, MF:=Mid field player, DF:=Defense player, GK:=Goalie, Fukui: Fukui University, JAIST: JAPAN Advanced Institute of Science And Technology, Keio: Keio University, Kinki: Kinki University, Mie: Mie University, NITECH: Nagoya Institute of Technology, TITECH: Tokyo Institute of Technology, UEC: The University of Electro-Communications

<table>
<thead>
<tr>
<th>Position</th>
<th>Player name</th>
<th>developer name</th>
<th>affiliation</th>
<th>Orig. team</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>FW</td>
<td>mitsuhide</td>
<td>Suzuki T.</td>
<td>UEC</td>
<td>YowAI 99</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>CZ125</td>
<td>Koto T.</td>
<td>UEC</td>
<td>TakAI</td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td>yukichi</td>
<td>Kinoshiba S.</td>
<td>Keio</td>
<td>11Monkeys 99</td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tanaka, N.</td>
<td></td>
<td>11Monkeys 00</td>
<td>C++</td>
</tr>
<tr>
<td>MF</td>
<td>Tipl</td>
<td>Shinoda T.</td>
<td>JAIST</td>
<td>Kakitsumabata</td>
<td>C++</td>
</tr>
<tr>
<td></td>
<td>wingless birdie</td>
<td>Morisita T.</td>
<td>Fukui</td>
<td>Zeng</td>
<td>C++</td>
</tr>
<tr>
<td>DF</td>
<td>Dajarenja</td>
<td>Igarashi H.</td>
<td>Kinki</td>
<td>RU-Sakura</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Kumanosuke</td>
<td>Hikoki T.</td>
<td>Mie</td>
<td>Shidante</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Bob</td>
<td>Ito H.</td>
<td>NITECH</td>
<td>Kakitsubata</td>
<td>Java</td>
</tr>
<tr>
<td></td>
<td>NS2k(verified)</td>
<td>Nakagawa T.</td>
<td>NITECH</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Esaki T.</td>
<td>NITECH</td>
<td>Java</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nurikabe</td>
<td>Kubo T.</td>
<td>Fukui</td>
<td>gnez</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Castol</td>
<td>Oota, M.</td>
<td>TITECH</td>
<td>gemini</td>
<td>C</td>
</tr>
</tbody>
</table>

図 2: all players coverage

では、試合結果からの分析を行う。

強力なゴーリーであるSaviorは、特徴的な行動を取っている。図4で分かるようにペナルティエリア付近で、ゴールを中心とした円弧を描いて移動している。ボールを持つ相手フォワードと1対1の時に、ゴールを背にして立つという人間が用いているヒューリスティックが実現されている。

他のフィールドプレイヤーの多くは、いわゆるタテの移動をしていることが図2から明らかである。各

図ではx軸方向に直線状の軌跡が濃く重なっていることが分かる。ボールや他のプレイヤーとの関係から、DF(ディフェンダー)もFW(フォワード)もそれぞれにオフサイドラインを意識した行動をしていることを示している。全て異なる開発者であるが、OZに参加したクライアントの属する元チームがそれぞれ、オフサイドラインに敏感な行動を基準としていることが分かる。

また、これらのクライアントは、タテの動きをもとに、必要に応じて左右(図では上下)に移動し、ボール
ルへのアプローチを行っている。逆にいえば、ボールから遠いときには、フィールドのデフェンダーがボシジョンライン付近にそれぞれ戻っていることが読み取れる。

3.3 特徴あるプレイアとチーム関係

特徴的なのは、10番のkumanosuke、2番ダジャレ、7番yukichiの3体の行動とその連係である。

kumanosukeは、図13で分かるように、初期ポジションからほとんど移動していないという個性的な行動をしている。これはクライアントが極力行動しないように方針で作られているためである。クライアント自身が本来発揮できるパフォーマンスと比較すると、機能性は低いプレイアであると言うことができる。しかしながら、全体を示す図2で見ると分かるように、このプレイアの行動域は他のプレイアのカバー域が大きめであり、チームとしては問題が無かったことが分かる。他のプレイアがkumanosuke領域に立ち入らなかったのは、ポジションのため、他のプレイアがkumanosukeの存在を見て、スタミナ保持などでそばにいるように作られていたためと考えられる。つまり、今回のkumanosukeの行動領域の大きさは、他のプレイアのスタミナ消費によってカバーされていないが、その負担は許容量のうちであったと言うことができる。逆に言えば、カバプレイアのスタミナ外でのカバーが必要であったなら、制御不能領域が発生してしまったかも知れない。

ダジャレは、典型的なボール投手型プレイアである。本来のポジションである中央地点は単なる始点であって、試合中は可能な限り移動している。このため、ボールが早いバスで移動してしまうと、戦線から離脱して戦力外となる場面も多かった。いっぽうで相手チームともみあってはいる間にボールに追い付き、ボール周辺での仲間プレイアを増やしボールコントロールが有利にあるという結果があった。

yukichiは、動的な役割を持つプレイアによるチーム11Monkeysからのプレイアである。行動軌跡において他のプレイア同様のタテ移動を中心として、他のプレイアと比較して広範囲まで移動している。このことからも動的な役割を持つ、必要な場面ではポジションを崩してボール周辺での優位性を増すという行動が分かる。すると、フォワード付近ではセント・サイド両域で攻撃に参加していること、左サイドバックでもディフェンダーとして行動していることが分かる。

全体としては、ボール移動領域に対して抜けのないフィールドカバリングができている。このことがバランスの良いチームとして実際に良い結果を導いていると考えられる。

しかしながら、このカバリングは計画された物ではなく、結果として現れたものである。とくに、kumanosuke周辺では移動軌跡の領域カバリングがまばらになっている。抜けが無かった理由は、kumanosukeの直前が、広範囲型プレイアのダジャレであり、また、その前には高機能プレイアのyukichiがいて、動的にカバーされるためである。このため今回は抜けが無かったが、これはある種の偶然であり、チームを編成するときの最低条件としてあらかじめ検討しておく必要がある。

3.4 オフカバリング領域とその解消

全体図2で分かるように、左下の部分は行動カバリングから外れている。これはボールがその付近に来なかったこともあるが、右サイドのプレイアが全て左サイドに比べて中央寄りを規定ポジションとして取っていることによることは図2から明らかである。今回の相手は、ラインがわの深いサイドを使わないチームだったため、この部分のオフカバリングが問題となかった。他のチーム相手の試合ではこの部分を攻撃され不利になっている場合も存在する。

いつもこのような左右の非対称性は、ヘテロチームの特徴である。プレイアをモデリングするようなチームが相手の場合には、こうしたプレイア每の個性がある方が、学習しやすくさを減らすという意味で有効となるだろう。

またsaviorのように一定の曲線しかうかがくことも、敵ボールを確実に止めるプレイア方式もある。これはボールキャッチというゴーリーの特殊性も生かし、かつ敵との位置の相対関係からゴールすそでのボール領域、いわゆるシュートコースを制限していることによっている。行動によってボール領域を間接的に制御して、結果として狭いカバリング領域ながら十分なカバーを行っている。
あわせて考えると、個々のプレイヤの持つべきカバリシング領域を単に広げるのではなく、相手に応じたボールコントロールでボール領域を狭め、そのうえで必要なカバリシング領域を作りだして重ねるということが、チームとして設計することが望ましい。

4 まとめ

プレイヤとボールの移動領域をそれぞれカバリシング領域として定義し、それのさなりによって OZ の試合結果を分析した。その結果対 VW 戦ではチームとして有機的に行動することができ、結果として勝利できたことが分かった。

プレイヤ個々を詳細に分析することなく、全体的な指標を提案しヘテロなチームの効果を判断する手法を提案できた。自分チームで 11 体、相手とボールを含めて 22 体の運動についての詳細な分析は計算量的に考えると常に困難である。このような方法でチーム分析ができれば、OZ のようなヘテロなチームはもちろん、通常のチームの分析にも応用できると考える。

今後の課題としては、実際には時系列での分析や、プレイヤどうしの相関についても、位相的な分析を行い、系統的なオフカバリシング領域の発見方法が必要である。さらにそのオフカバリシング領域解消のための効果的な方策、どのようなプレイヤをどのように修正するべきか、などを考える手法の開発も望まれる。

参考文献


図7: NS2k
図8: Castol
図9: nurikabe
図10: yuki
図11: wingless
図12: CZ125
図13: kumanosuke
図14: mitsuhide
AI チャレンジ研究会

Executive Committee
Chair

Hiroshi G. Okuno
Dept. of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University
Yoshida-honmachi Sakyo-ku,
Kyoto, 606-8501, JAPAN /
Kitano Symbiotic Systems Project, ERATO, JST

Secretary in Charge

Minoru Asada
Dept. of Adaptive Machine Systems
Graduate School of Engineering
Osaka University
2-1 Yamadagaoka, Suita,
Osaka 565-0871, JAPAN

© 2001 Special Interest Group on AI Challenges
Japanese Society for Artificial Intelligence
社団法人 人工知能学会 AI チャレンジ研究会
〒 162 東京都新宿区渋谷町 4-7 OSビル 402号室 03-5261-3401 Fax: 03-5261-3402
(本研究会についてのお問い合わせは下記にお願いします。)

AI チャレンジ研究会
主 委
奥乃 博
京都大学大学院 情報学研究科
知能情報学専攻 音声メディア分野
〒 606-8501 京都市左京区吉田本町
075-753-5376 Fax: 075-753-5977 /
科学技術振興事業団 ERATO
北野共生システムプロジェクト
okuno@nue.org

担当事務

浅田 稔
大阪大学大学院 工学研究科
知能・機能創成工学専攻 創発ロボット工学講座
〒 565-0871 大阪府吹田市山田丘 2-1
06-6879-7349 Fax: 06-6879-7348
asada@ams.eng.osaka-u.ac.jp

SIG-Challenge web page; http://www.symbio.jst.go.jp/sig-challenge/