人工知能学会
第17回SIG-Challenge研究会

2003

2003年5月2日
朱鷺メッセ
(ロボカップジャパンオープン2003)

目次

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-1 (5/2)

Q学習による未知環境下の経路探索
－多目的な問題への適用例－

Path search in an unknown environment by Q-learning

- Application to a multi-purpose problem -

飯土井修一，五十嵐治一*，田中一基
Shuichi Iidoi, Harukazu Igarashi, Kazumoto Tanaka

近畿大学工学部（広島県東広島市）
School of Engineering, Kinki University
* igarashi@ hiro.kindai.ac.jp

Abstract

In this paper, Q-learning was applied to a cliff walking problem, which is a typical example of path searching in an unknown environment. In order to test whether Q-learning is effective to multi-purpose problems, we added two constraints on safety walking and the goal time to the original cliff walking problem. Our experimental results shows that Q-learning can learn an optimal path for the multiple purposes.

1 はじめに

自律移動型ロボットの経路探索問題に関しては，様々な方法が提案されてきた[藤村,1993][Kortenkamp et al, 1998][Hwang&Ahuja,1992][Latombe,1991]．しかし，環境地図が得られない場合には，ポテンシャル法，スケルトン法，空間分割法といった伝統的手法は使えない．そこで，有力なアプローチの一つとして，強化学習のような学習手法を用いる方式がある．

強化学習の中においては，方策価値関数Q(s,a)の学習を行うQ学習が有名である．実際，Q学習を，未知環境下の経路探索問題の一種である「崖歩き問題」に適用した実験結果も報告されている[Sutton&Barto, 1998]．しかし，現実世界における経路探索問題は多目的多制約の探索問題となるのが一般的である．

本論文では，この「崖歩き問題」に対して，なるべく短い時間でゴールへ到達したいという最短性の目的のほかに，崖からの距離を保つという安全性の制約と，ゴールまでの到達時間に関する時間窓制約という２つの制約を持たせた．これは多目的多制約条件を有する未知環境下での行動学習問題の一つと考えることができる．

この例題に対して，報酬関数の与え方を工夫し，Q学習の手法を用いるとどの程度の解を得ることができるか，シミュレーションにより解析した結果を本稿で報告する．

2 Q 学習

2.1 強化学習とは

本章では強化学習（Reinforcement Learning）の概要について，文献 [Sutton&Barto, 1998][新田,2001]に従って述べる．図 1 は強化学習の枠組みを示している．この図でエージェント（agent）は，ロボットのように自律的に行動する主体である．環境（environment）は，エージェントがおかれた周囲の状況である．エージェントのとりうる行動（action）の集合を A とし，観測の結果エージェントが環境内でとりうる状態（state）の集合を S とする．

エージェントにはテレビカメラやマイク，温度計などの「センサ」がついており，自分がおかれた環境を観測することができる．エージェントは時刻 t において，観測された環境内の状態 $s_t \in S$ に基づき，ルールベースを参照して，次にとるべき行動 $a_t \in A$ を選択する．適用できるルールが複数あるときには，何かの手段でルールを選択し，次に行う行動を決定する．選択された行動を実行した結果，環境におけるエージェントの状態が s_t から s_{t+1} に遷移する．

エージェントが当初の目標の状態を達成すると，環境から報酬（reward）r を受ける．これによりエージェントは，

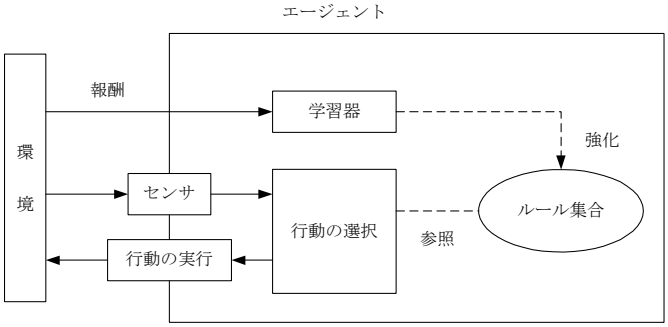


図１　強化学習の枠組み

自分がとった行動が適切だったことを知り，目標に達するために用いたルールの評価を高くする（ルールを強化する）．このようなルールの強化の結果，環境状態から次に選択すべき行動への写像が求まる．この写像を方策（policy）という．

2.2 マルコフ決定過程

状態と報酬値の個数は有限であると仮定する．いま，時刻tにおける環境状態をsとし，エージェントが行動a_tを選択し実行した結果，時刻$t+1$において，状態はs'に遷移し，エージェントは環境から報酬rを受け取るとする．これはエージェントの行動に対する環境の応答と見なすことができる．

この場合，$t+1$における環境の応答が，tにおける状態s_tと行動a_tのみに依存する場合には，次の確率

$$P_{ss'}^{a} \equiv P_r\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{1}$$

を指定することで環境のダイナミクスを定義することができる．また，報酬の期待値$R_{ss'}^{a}$や方策$\pi(a;s)$も現在の状態や行動だけに依存し，過去の履歴には依存しないということを前提とする場合が多い．このとき，環境とタスクもマルコフ性を持つと言い，マルコフ性を満たす強化学習タスクをマルコフ決定過程（Markov decision process：MDP）と呼ぶ．

2.3 価値関数

エージェントが受け取る報酬の総和を収益R_tという．一般に，時間の経過とともに報酬を割引率（discount rate）γ（$0 \le \gamma < 1$）で割引いて合計した次の値で定義される．

$$R_t \equiv r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

エージェントが将来受け取ると期待される報酬は，エージェントがどのような行動を取るかに依存する．したがって，エージェントが現在の状態にいることがどれだけ良いのかを表す価値関数は，特定の方策に依存する．エージェントが方策πをとるとき，時刻t以降の収益R_tの期待値$V^{\pi}(s)$は，時刻tで状態sにあり，それ以降方策πに従ったときの期待値となる．すなわち，マルコフ決定過程における$V^{\pi}(s)$の定義は次のようになる．

$$V^{\pi}(s) \equiv E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \tag{3}$$

$E_{\pi}\{\ \}$は，エージェントが方策πに従うとしたときの期待値を表す．関数V^{π}を方策πに対する状態価値関数（state-value function）と呼ぶ．

同様に，方策πの下で状態sにおいて行動aを取ることの価値を$Q^{\pi}(s,a)$で表し，状態sで行動aを取り，その後に方策πに従った期待報酬を次の式で定義する．

$$Q^{\pi}(s,a) \equiv E_{\pi}\{R_t \mid s_t = s, a_t = a\} \tag{4}$$

$$= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\} \tag{5}$$

Q^{π}を方策πに対する行動価値関数（action-value function），またはQ値と呼ぶ．

全ての状態$s \in S$に対して，状態価値関数が$V^{\pi}(s) \ge V^{\pi'}(s)$であるなら，方策$\pi$は$\pi'$より優れているという．マルコフ決定過程では，他のどの方策よりも優れているか，またはそれに等しい方策が少なくとも1つ存在する．これを最適方策（optimal policy）という．

2.4 Q学習における学習則

Q学習（Q-learning）は状態遷移確率$P_{ss'}^{a}$と期待報酬$R_{ss'}^{a}$とが未知の場合に，環境の各状態$s \in S$において，$V(s)$を最大にするような最適方策を決めることを目的とする．Q学習における学習アルゴリズムを以下に示す．

[Q学習のアルゴリズム]

1) すべての状態sと行動aに対して$Q(s,a)$の値を0に初期化する．
2) 状態sにあるとき，特定の戦略で行動aを選択し，それを実行する．
3) 環境から報酬$r(s,a)$を受け取る．
4) 遷移先の状態s'を観測し，以下の式で$Q(s,a)$の値を更新する．

$$Q(s,a) \leftarrow (1-\alpha) \cdot Q(s,a) + \alpha \cdot (r(s,a) + \gamma \cdot \max_{a'} Q(s',a')) \tag{6}$$

5) s'をsとして，2)へ．

α（$0 < \alpha \le 1$）は学習率とよばれ，Q値更新の強さを制御するパラメータである．

Q学習のアルゴリズムのステップ2)においては，次の行動をランダムに選択することができる．しかし，Q値が収束していない途中の段階でも多くの報酬が得られるようにするため，次の行動選択戦略が提案されている：(i) ε-greedy選択：εの確率でランダムに行動を選択し，そ

れ以外では Q 値が最大となる行動を選択する，(ii)ボルツマン選択：行動 a を exp(Q(s,a)/T)に比例して選択する．T の値は時間が経つにつれて 0 に近づける．次章では，Q 学習における行動選択として，ε-greedy 法を用いた．

3　崖歩き問題への適用について

3.1 崖歩き問題とは

例題として，図 2 に示すような 2 次元平面内をロボット（エージェント）が移動する“崖歩き問題”(Cliff Walking Problem) を考える．ロボットは周囲の 4 方向に移動することができ（行動 $a_1 \sim a_4$），ロボットが環境からぬけ出る方向に移動しようとする場合には，その状態にとどまるとする．このロボットの目的はスタート地点からゴール地点へたどりつくことである．ただし，環境内には「崖」が存在し，落ちてしまうとスタート地点に戻される．

文献[Sutton&Barto, 1998]では，最短経路を学習することを目的として，エージェントは，(i)崖に落ちた場合のペナルティとして報酬値-100 を得る，(ii)1step 移動ごとにペナルティとして報酬値-1 を得る，(iii)ゴール到達時の報酬は 0 とする，という報酬の与え方を採用し，行動の選択には，ε=0.1 の ε-greedy を用いて，Q 学習アルゴリズムを適用した．その結果，崖に沿って移動するという最短経路（図 2 の α）を得ている．

3.2 報酬の設定

前節では最短経路を求めることを学習の目的としていた．しかし，現実世界における経路探索問題は多目的多制約の問題となるのが一般的である．本研究では，崖歩き問題に対して，安全性の考慮やゴール到達時の時間窓制約（ゴール到達時刻の範囲指定）を設定した．

エージェントの位置と，スタート地点からのステップ数とを，エージェントの状態 s とする．ただし，Q 値は位置と行動の対の上で定義し，Q(s,a)と記す．状態 s に対する報酬を $R(s)$ として，$R(s)$ を以下のように定義する．

$$R(s) \equiv c_1 R_{step}(s) + c_2 R_{dist}(s) + c_3 R_{goal}(s) \quad (7)$$

ここで，$R_{step}(s)$ はゴール地点到達時に得られる報酬で，到達時刻 t_{end} が時間窓制約（$t_{end} \in [t_\ell, t_r]$：下限 t_ℓ，上限 t_r）の満足している度合いに応じて与えられる．$R_{dist}(s)$ はエージェントの状態 s の安全性を表しており，崖からの距離に基づき計算する．$R_{\mathrm{goal}}(s)$ はエージェントがゴール地点に到達したことに与えられる報酬値を表している．また，式(7)の右辺のパラメータ c_1, c_2, c_3 は，各項目の重要度を表現している．これらの値は，ユーザの要求に応じて設定する．

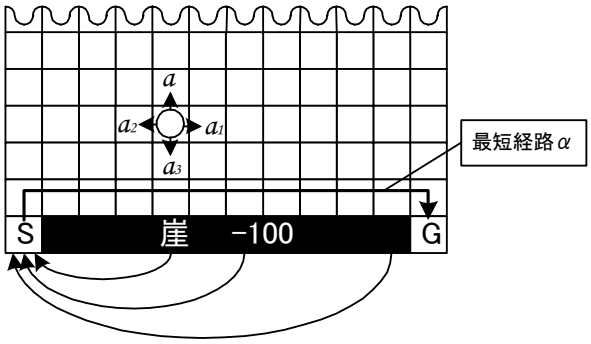


図２　崖歩き問題

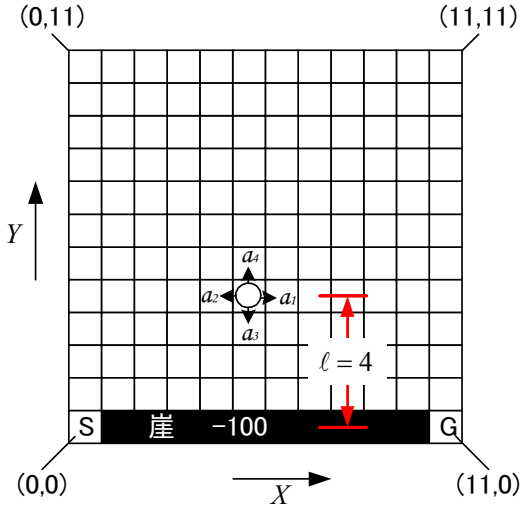


図３　実験で用いた問題

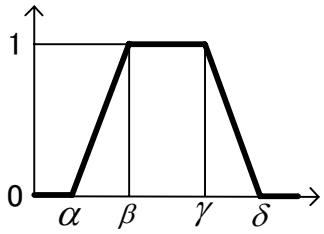


図４　報酬関数で用いた台形状の関数

今回，$R_{step}(s)$ と $R_{dist}(s)$ としては，図 4 のような台形状の関数を用いた．図 4 の横軸は，$R_{step}(s)$ の場合は到達時刻 t_{end}，$R_{dist}(s)$ の場合は崖からの距離 ℓ である．なお，$R_{\mathrm{goal}}(s)$ は，以下のように定義する．

$$R_{\mathrm{goal}}(s) \equiv \begin{cases} 1 & if \quad s\text{が}goal\text{地点} \\ 0 & else \end{cases} \quad (8)$$

4　実験

4.1 実験条件

実験に用いたフィールドマップを図 3 に示す．フィールドの大きさは縦, 横それぞれ12マスに設定した. なお, エージェントから崖までの距離ℓは，状態sにおける位置座標を(x, y)とすると，$\ell \equiv y$で定義する．したがって，図 3 のエージェントの状態sにおけるエージェントの崖までの距離は，$\ell = 4$である．

行動の選択には，ε＝0.1 の ε-greedy を用いた．行動ステップは 1000 ステップを限界とし，そのステップ数内にエピソードの終端状態（ゴールに到達する，または崖に落ちる）に達しない場合に，そのエピソードは強制的に終了し，スタート地点に戻される．以下に示すような 3 つの場合（実験１～実験３）を設定し，実験を行った．

4.2 実験1：最短性を目的とする従来法

3.1 で述べた報酬を与える．また，Q 値の初期値はすべて 0 とし，割引率γ＝0.1，学習率α＝0.1 として，5000 回のエピソードを学習する実験を 5 回（Case1～Case5）行った．1 エピソードの報酬値の結果を図 5 に示す．この図では, 5000 回のエピソードを, 50 回毎の区間に区切り, 各区間での平均値がグラフに描かれている．

実験の結果，5 回とも図２に示すような崖のふちに沿って右方向に移動するような最短経路を最適経路として学習した．しかし，行動選択に ε-greedy を利用しているために，崖から落ちるような事態がときどき起こり，必ずしも最適経路（報酬値-12）を毎回通ることはない．実際，図５の Case1～Case 5 でも，報酬値は-12 とはなっておらず，-40 前後で推移している．

4.3　実験2：時間窓制約

(7) の報酬関数において，$c_1 = 1, c_2 = 0, c_3 = 0$ として，時間窓制約のみを設定した．最短経路の行動ステップ数が 13 であることを考えて，関数$R_{step}(s)$の各パラメータ値はそれぞれ，$\alpha = 15, \beta = 20, \gamma = 25, \delta = 30$ とした．つまり，ゴール到達時刻t_{end}が，20～25 ステップであることが最も望ましく，最短経路（$t_{end} = 13$）は適切でないと要求していることになる．

したがって，実験 2 における報酬の与え方をまとめると以下のようになる：(i) 崖に落ちた場合のペナルティとして報酬値-100 を与える，(ii) 1step 移動ごとのペナルティはない，(iii) ゴール到達時の報酬は，行動ステップ数に応じて，$R_{step}(s)$のみで与えられる．

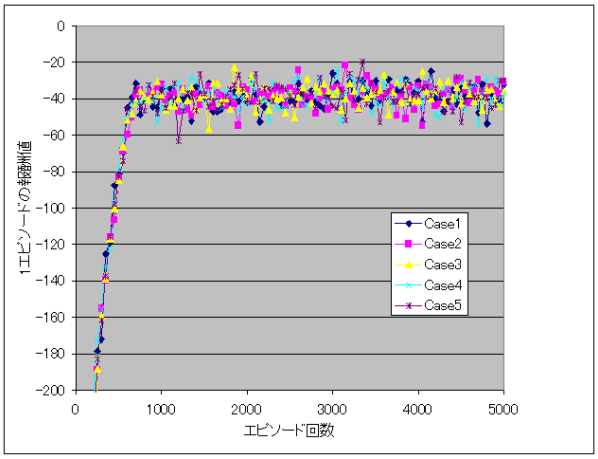
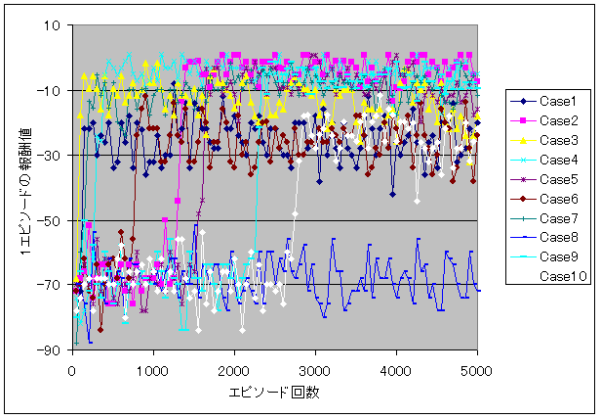


図５　実験１の結果



Q 値の初期値はすべて 0 とし，上記の条件下で，割引率γ＝0.1，学習率α＝0.1 として，5000 回のエピソードを学習する実験を 10 回（Case1～Case10）行った．1 エピソードの報酬値の結果を図 6 に示す．この図では，5000 回のエピソードを，50 回毎の区間に区切り，各区間での平均値がグラフに描かれている．図 6 のグラフの推移は大きく次の 3 タイプ(1)-(3)に分けられる．

（1）Type 1：Case 4 のように，初期の過渡期を経た後，高い水準（～-8）で推移するタイプ（例：Case 2, Case 4, Case 5, Case 7, Case9）．

Case4 では，1 エピソードの報酬値が，高い値（～-8）で推移している．これは，Q 値から得られた最適経路が，崖から一定の距離を保っており，ε-greedy による行動選択を行っても崖から落ちるような事態が起こりにくいためと推測される．

図 7 は学習後の$Q(s, a)$を視覚的に表している．各格子内に描かれている矢印は，各状態sにおける 4 つの$Q(s, a)$の値を矢印の長さで表現している．矢印の長さ

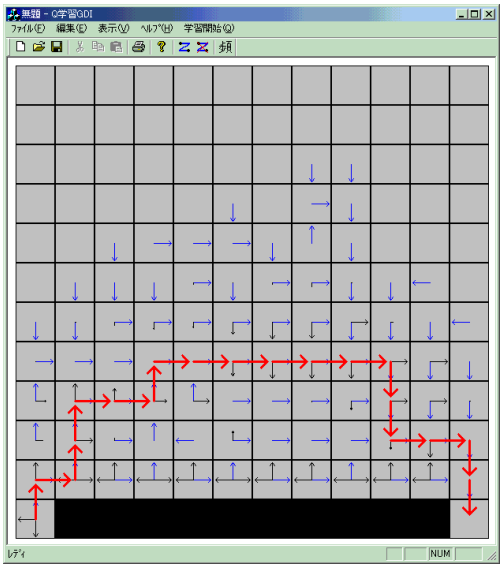


図７　実験２の case4 で得られたQ値と最適経路

$Length(s,a_i)$は式（9）に従って正規化してある．

$$Length(s,a_i)=\frac{1}{2}\cdot\frac{Q(s,a_i)-\min_j Q(s,a_j)}{\max_j Q(s,a_j)-\min_j Q(s,a_j)} \quad (9)$$

ただし，すべての方向で$Q(s,a_i)=0$となる場合は，値を定義できないので描画していない．また，図 7 の太い矢印は，学習の結果得られた最適方策（常に Q 値を max にする行動を選択）に従って，スタートからゴールまで移動して得られた経路（最適経路）を表している．

（２）Type 1’　: Case 1 のように，初期の過渡期を経た後，報酬値が低い水準（～-20）で推移するタイプ（例 : Case 1，Case 3，Case 6，Case 10）．

Q 値から得られた最適経路は，4.2 で行った実験 1 と同様に，崖のふちを右方向に移動して行く経路である．しかし，ε-greedy による確率的な行動選択では崖から落ちるような事態がときどき起こっているため報酬値が低い水準にとどまると考えられる．したがって，Type1’の解は，最短性という目的だけを満足した一種の局所解であると考えられる．

（３）Type 2 : Case 8 のように，報酬値がほとんど変化しない（～-70）タイプ．Case8 では，1 エピソードの報酬値が，初期の低い水準のまま推移している．これは，Q 値が学習できていない（したがって，最適経路も学習できていない）ためである．この実験では行動ステップの上限を 1000 ステップとして，移動を打ち切っている．そのため Case8 では，ゴールに到達し，報酬が得られるエピソードが極端に少なかったと推測される．

4.4　実験3 : 時間窓制約と安全性の制約

(7) において，$c_1=10, c_2=0.00001, c_3=0.1$ とおき，時間窓制約と安全性の制約とを設定した．c_1,c_2,c_3 それぞれの係数が大きく異なるのは，報酬の与えられる頻度が大きく異なることを考慮したためである．つまり，1 エピソード内において，それぞれの報酬が与えられる回数は，1step 毎に与えられる$R_{dist}(s)$が，最大で 1000 回与えられるのに対して，$R_{goal}(s)$はたかだか 1 回，$R_{step}(s)$においては$R_{goal}(s)$よりも制約が厳しいので，更に減少する．これらの発生頻度の違いを係数で調整し，学習における各報酬項の影響を平準化しようと上記のようにc_1,c_2,c_3を設定した．

時間窓制約である$R_{step}(s)$の各パラメータ値は，実験 2 と同様にした．また，安全性の制約である関数$R_{dist}(s)$の各パラメータ値は，$\alpha=0, \beta=5, \gamma=\delta=12$ とした．つまり，これは崖から距離$\ell \geq 5$が最も望ましいことを表している．

したがって，実験 3 における実験条件をまとめると以下のようになる : (i) 崖に落ちた場合のペナルティとして報酬値-100 を得る，(ii) 1step 移動ごとの報酬値は$R_{dist}(s)$により与える，(iii) ゴール到達時の報酬は$R_{goal}(s)$とゴール時の行動ステップ数に応じて$R_{step}(s)$とにより与える．

上記の条件下で，Q 値の初期値はすべて 0 とし，割引率γ=0.1，学習率α=0.1 として，5000 回のエピソードを学習する実験を 10 回（Case1～Case10）行った．1 エピソードの報酬値の結果を図 8 に示す．この図では，5000 回のエピソードを，50 回毎の区間に区切り，各区間での平均値がグラフに描かれている．

図 8 でのグラフの推移からはわかりにくいが，大きく次の二つのタイプ (1), (2) に分けられる．

（１）Type 1 : Case 8 のように，グラフの最大値が 0 を超えるタイプ（例 : Case 6，Case 7，Case 8，Case 9）．

図 9 は，Case 8 の学習後の Q 値とそれから得られた最適経路を表している．Case 8 では，1 エピソードの報酬値が，初期の過渡期を経た後，およそ-5～0 の間で推移しているが，次に述べる Type2 の Case 5 とは違い，最大値が 0 を越えるエピソードも出現している．これは，安全性を確保した上に，時間窓制約の条件をも満たしたステップ数の経路を経てゴールする回数が多かったためと考えられる．実際，図 9 から，安全性の条件$\ell \geq 5$を満足しつつ，望ましいステップ数 (20～25) の最適経路を Case 8 では学習できたことがわかる．

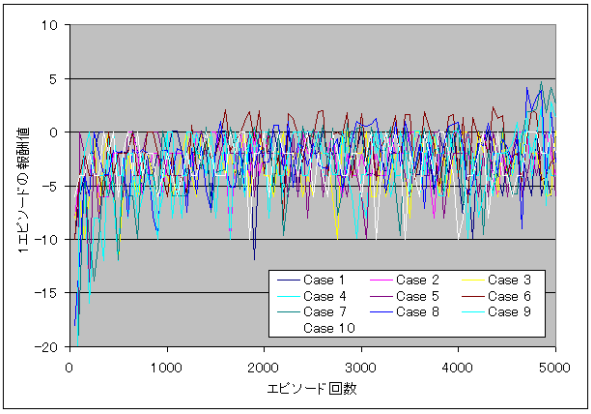


図８　実験３の結果

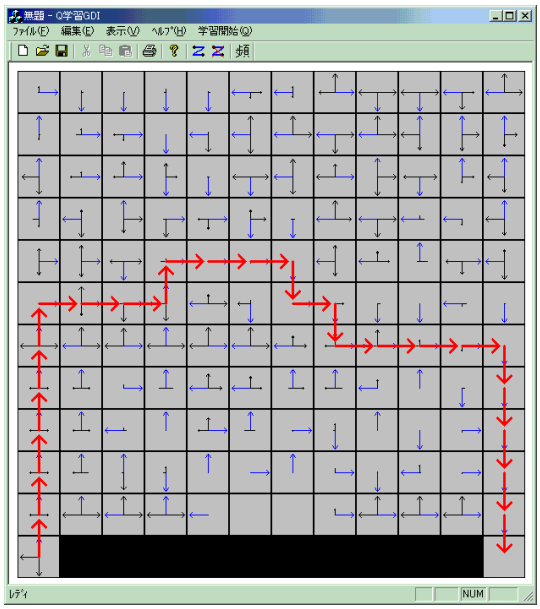


図９　実験３の case8 で得られた Q 値と最適経路

（2）Type 2：Case 5 のように，グラフの最大値が 0 付近のタイプ（例：Case 1，Case 2，Case 3，Case 4，Case 5，Case 10）.

Case 5 では，1 エピソードの報酬値が，初期の過渡期を経た後，−5〜0 の間で推移し，最大値が 0 でとどまっている．これは，実験 2 の Case 8 と同様で，Q 値が学習できていない（したがって，最適経路も学習できていない）ためである．実際，エージェントの行動範囲を調べてみると，殆ど崖から距離が 5 以上離れた領域にとどまり，ゴールに向かう行動の出現頻度が殆ど見られず，学習が失敗していることがわかった．

5　実験のまとめと考察

実験１では，経路の最短性だけを報酬としたために，崖のふちに沿って右方向に移動する最短経路を最適経路とする方策を発見した．しかし，この経路は最短ではあるが，崖のふちを歩く経路のため，常にマイナスの報酬(−100)を得るリスクを負う．

実験２では，経路の最短性に加え時間窓制約を考慮した報酬を与えた．その結果は，(i)Type1：時間窓制約を満足する経路を最適経路と学習する，(ii)Type1’：実験１と同じく最短経路を最適経路と学習する，(iii)Type2：最適経路を全く学習できない，という３つの場合に分かれた．Type2 の場合が生ずるのは，ゴールに到達するエピソードが殆ど無かったためである．

実験３では，経路の最短性，時間窓制約のほかに，さらに，安全性の制約を報酬関数に加えた．その結果，実験 2 での Type 1，Type 2 の 2 つだけが出現し，Type 1’のような局所解は出現しなかった．これは，実験 2 の条件に安全性の制約を加えたことで，実験 2 の Type 1’ のような崖のふちを通る危険な経路は不適切であると学習したためだと考えられる．しかし，安全性の制約を考慮したために，崖から遠く離れて，経路を学習できない Type 2 のケースが増えてしまう現象も見られた．

6 おわりに

“未知環境”の下で行う移動ロボットの多目的多制約な経路探索問題の例題として崖歩き問題を取り上げ，ユーザの要求を満足する経路をQ学習により求めた．この際，報酬関数の与え方について，一つの方法を示した．今回は，報酬関数の各項の重みパラメータ c_1, c_2, c_3 の値を試行錯誤的に決定したが，これらのパラメータ値を，適切に決定する方法が必要である．また，今回は崖歩き問題を例題として取り上げたが，本アプローチの他の行動決定問題への適用も，今後検討していく予定である．

参考文献

[藤村,1993] 藤村希久雄：”行動ストラテジーとアルゴリズム”，日本ロボット学会誌，Vol.11, No.8, pp.1124-1129, 1993.

[Kortenkamp et al, 1998] D.Kortenkamp, R.P.Bonasso, and R.Murphy(eds.): Aritificial Intelligence and Mobile Robots, AAAI Press/The MIT Press, 1998.

[Hwang&Ahuja,1992] Y.K. Hwang and N.Ahuja: "Gross Motion Planning -A Survey," ACM Computing Surveys, Vol.24, No. 3, pp.219-292, 1992.

[Latombe,1991]J.Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1991.

[Sutton&Barto, 1998] Sutton, R.S, and Barto, A.G.: *Reinforcement Learning*, MIT Press, 1998. (三上・皆川訳：強化学習，森北出版，2000)

[新田,2001] 新田克巳：人工知能概論，培風館，pp.144-148, 2001.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-2 (5/2)

Cooperative Behavior based on a Subjective Map with Shared Information in a Dynamic Environment

Noriaki Mitsunaga[*†], **Taku Izumi**[*], **and Minoru Asada**[*†]
∗Dept. of Adaptive Machine Systems, † HANDAI Frontier Research Center,
Graduate School of Engineering, Osaka University
{mitchy, asada}@ams.eng.osaka-u.ac.jp izumi@er.ams.eng.osaka-u.ac.jp

Abstract

This paper proposes a subjective map for a multiagent system to make decisions in a dynamic, hostile environment. A typical situation can be found in RoboCup legged league competition [1]. The subjective map is a map of the environment that each agent maintains regardless of the objective consistency of the representation among agents. Owing to its subjectivity, the method is not affected by other agent's information which may include not negligible errors due to dynamic changes in the environment caused by accidents such as falling down or being picked up and brought to other places by the referee. A potential field is defined on the subjective map in terms of subtasks such as ball reaching and shooting, and is dynamically updated to make a decision to act. The methods is compared with a conventional sharing and no sharing methods, and the future issues are given with discussion.

1 Introduction

In a multi robot system, it is expected that communications or information sharing between robots help aquiring the knowledge about thier environment. When multi robots communicate with each other, they seem to need a refference coordinate system to exchange their information about the environment. In case of mobile robots, the world coordinate or the coordinate fixed to the environment are often used. To convert the observation to the world coordinate, each robot localizes itself. Assuming that localization errors are small or neglectable, information exchange between robots are proposed for self-localization [2]. However, localization errors often become too large to ignore.

Methods to localize itself and to acquire knowledge about environment by shared information from other robots are proposed [3, 4]. They use geometric constraints between several robots which are commonly observed from each other, calculates their positions, and share the environment map. The errors of self-locations are minimized so that shared observations conform to geometric constraints. To observe several robots at one time, they used omnidirectional cameras rather than normal cameras with limited view angles. However, in case of robots with the latter cameras, there will be many situations that they will not be able to observe others. Then it becomes difficult to use such methods.

Although the representation of self-location by probabilistic form and use of beliefs are proposed and commonly used in order to handle the error of self-localization, it is difficult to obtain the accurate model to merge the maps by two or more robots and to maintain the shared map. Simple weighted average of information by robots may work when the errors are small. However when one of the robots has a large error on its self-localization it will affect the shared map used by all other robots. Designing weights or accuracy mesuarements to prevent such a case is difficult because there are always errors that is unknown to the designer and in many situations errors are not detectable by the robot itself. Then, we propose a subjective map based approach rather than shared map based one. A subjective map is for a multiagent system to make decisions in a dynamic, hostile environment. A typical situation can be found in RoboCup legged league competition [1]. The subjective map is maintained by each robot regardless of the objective consistency of representations among other agents. Owing to its subjectivity, the method is not affected by other agent's information which may include not negligible errors due to dynamic changes in the environment caused by accidents. A potential field is defined on the subjective map in terms of subtasks such as ball reaching and shooting, and is dynamically updated to make a decision to act. The methods is compared with no sharing and conventional sharing methods, and the future issues are given with discussion.

2 A subjective map generation

Let us assume that there are two robots (robot A and robot B) and one object (a ball) in an environment. These robots have localized themselves and they are

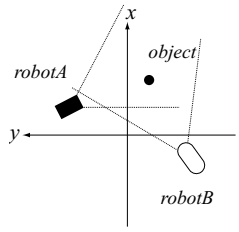


Figure 1: There are two robots watching at a ball.

watching at the ball but they cannot observe each other due to their limited view angles (Figure 1). If we ignore the localization errors and put information on a map, there will be contradiction about the ball position as shown in Figure 2(a).

If we use the weighted average of the ball location $\hat{\boldsymbol{x}}$,

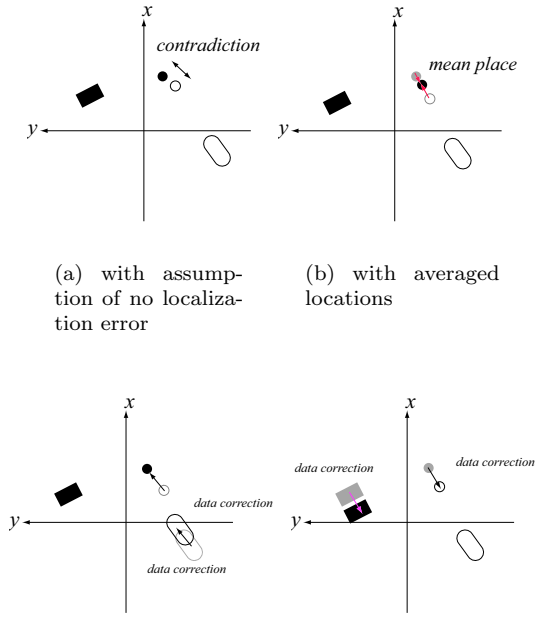
$$\hat{\boldsymbol{x}} = \frac{{}^B\sigma\,{}^A\boldsymbol{x} + {}^A\sigma\,{}^A\boldsymbol{x}}{{}^A\sigma + {}^B\sigma}, \tag{1}$$

where ${}^A\boldsymbol{x}$, ${}^B\boldsymbol{x}$, ${}^A\sigma$, ${}^B\sigma$ are the ball positions and their deviations estimated by robots A and B, respectively, assuming Gaussian distributions. Then we have a map shown in Figure 2(b). There is no contradiction in this map. However, this is not the true ball position in the world coordinate system. When robot A has correct estimation while robot B has incorrect one, robot A's estimation become worse because of the information sharing. Also there are cases the relative position to the robot itself is more important than the absolute position in the world coordinate system. Further, it becomes more complex when the robots can observe each other. If we can assume the simultaneous observations from several robots are available then we could use geometrical constraints and reduce errors [3, 4]. In case of robots with limited view angle cameras and they are moving, we cannot assume it.

Here, we propose that each robot constructs its subjective map and determines its action based on it. For example, robot A believes its observation for the ball and calculate position of robot B from the relative position between the ball and robot B as,

$$\begin{aligned} {}^A\hat{\boldsymbol{x}}_A &= {}^A\boldsymbol{x}_A, & (2)\\ {}^A\hat{\boldsymbol{x}}_Q &= {}^A\boldsymbol{x}_{ball}, & (3)\\ {}^A\hat{\boldsymbol{x}}_B &= {}^B\boldsymbol{x}_B + ({}^A\boldsymbol{x}_{ball} - {}^B\boldsymbol{x}_{ball}). & (4) \end{aligned}$$

Figures 2 (c) and (d) show the subjective maps of robots A and B. With these subjective maps, although reduction of localization error is not achieved, robot A is not affected by the localization error of robot B and can use the information from robot B. The subjective map method is expected to work for such a task and the environment that the relative positions is important rather than absolute ones, localization errors sometimes become large, and geometrical constraints are hard to use. In the following experiments, we compare the action decisions by shared map with average position method



(a) with assumption of no localization error

(b) with averaged locations

(c) the subjective map of robot A

(d) the subjective map of robot B

Figure 2: Constructed map with assumption of no localization error, with averaged locations, and subjective map of robots A and B

and decisions by subjective maps in a robot soccer environment with four legged robots. Robots determine their actions from potential field calculated from a shared map or its own subjective one.

3 A potential field for decision making

We define a potential field which depends on a subjective map of a robot. Each robot calculates the field from the map and decide its action according to the field. A robot has four actions, move forward, turn left, turn right and shoot a ball. It takes such an action that climbs the potential field if the ball is far and shoot it to the opponent goal as it approaches close.

The potential field $V(x, y)$ of robot i consists of three potentials. One is V_F which is the function of the position of a teammate j, ${}^i\boldsymbol{P}_j = ({}^ix_{R_j}, {}^iy_{R_j})$. The second is V_O which is the function of the position of an opponent k, ${}^i\boldsymbol{R}_k = ({}^ix_k, {}^iy_k)$. The last one is V_B which is the function of the ball position ${}^i\boldsymbol{Q} = ({}^ix_Q, {}^iy_Q)$. All the positions are derived from its subjective map. In the following, we give example potentials based on the setup shown in Figure 3.

Potentials by a teammate V_F and opponent V_O are calculated by,

$$\begin{aligned} V_F(x,y) &= -\sum_{j(j\neq i)} f({}^i\boldsymbol{P}_j), & (5)\\ V_O(x,y) &= -\sum_{k} f({}^i\boldsymbol{R}_k), & (6) \end{aligned}$$

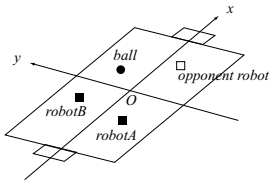


Figure 3: True object locations

where,

$$f(\boldsymbol{P}(\bar{x}, \bar{y}, \sigma_x, \sigma_y)) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2}\cdot\left(\left(\frac{x-\bar{x}}{\sigma_x}\right)^2+\left(\frac{y-\bar{y}}{\sigma_y}\right)^2\right)}. \quad (7)$$

These potentialls are to avoid robots in the field. Figure 4 shows the V_F and V_O of robot A in the example setup.

The potential from the ball is defined so that the robot closer to the ball will reach ball while others will go to the position where it can back up the shoot. The potential function of robot i is switched depending on if i is the closest to the ball or not,

$$V_B(x, y) = \begin{cases} f({}^i\boldsymbol{Q}) & (i \textit{ is the closest to the ball}), \\ f({}^i\boldsymbol{Q}') & (\textit{otherwise}), \end{cases} \quad (8)$$

where ${}^i\boldsymbol{Q}$ is the position of the ball, and ${}^i\boldsymbol{Q}'$ is the support position. ${}^i\boldsymbol{Q}'$ is defined as, ${}^i\boldsymbol{Q}' = \left({}^i\boldsymbol{Q} + \boldsymbol{G}\right)/2$, where $\boldsymbol{G}$ is the position of target goal. The example potentials of robots A and B are shown in Figures 4 (c) and (d). Final potential fields are shown in Figures 4 (e) and (f).

4 The experiments

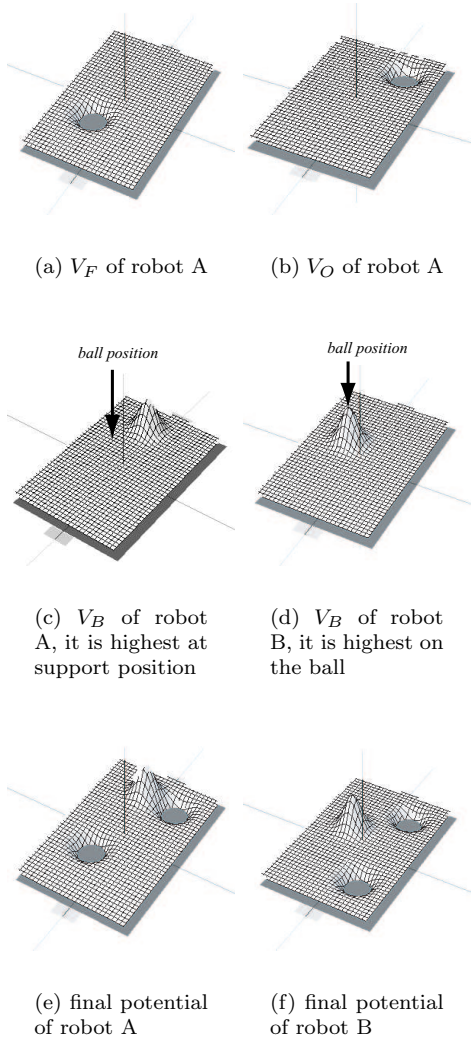
We used the field and the robots for RoboCup SONY Legged Robot League 2002 (Figure 5) for the experimental setup. In the environment there are six landmark poles and two goals that can be used for self-localization, and one ball. The task is to shoot the ball into the goal. Due to the inadequate lightning condition, it was not possible for robots to stably detect each other in their local vision system. Our self-localization program is based on Carnegie Melon University's CM-Pack'01 [5]. Its approach is a Kalman-filter based self location tracking with multi hypothesis which start from different points in the field. For the verification of the method, we put a color marker on the back of each robot and an overhead camera to measure the position of robots and the ball externally.

In the experiments we used two robots A and B and compared the decision of robot A to the deicision with correct positions from overhead camera for three methods,

(a) robots do not share information.

(b) robot B sends information to robot A and robot A uses the shared map made by taking average,

(c) robot B sends information to robot A and robot A uses its subjective map (proposed).



(a) V_F of robot A

(b) V_O of robot A

(c) V_B of robot A, it is highest at support position

(d) V_B of robot B, it is highest on the ball

(e) final potential of robot A

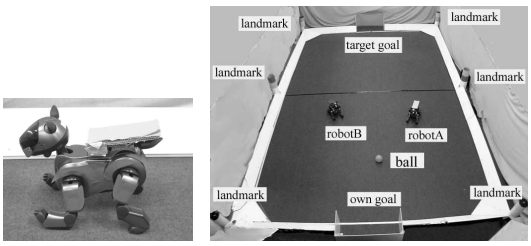
(f) final potential of robot B

Figure 4: Potential fields of robots.

With this condition we can see whether a method makes a wrong decision in spite of information sharing. Robots A and B localize themselves by local vision and large errors were added to robot B depending on the experimental conditions. We compared the results with the rate of the time when the decision of robot A matched with the decision with the overhead camera. Each trial ended when two minutes elapsed or the ball is kicked into the goal.

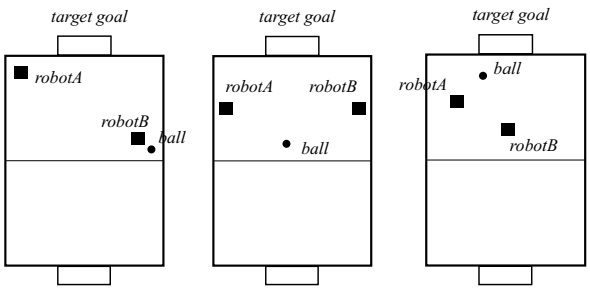
We show the experimental results with three initial conditions. Figure 6(a) shows the initial condition of case 1. Robot B localizes itself by local vision. In this case robot A cannot observe the ball but robot B observes it. Due to large errors on observation of a robot, robot A uses the robot B's position that robot B tells even when it adopts a subjective map. Then, in this case, methods (b) and (c) have the same map which ball and robot B's positions are transfered from robot B.

Figures 6 (b) and (c) show the initial conditions of cases 2 and 3. Both robots observe the ball. In these cases, we experimented with two conditions, 1) robot B localizes itself by local vision and 2) robot B localizes



(a) robot (b) field

Figure 5: The robot and the field for the RoboCup 2002 SONY legged robot league.



(a) case 1 (b) case 2 (c) case 3

Figure 6: The initail conditions of experiments.

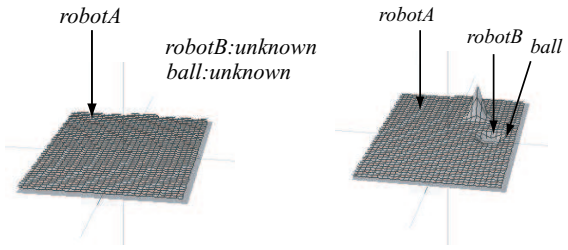
itself but large errors are always added to its position and direction.

Table 1 shows the rate of time that the robot A's decision matched with the decision by the overhead camera. We experimented ten trials for case 2 and 3 for each condition. We can see that with averaged shared map, when the large errors are introduced, the rate is worse than the one without information sharing. However, with the subjective map, in all conditions, the rate is better than the one without infomation sharing and the one with averaged shared map. This indicates the validty of the subjective map approach.

Figures 7 (a) and (b) show the potential field and the positions of objects with its non-shared map and

Table 1: The rate of time that the robot A's decision matched with the decision with overhead camera

error and method		rate of matched time	
		mean	variance
normal	no share	0.48	0.02
	averaged	0.64	0.01
	subjective	0.69	0.01
large	no share	0.48	0.02
	averaged	0.39	0.01
	subjective	0.59	0.02



(a) without information sharing (b) subjective map

Figure 7: Robot A's potential field and positions of objects in its map at the initiall condition in case 1. Without information sharing (left) and based on subjective map (right).

the subjective map in case 1. Without information sharing, robot A did not know the location of the ball (Figure 7(a)) and could not move to the support position. With subjective map, robot A showed the expected movements as shown in Figure 8.

In case 2, without information sharing, robots bumped into each other because they did not know where the other robot was, but they were initially placed at nearly same distance from the ball. With information sharing under normal localization errors, they showed cooperative movements. Figure 9 shows such movements.

In case 3, the matched time rate of averaged map showed small difference to no infomration sharing. While under conditions that robot B had large localization errors, the result of averaged map and subjective map showed a big diffrence. Figures 10 (a) and (b) show the potential field of robot A and the positions of objects in its averaged map and its subjective map at the initial position. Due to the robot B's large localization errors, robot A also became to have large errors with the averaged map and made wrong decisions. However, as shown in Figure 10(b), with the subjective map, robot B's error did not affect robot A and the rate increased with information sharing. Figure 12 shows the robots' movements with subjective map. We can see that robot A apropreately moved to the supporting position.

5 Discussions and Conclusions

We proposed to use a subjective map in a multi robot system under dynamically changing environment. Our experiments with a robot soccer task showed that even under large error situations, where a shared averaged map method lost the merit of information sharing, the proposed subjective map works effectively. Further investigation for a larger scale of multi robot systems and comparison with other information sharing methods are our future work.

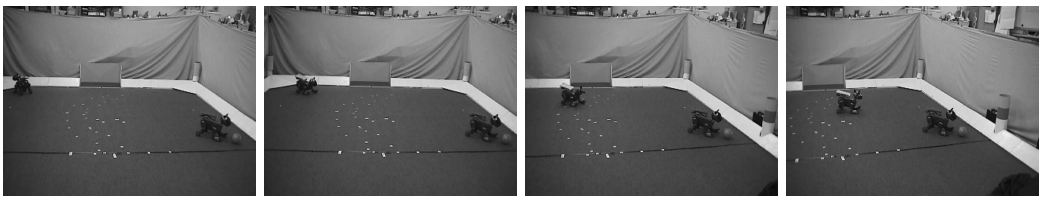


Figure 8: A sequence of robots' movements with subjective map in case 1.

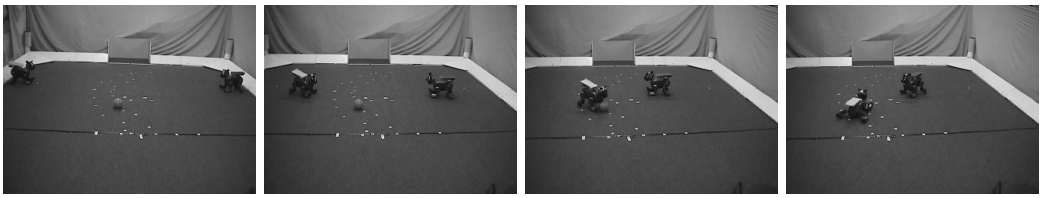
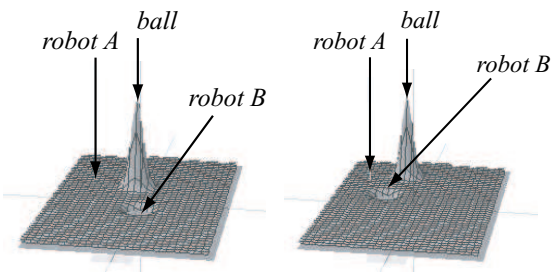


Figure 9: A sequence of robots' movemens using subjective map in case 2 under nomal localization error



(a) averaged map (b) subjective map

Figure 10: Robot A's potential field and positions of objects in its map at the initiall condition in case 3 with robot B's large errors. Based on the averaged map (left) and the subjective map (right).

Acknowledgment

We used the codes shared among the teams involeved in RoboCup SONY Legged robot league. We thank Carnegie Melon University for their self-localization codes, the United Team of Kyushu for their motion and walking codes, and the University of New South Wales for their walking codes.

p

References

[1] Manuela Veloso, William Uther, Masahiro Fujita, Minoru Asada, and Hiroaki Kitano. Playing soccer with legged robots. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 437–442, 1998.

[2] Ashley Stroupe and Tucker Balch. Collaborative probabilistic constraint-based landmark localization. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 447–453, 2002.

[3] K. Kato, H. Ishiguro, and M. Barth. Identifying and localizing robots in a multi-robot system environment. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 966–971, 1999.

[4] T. Nakamura, A. Ebina, M. Imai, T. Ogasawara, and H. Ishiguro. Real-time estimating spatial configuration between multiple robots by triangle and enumeration constraints. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2048–2054, 2000.

[5] William Uther, Scott Lenser, James Bruce, Martin Hock, and Manuela Veloso. CM-Pack'01: Fast legged robot waking, robust localization, and team behaviors. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, pages 693–696. Springer, Lecture Note in Artificail Intelligence (2377), 2002.

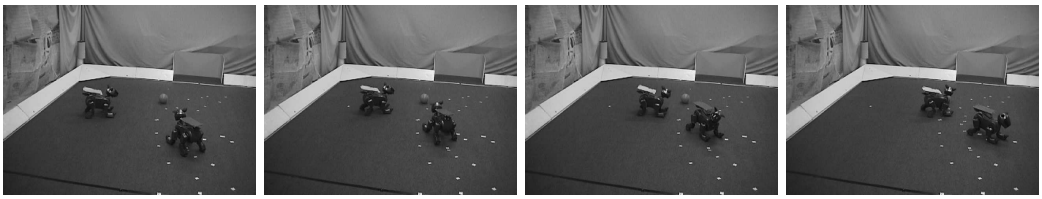


Figure 11: A sequence of robots' movements using averaged map in case 3 with large error on robot B.

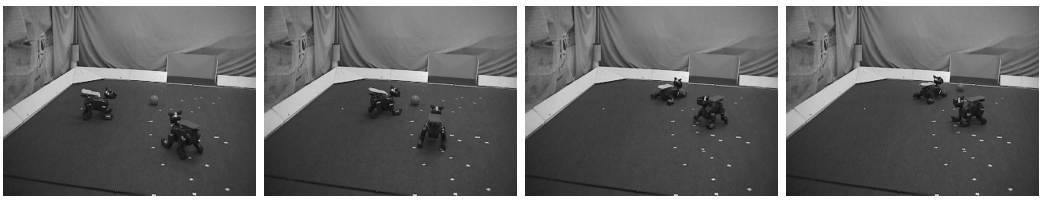


Figure 12: A sequence of robots' movements using subjective map in case 3 with large error on robot B.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-3 (5/2)

小型マルチロボットプラットフォームのための 天井カメラを用いた疑似ローカルビジョンシステム

Pseudo-Local Vision System Using Ceiling Camera for Small Multi-Robot Platform

升谷 保博　田中 志尚　重田 智也　宮崎 文夫

Yasuhiro Masutani　Yukihisa Tanaka　Tomoya Shigeta　Fumio Miyazaki

大阪大学大学院基礎工学研究科

Graduate School of Engineering Science, Osaka University

masutani@me.es.osaka-u.ac.jp

Abstract

Pseudo-local vision system, which simulates visual information derived from an on-board camera of mobile robot based on a ceiling camera image, is proposed. It consists of a vision server and a client module which communicate with each other in the SoccerServer-like protocol. An image processing module for the on-board camera in the control program is replaced with this system. The simulated visual information is not a two-dimensional image data but a one-dimensional array which represents the nearest edge in each direction around the robot. However, it contains much of essential information of the on-board camera image. This concept was implemented for our robot system for the RoboCup Small-Size League. The server can transmit the edge data to 10 clients 30 times per 1 second. The time lag between grabbing image on the server and extracting visual information on the client is about 10ms.

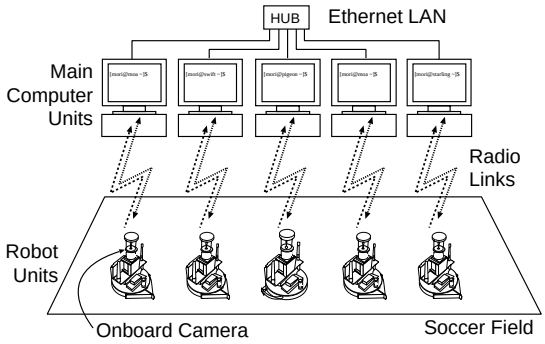


Figure 1: System of the Team OMNI in 2001 (real local vision)

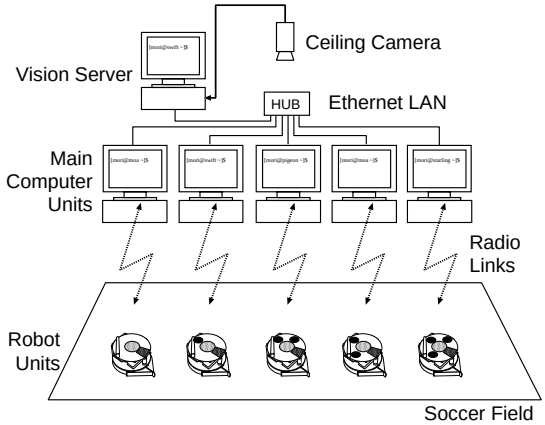


Figure 2: System of the Team OMNI in 2002 (pseudo-local vision)

1　はじめに

視覚を有する複数のロボットの協調は大変興味深い研究テーマであるが，RoboCup の中型リーグのように，ロボットが大きくなりがちで，広い実験環境が必要である．それに対して，我々は，ロボットに搭載したカメラ (ローカルビジョン) を用いるプラットフォームを開発し，RoboCup 小型リーグに参加してきた．しかし，CPU の処理能力等のために，全てのシステムをロボット上に搭載することは容易ではない．そこで，搭載カメラの映像を電波でフィールド外の PC に伝送し処理を行なっている (Figure 1 と Figure 3 左 [Sekimori, 2002b])．このため，混信などの映像伝送の不具合に非常に弱く，RoboCup2001 においては研究成果を十分に発揮できなかった．

そこで，天井カメラの映像を用いて，擬似的に搭載カメラから得られる情報を生成し，搭載カメラ用の画像処

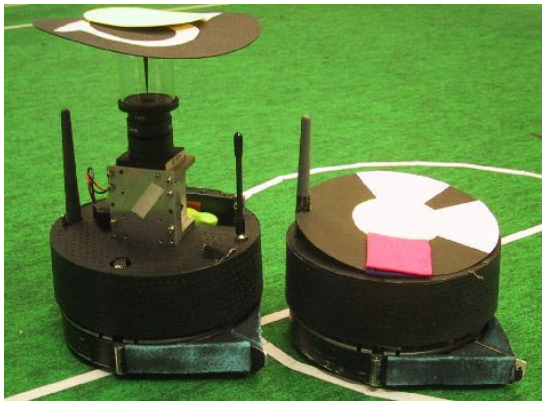


Figure 3: Robots in 2001 (L) and 2002 (R)

理モジュールを置き換える「疑似ローカルビジョンシステム」を開発した．これは，サーバクライアントシステムであり，天井カメラの映像を取得したサーバが，ネットワークを介してそれぞれのロボットの制御を担当するクライアント群に情報を送る．我々は，このシステムの初期のバージョンを使って，実際に RoboCup2002 に参加した (Figure 2 と Figure 3 右[Masutani, 2003])．

このようなシステムにおいて，天井カメラの情報から搭載カメラの情報を生成するにはいくつかの方法が考えられる．しかし，2 次元データである画像を最大 10 台のクライアントに 1 秒間に 30 枚送るというのは，ネットワークや PC に対してかなり負担が大きい．そこで，本論文では，各ロボットを中心とする各方向の最近接エッジの情報をサーバがクライアントに送るという方法を提案する．最近接エッジとは，ロボットを始点としてある方向に床面上を辿った際に始めてぶつかるものとの交点であり，各方向のエッジデータをまとめて 1 次元の配列を成している．一例として，Figure 4 には，フィールド上の robot A の最近接エッジを示す．それを 1 次元データとして表現したものを Figure 5 に示す．本論文では，この 1 次元のエッジデータを「疑似ローカルビジョン情報」と呼ぶ．これは，本当の画像ではないが，搭載カメラの画像から得られる本質的な情報を多く含んでいる．例えば，物体の隠れや重なりという現象を模擬できる．このシステムは，RoboCup に限らず，自律分散型の移動ロボットの研究プラットフォームとして有用である．

既報では，サーバとクライアントの間の通信量を減らすために，SoccerServer のプロトコルと同じように，サーバから極座標の位置情報を送って，クライアント側で最近接エッジの計算を行なっていた[田中, 2002a][田中, 2002b]．その後，サーバ側で最近接エッジを計算しそれをクライアントに送る方法を試したところ，十分な性能が得られたので，本論文ではその方法を前提として，疑似ローカルビジョンシステムの提案をまとめる．

以下では，**第 2 節**で，疑似ローカルビジョンシステム

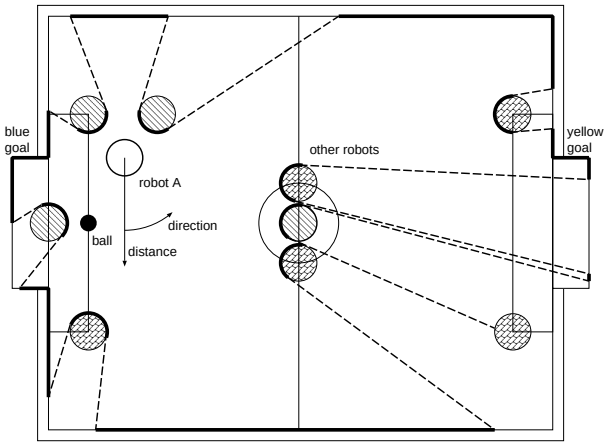


Figure 4: Concept of the nearest edge

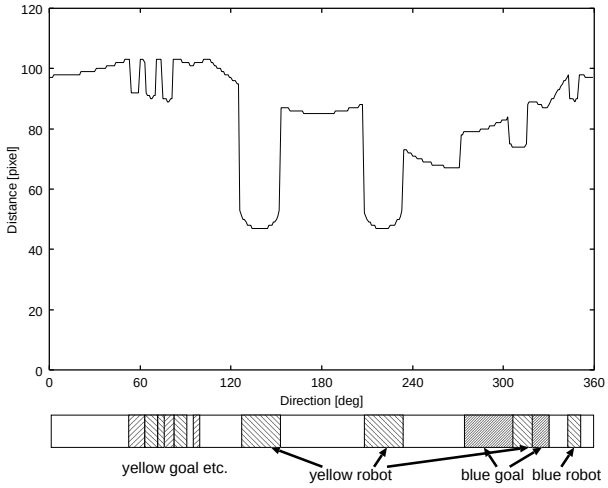


Figure 5: Pseudo local vision information

に必要とされる機能に関して議論する．次に，**第 3 節**でシステムの構成を説明し，**第 4 節**で開発したシステムの評価する．最後に**第 5 節**で今後を展望について述べる．

2 必要な機能に関する検討

2.1 要件

疑似ローカルビジョンシステムに求められる要件について考える．

使いやすいプラットフォームとしては，以下のような項目が必要である．

- 搭載カメラから得られる情報を模擬できること．
- 搭載カメラ用のプログラムのコードをできるだけそのまま使えること．
- 天井カメラ以外に特別な追加のデバイスや装置を必要としないこと．
- 複数のロボットに，1 秒間に 30 回データを提供できること．

さらに，RoboCup の競技用の枠組みとしては，以下の

ような項目が必要である．

- 全てのチームが公平にデータを共有できること．
- 最大 10 台のロボットに，1 秒間に 30 回，少ない伝達遅れでデータを提供できること．
- わかりやすいプロトコルを利用すること．

2.2 視覚情報の模擬

天井カメラから得られるのは床面の 2 次元的な情報なので，その像を直接変換して，床面上に存在する 3 次元的な物体の画像を作ることは不可能である．つまり，天井カメラで得られた位置情報と物体の 3 次元の幾何モデルから，画像情報を再構築する必要がある．最近の高性能なプロセッサを使えば，幾何モデルに基づいて，リアリティの高い CG 画像を高速に作ることができるかもしれない．しかし，10 台分を 1 秒間に 30 枚となると，かなりの能力が要求される．

2.3 データの伝送

サーバが，CG 画像をアナログのビデオ信号として出力できれば，画像の取り込みのハードウェアも含めたシステムの検証が可能であり，我々のチームにとってはソフトウェアの置き換えを一切しなくてもよい．しかし，搭載カメラのシステムを持っていないチームにとっては，余分なハードウェアが必要である．

サーバが，画像をデジタルのまま非圧縮で送る場合，8bit/pixel，640 × 480，30frame/s を 10 台分とすると，703Mbps である．現在の標準的なイーサネットで送るには厳しい．ギガビットイーサネットや IEEE1394 を使えば解決するかもしれない．

2.4 提案する方法

全方位カメラを搭載したロボットを使って RoboCup 小型リーグに出場してきた経験からすると，搭載カメラの情報でもっとも重要なのはロボットの周囲の移動可能な床領域とその先にあるオブジェクトの種類であり，また，周囲のオブジェクトの情報を獲得するにあたって，オブジェクト同士の重なりや隠れが大きな問題になる．そこで，我々は，2 次元の画像の代わりに，ロボットを中心とした各方向の床面上の最近接なエッジという 1 次元の配列を擬似的な画像情報として使うことを提案する．この情報を「疑似ローカルビジョン情報」と呼ぶ．一例として，Figure 4 のようなフィールド配置における robot A の最近接エッジ情報を太線で示す．また，これをロボットのローカル座標における角度に対する分布として変換したものを Figure 5 に示す．Figure 5 の上段は最近接エッジまでの画像上の距離，下段は最近接オブジェクトの種類を示している．

3 開発したシステム

3.1 全体構成

擬似ローカルビジョンシステムはサーバクライアントモデルを使用しており,「ビジョンサーバ」と各ロボット用の制御プログラムの一部である「クライアントモジュール」から成る (Figure 6)．両者は，ネットワークを介して接続されている．1 つのサーバを同時に 2 つのチームが使うことを考慮して，最大 10 クライアントまで接続することを想定している．

ビジョンサーバは天井カメラから得られる画像を処理し，各ロボットとボールの位置・姿勢を認識し，接続されている全てのクライアントに対応するロボットの近接エッジ配列を計算し 33ms 間隔でそれを送信する．

クライアントモジュールは，サーバと通信し，送られてきたエッジデータを上位のルーチンで使えるように変換する．これは，搭載カメラ用のプログラムの画像処理モジュールと置き換えられる．

3.2 プロトコル

通信のプロトコルは，UDP/IP 上で SoccerServer[SoccerServer, -]に似たものを用いる．接続の初期化は，SoccerSover と同じである．つまり，最初にクライアントからサーバへ (`init`) メッセージを送り，その後，サーバから指定のポートへ視覚情報が送られる．サッカーサーバと異なるのは，クライアントの方からチーム色と背番号を指定することである．

視覚情報は，画像取得時刻，エッジ情報，ボール情報よりなる．ボールを別にしているのは，他の物体に比べてボールが小さいからである．視覚情報のメッセージのフォーマットを以下に示す．

(`see` *Time* (`(edge)` *EdgeSstring*) (`(ball)` *Distance Direction Size*))

Time 画像をキャプチャした時刻．その日の深夜 0 時からの経過秒数．後述するように，サーバとクライアントで 1ms 以下の精度で時刻の同期を合わせることによって伝達遅れの掌握が可能．

EdgeString 最近接エッジのデータを表す文字列．3 文字で 1 方向をあらわす．1 文字目がエッジの種類．2〜3 文字目が画面上での距離 (画素単位) を 16 進数で表したもの．エッジの種類のうちロボットの表現方法については 3 つの選択肢がある．

- 全てのロボットが同じ．
- チームの区別ができる．
- ロボット番号の区別までできる．

Distance ボールまでの画像上の距離 (画素単位)．

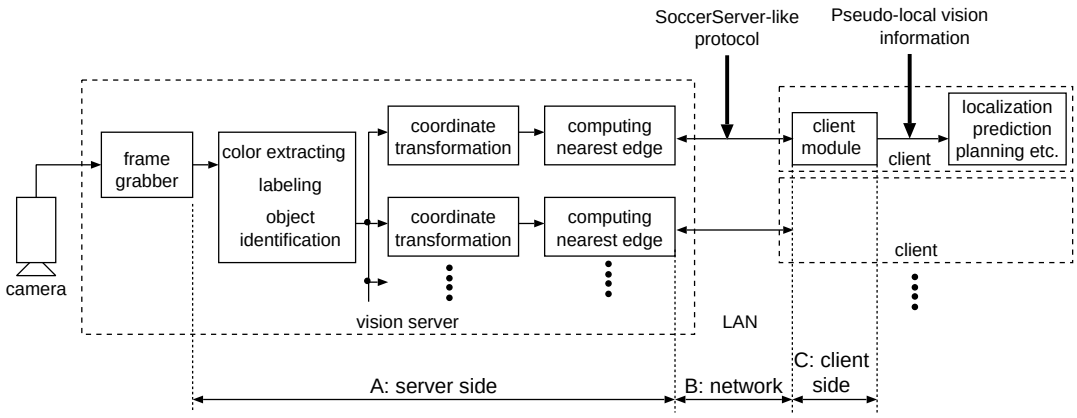


Figure 6: Information flow in pseudo-local vision system

Direction ボールまでの方向 (度単位).

Size ボールの画像上で大きさ．周方向の幅 (度単位).

視覚情報の一例を以下に示す.

```
(see 20306.148 ((edge) r59r5ar5av3e···
) ((ball) 61 121 3))
```

サーバは天井カメラの画像が取得されるたびに，つまり，33ms 間隔で，これらのメッセージを全てのクライアントに送信する．メッセージを送信する順は毎回ランダムに決める.

3.3 ビジョンサーバ

ビジョンサーバは天井カメラの映像に基づいて，ロボットに取り付けられたマーカで個体の識別を行い，位置と姿勢を算出し，さらにボールの位置を測定する．画像の取り込みには安価なキャプチャーカードを使っている．色領域の抽出には，CMVision[Bruce, 2000]を改良したものを用いている．また，情報の質を向上するために，カルマンフィルタを使っている.

その後の処理は，接続しているクライアント毎に別々に行なう．全ての物体の座標を着目しているロボットの座標系に変換する．ロボット座標系において，原点を通る各方向の直線と物体の交点を求めて，最近接エッジを求める．ロボットとボールは円として表現する．次に，その距離を画像上の距離に変換し，整数化する.

現在の実装では，視野角は 360 度 (全方位視覚)，方向の量子化は 1 度刻み，床面の距離と画像上の距離の関係は我々の使っている全方位カメラの双曲面ミラーに合わせている．その変換式は，以下のように表現される.

$$R = \mathrm{int}(f^{-1}(r)) \tag{1}$$

$$f(R) = \frac{R}{A - B\sqrt{C + R^2}} \tag{2}$$

ここで，r はロボット中心からの床面上の距離，R は画像中心からの画像上の距離，A, B, C は定数，int() は整数化の関数である．現在は，これらの設定は全て固定されているが，次のバージョンでは可変にする予定である．また，時々画像が取得できなかったり，画像が乱れたりという現象を模擬することもできる．実際，自己位置推定の処理のロバスト性を検証するためにそのような処理を実装している.

以上の処理を最大 10 台のクライアントに対して行なう．全てのクライアントの最近接エッジを計算した後に，ランダムな順番でクライアントに前節のプロトコルで情報を送る.

3.4 クライアントにおける処理

サーバから送られてくるエッジデータを用いて，搭載カメラ画像に基づく情報を置き換える．我々の搭載カメラ用のシステムではマルチスレッドライブラリを使ってプログラムを記述しており，その中で，最下位の画像処理 (画像のキャプチャ，色抽出，ラベリング，テンプレート画像との AND 処理) を 1 つのスレッドで受け持っている．そこで，そのスレッドを疑似ローカルビジョン用のものに置き換える．他のスレッドは全く変更なしに使うことができている.

我々の実装では，そのスレッドで，床領域の外周点の抽出の模擬，ボール領域と 2 つのゴール領域の重心計算の模擬，床領域とテンプレート画像の AND 処理の模擬を行っている．それらの情報を使って，上位のスレッドでは，移動可能領域の検出，自己位置推定，ボール位置推定などを行っている[Sekimori, 2002a].

3.5 サーバとクライアント間の時刻の同期

動的な物体の画像を扱うには，画像を取得した時刻を正確に知る必要がある．このシステムでは，画像を取得するビジョンサーバとそれを使うクライアントは別のマシンであるので，2 つの間で時刻の同期を取る必要がある.

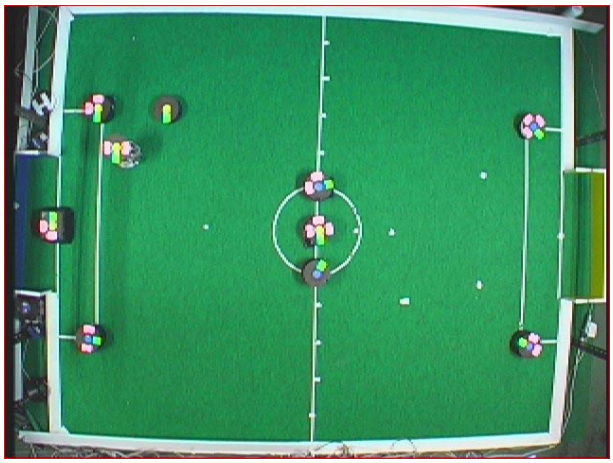


Figure 7: Example of ceiling camera image

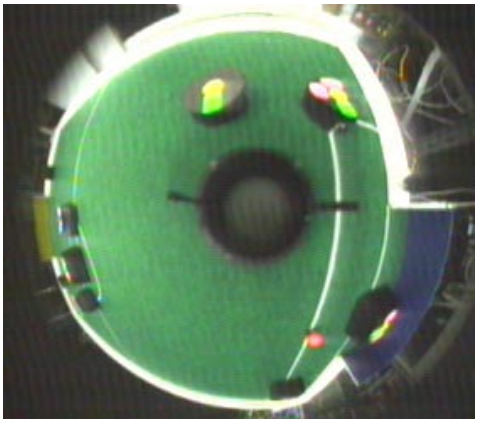
Linux のシステムクロックと NTP を使う方法では，この目的のためには不十分であることがわかった．そこで，clockspeed[Bernstein, -]を参考にして，Pentium CPU の RDTSC を使った時間計測と時間同期の方法を開発した．ビジョンサーバは，時間サーバの役割もしている．クライアントは，接続前にビジョンサーバと時刻の同期を取る．この方法によって，1ms 以下の精度で時刻が計測できるようになった．これを使って，システムの性能評価や伝達遅れを考慮した制御を行なう．

4 評価

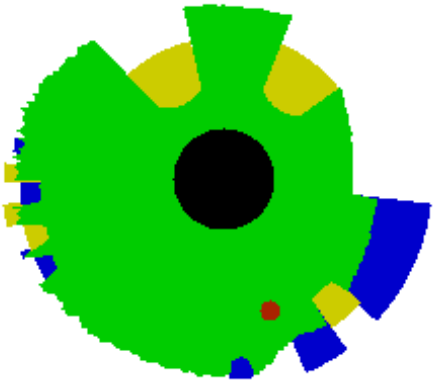
我々はこのようなシステムを Linux 上に実装した．実験ではビジョンサーバは CPU が PentiumIV 2GHz の PC で，BT878 を使ったビデオ取り込みボードを用いて天井カメラの画像を解像度 640×480 で処理している．このサーバに 10 台のクライアントを接続する．クライアントとなる PC は様々で，1 台の PC 上で 2 つ以上のクライアントプログラムが動作している場合もある．ここで評価データを取得したクライアントは，CPU が Pentium III 500MHz の PC 上で 1 つだけで動作している．サーバ PC とクライアント PC は 100BaseTX のイーサネットで接続している．

4.1 処理例

ビジョンサーバが処理する天井カメラの画像の一例を Figure 7 に示す．これは，Figure 4 の配置に対応している．この画像をもとに計算した robot A の最近接エッジデータが Figure 5 である．さらに，これを全方位画像の座標系上に描いたのが Figure 8 の下である．このロボットには，全方位カメラも搭載しており，その画像は Figure 8 の上である．両者を比較すると，疑似ローカルビジョン情報では，最近接エッジより遠い物体は全て隠されている．しかし，それ以外は，実際の画像に含まれる多くの情報を表現できていることがわかる．なお，Figure 8 の下の画像は，人間の理解を助けるものであって，実際の処理に使っているわけではない．



Real on-board camera image



Made from pseudo-local vision information

Figure 8: Examples of on-board camera image

4.2 処理周期，伝達遅れ

サーバでは画像の取り込みと処理を並行して行っており，画像から 10 台のロボットと 1 つのボールの位置姿勢を算出して 10 台のクライアント毎にデータを変換して送り出す処理を 33ms の周期で繰り返すことができている．

Figure 9 に 1 フレーム分の画像がメモリに格納されてから情報が抽出されるまでの所要時間を時間推移を示す．擬似ローカルビジョンの所要時間の内訳である A,B,C は Figure 6 に示した処理の流れに相当する部分を意味する．多少ばらつきはあるものの，合計の所要時間は約 10ms である．撮像されてからの経過時間はさらにフレームキャプチャに必要な時間 33ms が加わる．

我々の搭載カメラ用のプログラムを Pentium III 500MHz の PC 上実行した場合，カメラ画像を 320×240 で取り込んでから情報が抽出されるまでの所要時間は約 23ms であった．サーバに高速な PC を導入することで，10 台分の処理をさせても，我々の搭載カメラシステムよりもむしろ少ない所要時間で処理が行われている．

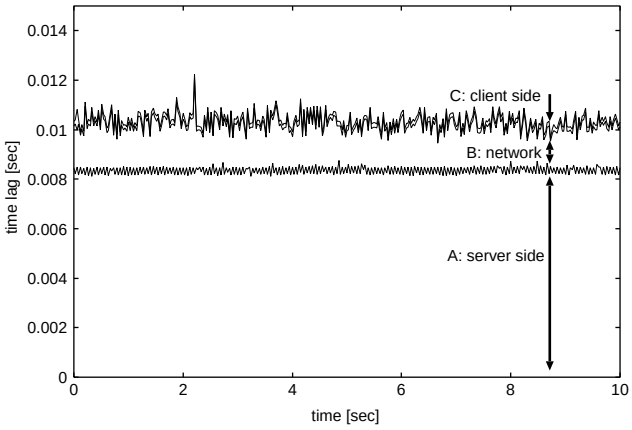


Figure 9: Time delay in vision processing

4.3 解像度，ボールの隠れ

解像度 640×480 である天井カメラを用いた場合，フィールド全域において分解能はほぼ一様で，1pixel は 6mm×6mm にあたる．一方，搭載カメラを用いた場合，我々のロボットの全方位画像では解像度 320×240 であり，最も解像度が高くなったとき，1pixel は 4.4mm×4.4mm にあたり，ロボットからの距離が 220mm となったとき天井カメラと分解能がほぼ一致し，それよりも遠くなると搭載カメラの分解能が劣る．このデータは一例であり，カメラの解像度や視野角によって天井カメラの分解能がより不足する場合も考えられる．さらに，天井カメラを用いた場合，ロボット近傍のボールがロボットの影に隠れることで見えなくなる場合があり，現時点では，近傍はむしろ見やすいという搭載カメラの特性を完全には摸擬できてはいない．この問題を解消するために，複数の天井カメラを用いることが今後の課題である．

5 おわりに

本論文では，マルチロボットプラットフォームのために，天井カメラの情報を用いて搭載カメラの情報を疑似的に生成する簡便な方法を提案した．模擬される搭載カメラの情報は簡単なものであるが，隠れや重なりなど，移動ロボットの搭載カメラ画像における本質的な問題をいくつも表現している．できあがったシステムを評価したところ，処理の周期や伝達遅れは十分に小さく，十分実用になることがわかった．また，我々は，提案したシステムの初期のバージョンで RoboCup2002 に参加した．

近いうちにサーバのソースコードを公開するつもりである．また，この枠組みを使って,「疑似ローカルビジョンリーグ」を実現したいと考えている．これは，RoboCup におけるシミュレーションリーグと実機リーグの橋渡しにもなるはずである．

参考文献

[Bernstein, -] D. J. Bernstein: clockspeed, `http://cr.yp.to/clockspeed.html`

[Bruce, 2000] J. Bruce et al.: Fast and Inexpensive Color Image Segmentation for Interactive Robots, Proceedings of the 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS2000), pp. 2061–2066, 2000.

[Masutani, 2003] Y. Masutani et al.: Team OMNI in 2002 — Pseudo Local Vision Approach —, RoboCup 2002: Robot Soccer World Cup VI, Springer 2003.

[Sekimori, 2002a] D. Sekimori et al: High-Speed Obstacle Avoidance and Self-Localization for Mobile Robots Based on Omni-Directional Imaging of Floor Region, Robocup 2001: Robot Soccer World Cup V, pp. 204–213, Springer, 2002.

[Sekimori, 2002b] D. Sekimori et al.: The Team Description of the Team OMNI, Robocup 2001: Robot Soccer World Cup V, pp. 599–602, Springer, 2002.

[SoccerServer, -] SoccerServer web site `http://sserver.sourceforge.net/`

[田中, 2002a] 田中 志尚 他: ロボカップ小型リーグの天井カメラ画像を用いた疑似ローカルビジョンシステムの開発, 日本機械学会ロボティクスメカトロニクス講演会'02 (ROBOMEC'02) 講演論文集, 1P1-C08, 2002.

[田中, 2002b] 田中 志尚 他: ロボカップ小型リーグ用の疑似ローカルビジョンシステムの評価，第 3 回計測自動制御学会システムインテグレーション部門講演会 (SI2002) 論文集, Vol. II, pp. 183–184, 2002.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-4 (5/2)

2 台の非同期カメラによる仮想高速ビジョンシステムの実現

Pseudo High Speed Vision System with Two Unsynchronized Cameras

清水彰一 † **西 貴行** † **藤吉弘亘** † **長坂保典** † **高橋友一** ††

Shoichi Shimizu, Takayuki Nishi, Hironobu Fujiyoshi, Yasunori Nagasaka, and Tomoichi Takahashi

† 中部大学, †† 名城大学

Chubu University, Meijo University

shiyou@vision.cs.chubu.ac.jp

Abstract

In order to control robots precisely and rapidly, an expensive vision system with 60 fps has been used. In this paper, we present a vision system with two normal unsynchronized cameras, that performs as fast as the expensive vision system. Frame grabbers for each camera are installed on a PC. Images from the camera are processed at 30 fps. Our two camera vision system processes robots positions as fast as the expensive vision system with 60 fps and gives better accuracy in position prediction than a single vision system.

1 はじめに

ロボットが行動するためには，視覚センサによる環境認識・理解技術による情報を基に行動プランニングを行いロボットを制御するビジュアルフィードバック機構が必要である．ロボットを迅速かつ正確に制御するためには，このビジュアルフィードバックのサイクルを高速に行う必要があり，特に，カメラから得られる画像から有用な情報を得るための高速なビジョンシステムが不可欠となる．その際には，ロボット等の対象とする物体の位置をより正確に検出する必要がある．

これまでにビジョンシステムの高速化として，1) 画像処理のハードウェア化による手法[Akita, 2000]，2) 高速な画像処理アルゴリズムの提案[Bruce, 2000]，3)60fps の高速カメラを用いる手法[D'Andrea, 2002, 加藤, 2003] が提案されている．1)，2) の手法は，カラー解析や領域検出等の画像処理アルゴリズムをハードウェア化や最適化することによる処理の高速化である．これらは，ビジュアルフィードバックにおけるビジョン処理にかかる時間を短かくするものであり，30fps のカメラを用いた場合それ以上のサイクルの高速化は期待できない．3) の手法は，60fps のカメラを用いたダブルバッファリングによる高速処理を行うものである．この場合，カメラと専用のフレームグラバが高価であるという問題がある．

本報告では，通常のカメラ (30fps) を複数用いることにより，ビジュアルフィードバックのサイクルを高速化することを目的とする．このとき，複数のカメラを効果的に配置することで，オクルージョン領域の減少，画像の高解像度化等に対して大きな効果がある．このような複数カメラ群のシステムの取り組みとして分散協調視覚プロジェクトが挙げられる[松山, 1998]．中でも分散協調型対象追跡システムでは，多数の自律したエージェント間のコミュニケーションにより対象の追跡を実現している[中澤, 2001]．しかし，これらの多くは処理サイクルの高速化についてはまだ検討されていない．

我々は，分散協調型ロボットビジョンシステムの一アプローチとして，通常の非同期カメラ (30fps)2 台を用いた仮想高速ビジョンシステムについて提案する．2 台のカメラは，1 台の PC に接続されており，それぞれ 30fps で処理される．それぞれのカメラの処理結果を統合することで擬似的に 30fps 以上の結果を得ることが可能となる．

本報告では，2 台の非同期カメラによる高速ビジョンシステムを提案し，物体の運動を予測する実験により，対象物の位置を高速カメラと同等のスピードで得ることができ，その位置の精度は 1 台のビジョンシステムよりも良いことを述べる．

2 2 台の非同期カメラによるビジョンシステム

高速なビジュアルフィードバック機構を必要とするタスクとして，RoboCup 小型リーグにおけるグローバルビジョンシステムが挙げられる[RoboCup, 2003]．グローバルビジョンシステムは，サッカーフィールド上の設置されたカメラ映像からロボットの位置，姿勢，ID とボールの位置を抽出し行動を決定するものである．本研究では，RoboCup

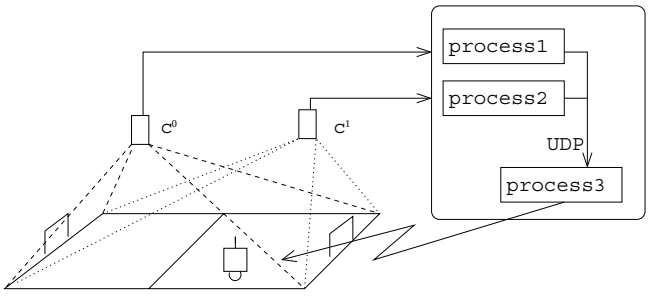
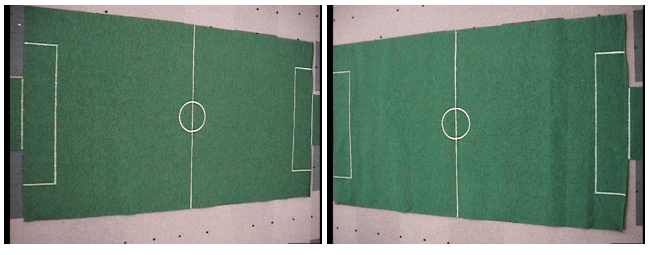


Figure 1: ビジョンシステムの概略



(a) c^0からの映像　　(b) c^1からの映像

Figure 2: 2 台のカメラ画像

小型リーグのグローバルビジョンシステムにおけるボールの運動復元を対象とする．

2 台の非同期カメラ c^0 と c^1 を用いたビジョンシステムの概略を Figure 1 に示す．各々のカメラは，両ゴールの上部高さ 2,800[mm] にサッカーフィールド全体が視野に入るように固定されている．サッカーフィールドの大きさは，縦横約 2,000 × 3,000[mm] である．カメラからの映像信号は，1 台の PC にインストールされた 2 枚のフレームグラバに入力され，それぞれの画像が取り込まれる．Figure 2 に各々のカメラ画像を示す．本ビジョンシステムの処理は，各カメラ画像内のオレンジ色のボール領域を検出し，その中心に対応するフィールド上の位置を求める．

2.1　位置推定

ロボットを制御するシステムにおいて，その制御領域は 3 次元である．画像処理の結果から得られたデータは 2 次元であり，その 2 次元情報からロボット等の，3 次元情報を求める必要がある．これにより，行動計画を 3 次元で考えることができるため，制御システムを実空間に合わせて容易に作成することができる．

RoboCup 小型リーグでは，ロボットやボール位置を推定する手法に，カメラパラメータを用いる手法や，ホモグラフィを用いる手法[Ryad, 2001]が提案されている．カメラパラメータを用いる手法では，ピンホールカメラに基づく幾何学キャリブレーションを必要とする．単一のカメラに関する内部・外部パラメータを推定する手法として Tsai のキャリブレーション手法[Tsai, 1987]が多く用いられている．

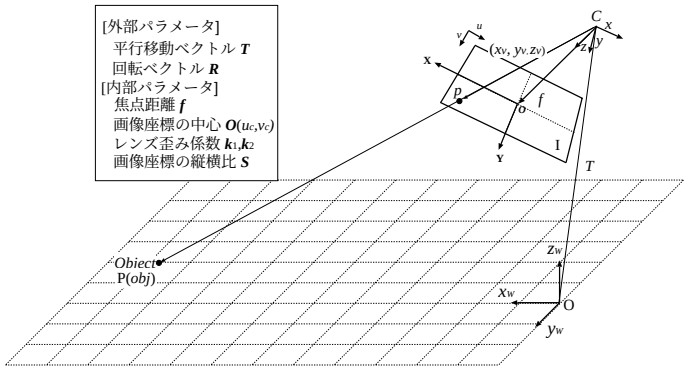


Figure 3: 視線交差によるフィールド上の位置推定

本システムでは，Tsai の手法で求めたカメラの内部・外部パラメータから，3 次元空間の光線情報に基づく位置推定を行う．Figure 3 に示すようなカメラ中心とカメラ画像上の座標を通る世界座標空間における直線を求め，その直線がフィールドの面に交差する点を対象物の位置として計算する．これにより，カメラ画像の座標 (u, v) に対応したフィールド上の世界座標 (X, Y, Z) を求めることが可能である．このとき我々のシステムでは，世界座標 (x_w, y_x, z_w) とカメラ座標 (x, y, z) の関係は次式で表される．R は 3 行 3 列の回転行列，T は平行移動ベクトル (T_x, T_y, T_z) である．

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R(\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} - T) \tag{1}$$

以下に画像座標 (u, v) より，世界座標 (x_w, y_w, z_w) を求める手法を示す．

Step1 画面平面 $X - Y$ 上の点 (X_d, Y_d) を次式から求める．(u_c, v_c) は画像座標の中心である．

$$X_d = u_c - u \quad , \quad Y_d = v_c - v \tag{2}$$

Step2 レンズ歪みを修正した点 (X_u, Y_u) を次式から求める．r は画像中心からの距離であり，S は画像座標の縦横比，k_1, k_2 はレンズ歪み係数である．

$$\begin{aligned} X_u &= SX_d(1 + k_1 r^2 + k_2 r^4), \\ Y_u &= Y_d(1 + k_1 r^2 + k_2 r^4) \end{aligned} \tag{3}$$

Step3 Figure 3 において，カメラ座標の原点から画像上の任意点 p を通過する視線ベクトル (x_v, y_v, z_v) は次式となる．f はカメラ座標における焦点距離とする．

$$\begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = R^T \cdot \begin{bmatrix} X_u \\ Y_u \\ f \end{bmatrix} \tag{4}$$

Step4 視線ベクトルを 3 次元空間上の直線式で表すと，

次式となる．

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \alpha \cdot \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} + T \tag{5}$$

Step5 ここでフィールド面が $x_w - y_w$ 平面に平行であり，$z_w = 0$ であると仮定すると，直線が平面と交わるときの α を決定できる．

$$\alpha = -\frac{T_z}{z_v} \tag{6}$$

Step6 α の値を式 (5) に代入することで，画像座標点 (u, v) に対応した世界座標点 $(x_w, y_w, 0)$ を求めることができる．

$$x_w = -\frac{T_z}{z_v}x_v + T_x \quad , \quad y_w = -\frac{T_z}{z_v}y_v + T_y \tag{7}$$

本手法では，Step5 における z_w の値として，ロボットやボールの高さを与えることにより，高さの異なる各対象物体の 3 次元位置を正確に求めることができる．

2.2 位置推定の誤差

前節の手法でカメラ校正時に用いたフィールド上の特徴点 ($c^0 = 98$ 点，$c^1 = 95$ 点) の位置を推定した結果，その位置推定誤差は平均 4.0[mm]，最大誤差は 17.8[mm] であった．フィールド上のオブジェクト obj の位置をカメラ c^i の画像を用いて算出したものを，$\mathbf{p}^i(obj) = (x^i(obj), y^i(obj))$ とする．ボールを片方のゴールからもう一方のゴールまで約 3.25[m/s] の速度で移動した際の軌跡を Figure 4 に示す．Figure 4 の横軸と縦軸はフィールドのサイドラインを示す．白丸はカメラ 0，黒丸はカメラ 1 より検出されたボールの位置を推定した結果である．カメラ校正時に用いたフィールド上の特徴点における位置推定誤差を求めることはできるが，それ以外の点に関しては対応する世界座標の真値が分からないため，誤差を求めることができない．本報告では，Figure 5 のように，画像座標 (u, v) が 4 近傍に 1 画素ずれた際の対応するフィールド上の世界座標点 (x_w, y_w) 間のユークリッド距離を推定誤差とする．

Figure 4 における三角形は，各カメラにおいてボールの位置を推定した際の誤差 $\mathbf{e}^i(obj) = (e^i_x(obj), e^i_y(obj))$ を表している．位置推定誤差は，3.5[mm] から 6[mm] の範囲で各カメラから離れるほど増加していることが分かる．これにより，本システムでは，あらかじめ各カメラの画像座標における世界座標での誤差を計算できるため，位置に応じた誤差を行動プラニングの際に不確定要素の大きさとして扱うことが可能である．

2.3 2 台のカメラシステムの特徴

2 台のカメラを用いたビジョンシステムの特徴は，(a) オクルージョン領域の減少，(b) 各々カメラの位置推定結果を合成することによる誤差の減少，(c) ビジョンの高速化の三つが挙げられる．

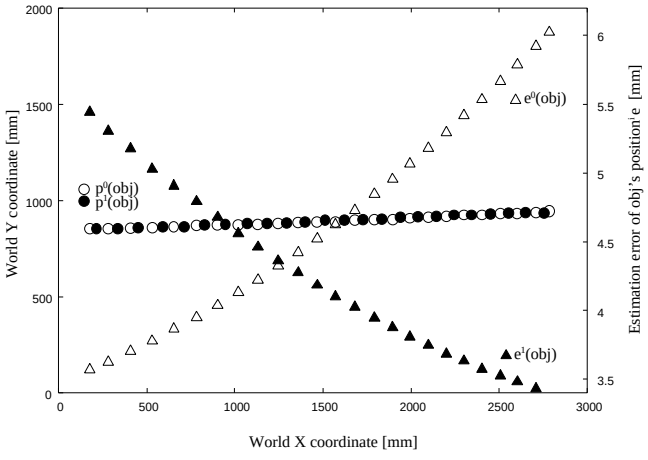


Figure 4: ボールの移動軌跡と位置推定誤差

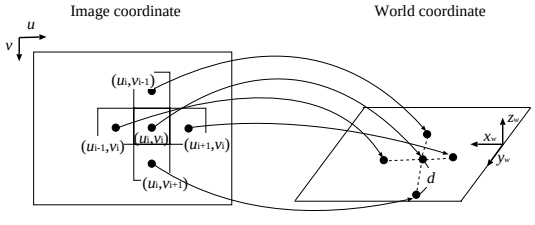


Figure 5: 推定誤差の計算法

最初の特徴は，同じ領域を異なる方向から複数のカメラで捉えることで，オクルージョン領域を減らすことが可能となる．

2 台のカメラを用いて位置推定した際の誤差がガウス分布であり，互いに相関がないと仮定すると，静止した対象物体 obj の位置 $\mathbf{p}(obj)$ とその誤差 e は次式により計算される．

$$\begin{aligned} \mathbf{p}(obj) &= \frac{w_0}{w_0 + w_1}\mathbf{p}^0(obj) + \frac{w_1}{w_0 + w_1}\mathbf{p}^1(obj), & (8) \\ e^2 &= (\frac{w_0}{w_0 + w_1})^2 * {e^0}^2 + (\frac{w_1}{w_0 + w_1})^2 * {e^1}^2, & (9) \\ w_i &= 1/{e^i}^2 \end{aligned}$$

合成した誤差 e は，それぞれのカメラの位置推定誤差 e^0 と e^1 より小さくなる．Figure 6 は，同時刻における各カメラのボール位置推定誤差を式 (9) により合成した結果を示す．

上記の二つの特徴は複数のカメラを用いた特徴としてよく知られているものである．本報告では，三つ目の特徴であるビジョン処理の高速化に関する手法について提案する．通常のカメラシステムでは 30fps で画像を取り込むため，ビジョン処理された結果は 30fps を超えることはない．

そこで，本システムでは 2 台のカメラの結果を統合することでビジョンの高速化を行う．本システムは，2 枚のフレームグラバを 1 台の PC 上に挿入したものでカメラ

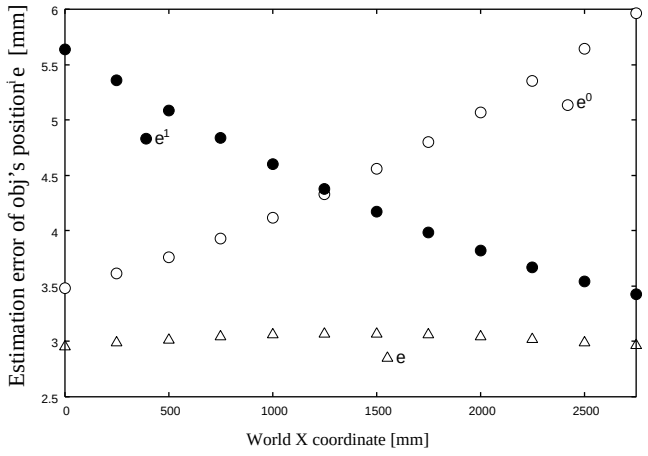


Figure 6: 合成誤差

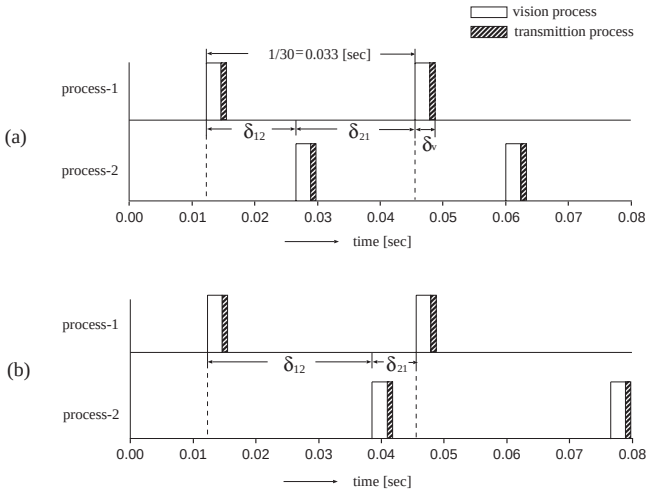


Figure 7: process-1process-2 のタイミングチャート

間の同期機構を持たない．Figure 4 におけるカメラ 0 による結果 (白丸) とカメラ 1 の結果 (黒丸) は交互にプロットされており，これは 1 台のカメラシステムよりも早い周期でボールの等速直線運動を捉えていることを意味する．特別なカメラ同期機構を持たないでより高速な処理結果を得ることが，2 台のカメラを用いる三つ目の特徴であり，本報告で提案する仮想高速ビジョンシステムの特徴である．

3 高速な動き予測

2 台のカメラは外部同期を使用しない場合，それぞれのキャプチャタイミングは異なる．本報告では，この異なりにおける 2 台の非同期カメラによる仮想高速ビジョンシステムの特徴について検討する．

3.1 位置データの統合

Figure 1 における process-1 と process-2 はそれぞれのカメラからの画像を 1/30 秒毎に処理する．各プロセスは，カメラ c^i の処理結果である物体の位置 $\mathbf{p}^i(obj)$ と，その処理した画像のキャプチャ終了時刻を UDP 通信により process-3 に送信する．process-1 と process-2 は，1 台の PC 上で実装されているため，それぞれのプロセスからキャプチャ終了時に参照する時刻のずれは生じない．process-3 では，process-1 と process-2 から送信された結果をキャプチャ終了時刻で整合し，その情報を基にロボットを制御する．

2 台のカメラのキャプチャタイミングが同じである場合，Figure 4 におけるカメラ 0 による結果 (白丸) とカメラ 1 の結果 (黒丸) は重なる．process-1 と process-2 の時間間隔が通常のビデオレートの半分 (1/60 秒) の場合，process-3 は 1/60 秒毎に位置データを交互に受け取ることができる．

Figure 7 に process-1 と process-2 のタイミングチャートを示す．ここで，δ_{12} を process-1 から process-2 までの時間間隔，δ_{21} を process-2 から次の process-1 までとする．Figure 7 における (a) の場合は，2 台のカメラのキャプチャタイミングがほぼ等間隔 ($\delta_{12} \simeq \delta_{21} \simeq \frac{1}{60}$) に異なった例である．(b) の場合は，キャプチャタイミングが近い例 ($|\delta_{12} - \delta_{21}| \simeq \frac{1}{30}$) である．本システムは，Dual processor Xeon の PC 上に実装され，ビジョンプロセスに 0.0026 秒，UDP 通信に 0.00018 秒を費やしている．

本システムでは，δ_{12} と δ_{21} を等間隔 ($\delta_{12} = \delta_{21}$) にするためのカメラの外部同期機構を持たない．従って，タイミングはそれぞれのカメラのシャッタータイミングに依存している．本システムでは，画像のキャプチャとビジョンプロセス δ_v が δ_{12} と δ_{21} よりも短い時間で処理を行う必要がある．この処理時間が満足されないと，二つのカメラシャッターは同期した状態に近くなり，2.3 で述べた特徴 (b) の合成による位置精度の向上のみに使用されることになる．

3.2 線形予測による動き推定

ロボットを正確に制御するためには，対象物の動きを予測して制御する必要がある．得られた最新のボール位置から次の予測位置 $\mathbf{p}_{t+1}(obj)$ を求める式を以下に示す．次式は，カメラ 0 の画像から計算された位置が最新の場合である．

$$\begin{aligned} \mathbf{p}_{t+1}(obj) &= \mathbf{p}_t^0(obj) + k(\delta_{12})\mathbf{v}_t(obj), \\ \mathbf{v}_t(obj) &= (\mathbf{p}_t^0(obj) - \mathbf{p}_{t-1}^1(obj))/\delta_{21} \end{aligned} \tag{10}$$

予測位置 $\mathbf{p}_{t+1}(obj)$ は，現在と一つ前の結果より物体 obj の速度 $\mathbf{v}_t(obj)$ を求め，次の位置を線形に予測するもである．カメラ 1 の結果が最新の場合は，上記の式の添字が入れ替わる．

3.3 実験

等速円運動における動き予測の実験として，Figure 8 に示すようにターンテーブルとボールを用いた．長さ 1000[mm]

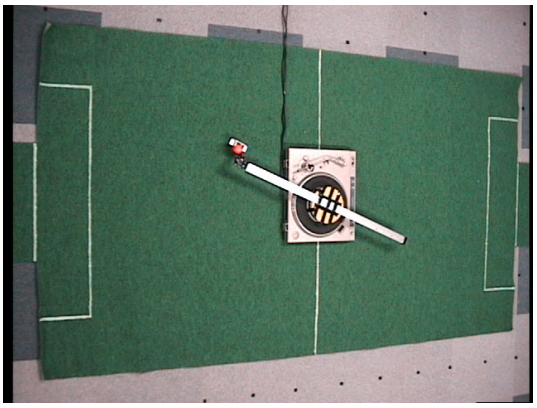


Figure 8: ターンテーブルを用いた実験の様子

の物差しの端にボールを固定し，ターンテーブルを回転させることで半径 500[mm] の等速円運動となる．ターンテーブルの回転数は 45[rpm] であるため，ボールの回転速度は $\frac{45}{60}2\pi = 0.478$[rad/s]，また，速度は 1023.7[mm/s] となる．

3.3.1 等速円運動の復元

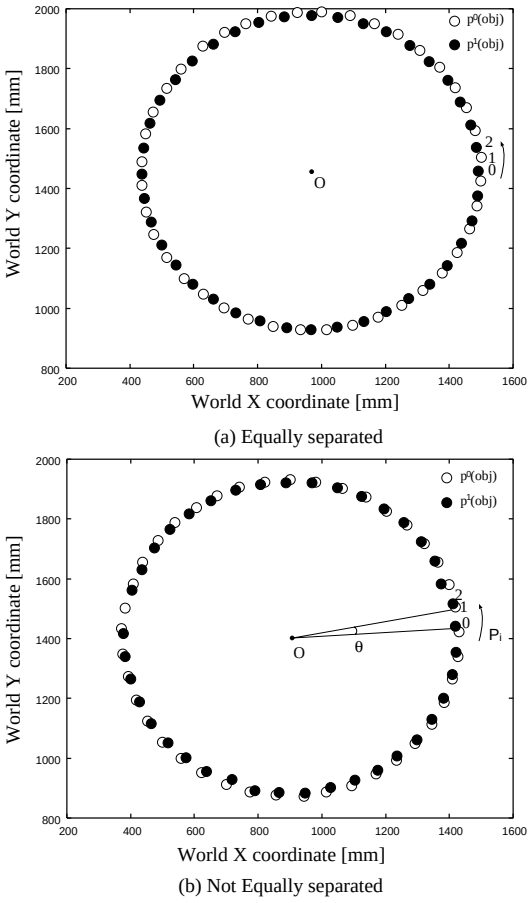
Figure 9 にボールの等速円運動における検出した結果の軌跡を示す．各プロット点は，時系列の順に 0, 1, ... とラベル付けされている．白丸 (奇数) はカメラ 0 のボールの位置推定結果，黒丸 (偶数) はカメラ 1 の結果を示す．

60fps の高速カメラを用いたビジョンシステムの場合，ボールが一周するには 1.33 秒かかるため，$1.33 \times 60 \approx 80$ 点のデータが得られることになる．Figure 9 は一周の実験結果であり，80 個のデータがプロットされている．これより，本ビジョンシステムのサイクルは高速カメラ (60fps) と同等のスピードを実現できていることを示している．

60fps で処理した場合，連続した 2 点 P_i と P_{i+1} の回転角度 $\angle \mathrm{P}_i\mathrm{OP}_{i+1}$ の理論値は 0.0785[rad] となる．Figure 10 は，回転角度 $\angle \mathrm{P}_i\mathrm{OP}_{i+1}$ の分布である．2 台のカメラの時間間隔が等間隔 ($\delta_{12} \simeq \delta_{21} \simeq \frac{1}{60}$) の場合，回転角度の平均は 0.0783[rad] であり 60fps で処理した際の理論値に近いことが分かる．角速度は，$\angle \mathrm{P}_i\mathrm{OP}_{i+1}/\delta$ で求められる．角速度の平均は，$\delta_{12} \simeq \delta_{21} \simeq \frac{1}{60}$ の場合 4.70[rad/s]，$|\delta_{12} - \delta_{21}| \simeq \frac{1}{30}$ の場合 4.50[rad/s] であった．その際の分散は前者が 1.06，後者が 1.52 であり，間隔が $\delta_{12} \simeq \delta_{21} \simeq \frac{1}{60}$ と等間隔である場合，より精度のよい円運動の復元が可能であるといえる．

3.3.2 動き予測

等速円運動するボールにおける次の予測位置を式 (10) より求めた．その際の予測誤差を Figure 11 に示す．予測誤差は，予測した次の位置と実際に観測された位置とのユークリッド距離である．グラフ中の＋は，2 台のカメラシステムによる予測誤差，*は 1 台のカメラシステムによる予測誤差である．Figure 11 より，1 台のカメラシステムよりも 2 台のカメラシステムの方が平均予測誤差が小さいことが分かる．Table 1 にカメラの時間間隔が等間隔である場合を A，そうでない場合を B としたときの予測誤差の平均と分散を示す．平均は時間間隔に関係なくほぼ同じであった．しかし，分散は等間隔の方が小さく，2 台のカメラシャッターが 1/60 秒ずれているとより精度のよい予測が可能であるといえる．



(a) Equally separated

(b) Not Equally separated

Figure 9: 等速円運動の復元

4 まとめ

本報告では，ロボットを迅速にかつ正確に制御するために，通常の非同期カメラ (30fps)2 台を用いた仮想高速ビジョンシステムについて提案した．2 台のカメラは，1 台の PC に接続されており，それぞれ 30fps で処理される．

Table 1: 予測誤差 [mm]

		two cameras	single camera	
			c0	c1
A	Ave	9.4	23.8	22.8
	Var	23.2	17.0	10.4
B	Ave	10.2	22.2	21.8
	Var	42.3	10.3	36.4

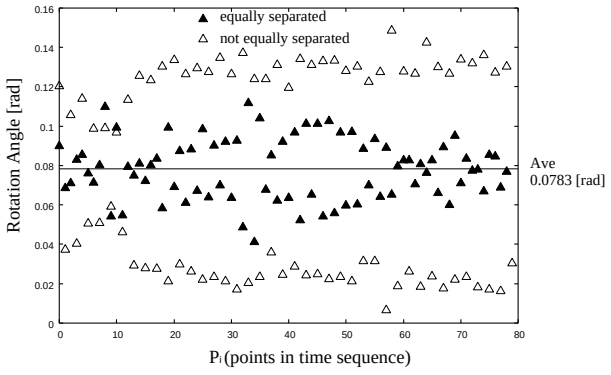


Figure 10: 回転角度の分布

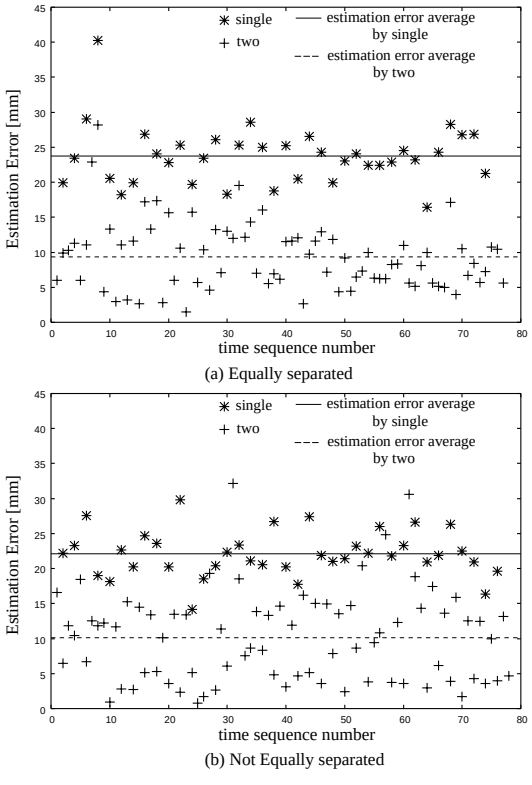


Figure 11: 予測誤差

それぞれの処理結果を融合することで擬似的に 30fps 以上の結果を得ることが可能となる．対象物の次の位置を予測する際に，2 台のカメラシャッターが約 1/60 秒ずれているとより精度のよい予測が可能であることが分かった．

参考文献

[Akita, 2000] J. Akita.: Real-Time Color Detection System Using Custom LSI for High-Speed Machine Vision, in *RoboCup-99: Robot Soccer World Cup III*, Spriner, pp128-135, 2000.

[Bruce, 2000] J. Bruce, T. Balch, M. Veloso: Fast and Inexpensive Color Image Segmentation for Interactive Robots, In Proceedings of IROS-2000, Japan, 2000.

[D'Andrea, 2002] R. D'Andrea, *et al.*: Detailed vision documentation, http://robocup.mae.cornell.edu/

[加藤, 2003] 加藤恭佑，日比野晋也，児玉幸司，村上和人，成瀬正: 情報処理学会研究報告, Vol. CVIM-136, No.16, pp115-122, 2003.

[Ohta, 2001] 太田紘高，遠藤龍矢，山本和彦，加藤邦人，神成敦司: 画像処理と戦略の協調による Robocup 小型システム，システムインテグレーション部門学術講演論文集, pp59-60, 2001.

[Tsai, 1987] R. Y. Tsai: A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses, IEEE Journal of Robotics and Automation, Vol. RA-3, No.4, pp323-344, August 1987.

[Ryad, 2001] R. Benosman, J. Douret, J. Devars: A Simple and Accurate Camera Calibration for the F180 RoboCup League, in *RoboCup-2001: Robot Soccer World Cup V*, Spriner, pp273-280, 2001.

[RoboCup, 2003] RoboCup: http://www.robocup.org/

[松山, 1998] 松山隆司: 分散協調視覚–視覚・行動・コミュニケーションの機能統合による知能の開発，画像の認識・理解シンポジウム (MIRU '98), ppI-343–352, 1998.

[中澤, 2001] 中澤篤志, 加藤博一, 日浦慎作, 井口征士: 分散視覚エージェントを用いた複数人物追跡システム, 情報処理学会論文誌, Vol.42, No. 11, pp2699–2710, 2001.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-5 (5/2)

エンターテイメント向けサッカーログフォーマット 2goOZ 形式の検討

2goOZ : A compression format of soccer game log to see

鹿田和之進、西野順二
Kazunoshin Shikada, Junji Nishino
電気通信大学 システム工学科
Dept. systems engineering, UEC
shika@fs.se.uec.ac.jp, nishino@se.uec.ac.jp

Abstract

Recently, multi agent games such as RoboCup soccer have grown and made great progress so that their games become very close to human games. In this paper, we introduce an compression method on multi agent game logs for XML data format. Combination of linear-predictor(LP) and vector quantization(VQ) shows good performance in both compressing rate and errors for this purpose. This result indicates that teams use cooperative formation play style.

1 はじめに

本論文は、RoboCup サッカーに代表されるマルチエージェントゲームの記録方式について、ユーザによる娯楽的観賞を目的とした、符号化と圧縮表現の方法について検討することを目的とする。

PDA など携帯情報端末の処理速度や記憶容量に制限のある環境下では、提示データの圧縮保存が必要である。いっぽう、サッカーに代表される多人数ゲームのより臨場感のある再生には、鳥瞰やボール周辺だけを自由に表示できるスケーラビリティが必要である。計算機的には、各プレイヤやボールの位置情報を、時間経過にともなってコーディングすれば良いことになるが、その容量は、プレイヤ数とゲーム時間に比例して巨大なものとなる。行動分析など学術目的のための符号化では可逆性が不可欠だが、人による観賞を目的とすればそれに応じた非可逆的な圧縮の可能性がある。

一般に、あるデータに対して最適な情報圧縮の手法は目的に応じて異なるものである。可逆なデータ圧縮と、静止画像、動画像および音声に関する非可逆圧縮については、市場の要求から広範な研究がなされており [2, 3]、ほぼ完了していると言っても過言ではない。

いっぽうでサッカーのような多人数ゲームの行動記録について、効率的に圧縮する方法についての検討はほとんどみられない。

本研究は、多人数ゲームの行動記録を圧縮に基づいた XML フォーマットで記録することを目指している。これまでに人による観賞を主たる目的とした圧縮について、基礎的な圧縮方式について検討し [4] 線形予測とベクトル量子化をあわせた方法を提案した。本論文ではこの方式について検討する。

2 Soccer Log の観賞

2.1 ログの質の向上

RoboCup サッカーシミュレーションリーグ [1] では、11 対 11 の人工プレイヤがそれぞれ個別のプログラムで作成され、ネットワークを介してバーチャルサッカー場であるサッカーサーバに接続し、サッカーの試合を行う。このときボールと各プレイヤの行動、すなわち試合の全体像はサッカーサーバによってログとして記録されている。

近年、人工プレイヤの行動判断のレベルが上昇しており、試合のログを見ているかぎりでは、人間の行っているサッカープレイとみまごうばかりと言われている [5]。単純にサッカーの試合のリプレイとしての観賞にも耐えるものとなっている。

2.2 XML による注釈付きログ

さらに、より高度な戦略を目指し、システム、アルゴリズムの更新を行なうため、チェスプログラムの開発同様、サッカーの専門家による評価のフィードバックも有効と考えられる。このとき評価の記述と実際の行動を時間軸に沿って対応させ記録する必要がある。

XML など構造化された記述形式を援用することが望ましいが、行動記録のサイズがあまり大きくては、実装にお

```
<?xml version="1.0"?>
<RCG SOME-Properties>
..
<2goOZCodeBook>
<Code name="c00">..vector</Code>
<Code name="c01">..vector</Code>
..
</2goOZCodeBook>

<ShowInfo time="1">
  <2goOZCode>..Code(c01)..</2goOZCode>
  <Comment>
    Comments at this time
    by human
  </Comment>
</ShowInfo>
<ShowInfo time="2">
..
</RCG>
```

Figure 1: XML ログフォーマット (一部)

いて非現実的となる。PDA の記憶サイズ (10M 程度) で数試合分の試合ログと、コメントログを同時に保存できることが目標である。

RoboCup サッカーログに関しては、公式配布されているサッカーサーバ rcssserver Ver. 9.* に、XML フォーマットログ生成機能が実装されている。これは 2003 年 4 月の段階では、実験的なものである。ログとして保存された全ての情報を圧縮無く時系列に保存している。このため 1 ステップあたり 10Kbyte 程度の文字量で、6000 ステップの試合全体では 60Mbyte、gzip による単純圧縮によっても 5.5Mbyte の容量となる。

コメントの追加を想定すると、圧縮済データの容量として、1 ゲームあたり 10〜100Kbyte(0.1Mbyte) 程度が望ましい。

本稿で検討するベクトル量子化に基づいた圧縮データは、コードブックとコード本体からなる。コード本体の生成を各時間ステップ毎に行なうことで、XML データ中での時間とコメントの対応が可能となる。

以下では、図 1 に示したような、rcssserver の XML ログを改変したフォーマットを想定する。

3 サッカーとしての圧縮

ここでは、対象となるサッカーの試合が持つ情報量について、そのモデル化と見積もりを行なう。

3.1 サッカー試合情報のモデル化

サッカーの試合内容を再現する情報は、全てのプレイヤとボールの動きが主体である。このほかに、プレイヤの手や姿勢、首の向き、表情、といった詳細情報や、風の状態、芝の状態、照明、天候の状態といった外乱要因情報もある。しかしこれらは、ボールを取り合うゲームとして見たサッカー試合の情報の再生においては、さして重要ではない。

プレイヤは試合開始 $t = 0$ から終了時刻 $t = T$ までの間、一連の移動を行なっており、これがゲームの本質である。よってプレイヤ i の試合全体での行動は、時間をパラメータとする平面座標の関数 $P_i(t) = (x_i(t), y_i(t))$ で表される。22 人のプレイヤとボールも同様に含め次の 23 点さらに x,y 軸に分け、46 次元のベクトル関数となる。

$$P(t) = (P_0(t) \cdots P_{22}(t))$$

ただし、$0 \leq t \leq T$、$i = 0$ はボール、$i = 1$〜11 はチーム L のプレイヤ、$i = 12$〜22 はチーム R のプレイヤを表す。

一般に $P(t)$ は連続な関数となるが、実際上は、適当な Δ t でサンプリングし、センサシステムによる量子化が行なわれる。ここでも $P(t)$ は離散化された時系列データとして扱う。

3.2 基礎的なサイズ見積

RoboCup 公式サッカーサーバが生成するサッカーシミュレーションの行動ログには、管理情報など行動情報以外のデータが含まれており、おおむね 10Mbyte〜30Mbyte というサイズである。これは不要に大きい。

ここでは、再生に必要な情報に限定したうえで、必要容量の見積りを行なう。

RoboCup サッカーは 2 チーム合計 22 プレイヤエージェントにより行なわれる。試合時間は 600 秒間で、サンプリングタイムは 0.1 秒となっており、ロスタイムを除いて試合全体は 6000 ステップとなる。人間サイズにモデル化されたエージェントとサッカーフィールドのサイズから、その位置情報を表現するに必要なビット数を考える。観賞に必要な精度としては、0.1m 刻み程度の解像度で十分である。フィールドサイズは 100m に収まるので、位置情報は $log_2(100/0.1) \approx 10$ bit となる。

以上の仮定から、基本的なデータサイズを見積もると次のようになる。

$$size = (10 + 10) \times 23 \times 6000 = 345KB.$$

3.3 圧縮システム

本論文で扱うサッカー試合のデータの主たる用途は、人間が観賞することにある。このため、ユーザの感性にとって違和感のない範囲での、非可逆な圧縮が可能な対象である。

Table 1: 手法の比較

手法	サイズ	圧縮率	平均誤差	最大誤差	備考
オリジナルデータ	1467 KB	0 %	0	0	
gzip のみ	361 KB	75 %	0	0	
１次線形予測	205 KB	86 %	0	0	可逆圧縮
再量子化	134 KB	91 %	0.24	0.5	平均 0.5m の誤差
ベクトル量子化	58 KB	96 %	4.69	25.9	46 次元の状態を直接コーディング
線形予測＋ベクトル量子化	50 KB	97 %	0.62	2.1	停止点近傍で摂動する

一般に、このような情報源符号化の場合、1) サンプリング、2) 非可逆圧縮 (量子化)、3) 可逆圧縮を順に行なう。

可逆圧縮はハフマン符号化やLZ アルゴリズムに代表される、lossless 圧縮である。すでに原理上改善の余地が無いほど優秀な手法が実用化されている。

そこで、本研究では圧縮システムのうち非可逆な圧縮のみを対象とし、可逆圧縮には一貫して gzip を用いることとする。以降の圧縮データ量は、提案した手法で量子化し作成したテキストデータを、gzip で圧縮したものである。

3.4 LP＋VQ によるログの圧縮

圧縮法として、線形予測とベクトル量子化を組み合わせた方式が、多少の静止位置での誤差を含むものの良好であった [4]。ここでは本手法の基礎となる方式とそれらの比較を行なう。

１次線形予測 (LP) 線形予測モデルを用いて残差情報のみ圧縮する方法で、音声の圧縮等に用いられる。

予測関数を $f(t) = 1.0 \times P(t-1)$ で表される１次線形関数とした。すなわち予測誤差ベクトルは前時刻との差と等しくなり、$E(t) = P(t) - P(t-1)$ となる。この $E(t)$ を後段の可逆圧縮の対象とした。

ベクトル量子化 (VQ) 各時刻での 46 次元位置状態ベクトル、$(P_0(t) \cdots P_{22}(t))$ を、ベクトル量子化してコーディングした。コードブックの生成には、自己組織化マップによる学習を用いた。コードブックとして 16x16 の２次元マップを用いた。

3.5 2goOZ フォーマット：１次予測＋ベクトル量子化

提案した 2goOZ フォーマットは、LP と VQ を融合した方式である。

図 2 にシステム構成を示す。

１次予測残差、$(E_0(t) \cdots E_{45}(t))$ について、ベクトル量子化を行なった。

なお、歪みのある $\hat{E}$ を積算することによる累積誤差を排除するため、各時点の残差を源情報と伸長予測値の残差を量子化したものである。次の式で計算する。

$$\hat{E} = CodeBook(P(t) - \hat{P}(t-1))$$

ただし、伸長予測値は次の式で与える。

$$\hat{P}(t) = \hat{P}(t-1) + \hat{E}(t)$$

ここでの残差ベクトルの分布は、コードブックを作成したときの残差ベクトルの分布と多少異なっており、最適性には疑問が残るものの、現実的に大きな問題はない。

この方式でのサイズは 49.5Kbytes(96.6 %) で、図 3 に示す。

4 性能評価

各手法の圧縮サイズと誤差を表 1 にまとめる。

1 次線形予測モデルを用いた圧縮は、可逆圧縮でありながら、比較的高いパフォーマンスをあげている。gzip が実質的には統計的圧縮を非常に高い精度で行なっているものの、時刻 t-1 との差相関には予測モデルを使わなければ対応できないことを示している。この時点で 200KB 程度であるが、試合に対するコメントが 10〜50KB であることと比較すると、まだ大きすぎると言える。

刻み幅を単純に大きくした再量子化は、アルゴリズムが単純であるという利点がある。いっぽうで量子化刻みが、逆にそのまま誤差に現れる。また、今回の観賞用データの圧縮という目的にはやや圧縮率が低い。

ベクトル量子化 (VQ) は、非可逆圧縮の手法としては古典的であるものの、一般的に行なうには計算時間がかかるという難点がある。そのため通常はベクトルの要素数が 2〜5 程度で用られる。

本研究で用いた、自己組織化マップによって圧縮用コードブックを作成する手法は、46 次元という多次元ベクトルを効率的に扱える方法である。また、その圧縮パフォーマンスも 96 %と非常に高いものとなっている。しかし、6000 点を 256 種の辞書でそのまま表現することで、大きな誤差が発生しており、再生には適さない。

提案した、１次線形予測誤差に対して、VQ を適用する、LP+VQ 方式は、全域で誤差が少なく、平均誤差も小さくなっている。全域では誤差が少ないものの、図 3 中の拡大部に示すように、誤差により停止点近傍での実在しない摂動が発生する現象が問題である。

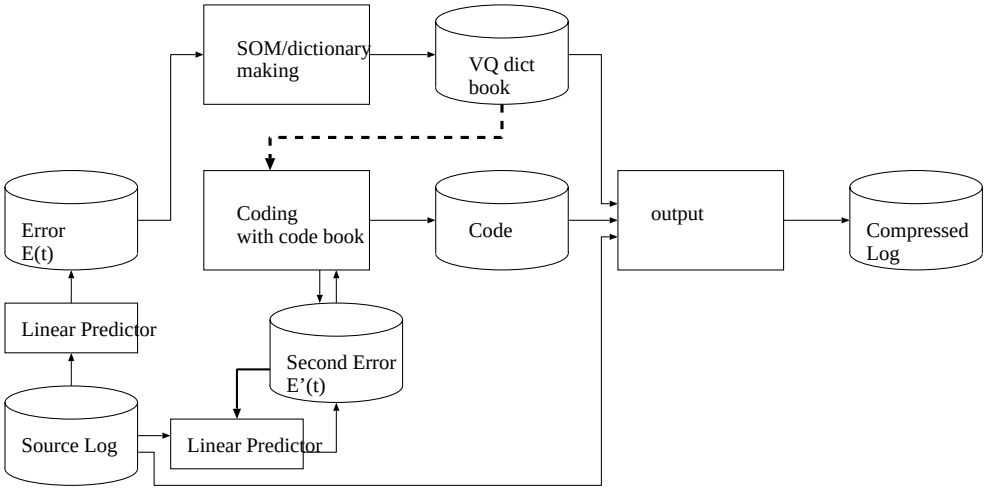


Figure 2: LP+VQ 圧縮システム

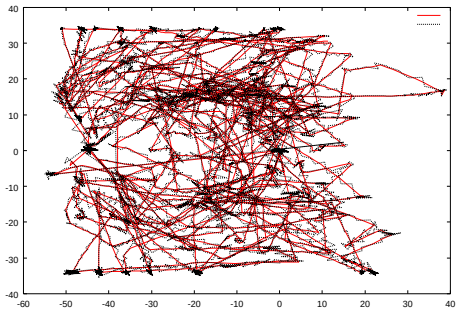
圧縮率と精度のバランスを考慮すると、今回対象としている、サッカー型マルチエージェントゲームの試合ログの圧縮に有効であると言える。

5 おわりに

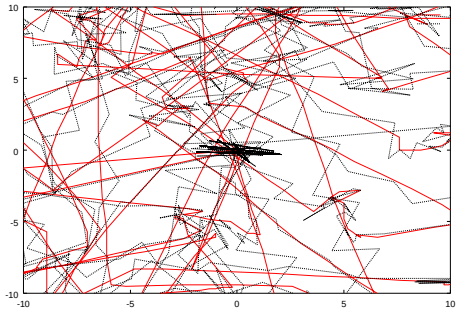
従来は、音響や画像情報にのみ用いられてきた非可逆圧縮手法を、人間の観賞を対象とした、サッカーの試合記録の情報圧縮に適用した。１次線形予測とベクトル量子化の両方を利用した手法が、圧縮度では満足のいくものであった。また、ベクトル量子化の結果各時点でのコードを元に XML フォーマットを構成することができた。いっぽう停止点付近での摂動は、観賞には適さない誤差であるため、その改善は必要である。

サッカーの試合におけるフィールド全体での定型的な動きの冗長度を捉えていると言える。特に、１次線形予測を用いたモデルでの、ベクトル量子化が高圧縮率であったのは、全体での並行移動が多々みられたからにほかならない。

今回は時間方向については、１次線形予測のみの対応であった。チームワークの存在を考えると、時間とプレイヤの両軸に広がったより高次元ベクトルでの情報冗長度が存在すると考えられる。これらのチームワークの存在に基づき、協調行動を表す小さな知的予測モデルが構成できれば、より効率の良い圧縮が可能となる。



(全体)



(停止点付近拡大図)

Figure 3: 予測＋ベクトル量子化

参考文献

[1] A. Birk, S. Coradeschi, and S. Tadokoro, editors. *Robocup 2001 : Robot Soccer World Cup V*, volume 2377 of *LNAI*. Springer, 2002.

[2] 情報理論とその応用学会, editor. **情報源符号化 無歪みデータ圧縮**. 培風館, 1998.

[3] 情報理論とその応用学会, editor. **情報源符号化 歪みのあるデータ圧縮**. 培風館, 2000.

[4] 西野順二. サッカーシミュレーションログデータの圧縮フォーマット. **情報処理学会ゲーム情報学研究報告**, 2002-GI-9(35):7–12, 3 2003.

[5] 野田五十樹. ロボカップシミュレーションリーグ. In **第７回ゲームプログラミングワークショップ講演論文集**, pages 22–27. 情報処理学会, 2002.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-6 (5/2)

人間プレイヤのポジションカバー行動の発現

A discovery of position cover action by human soccer player

戸田英治、西野順二、鹿田和之進、本多中二
Eiji Toda,Junji Nishino,Kazunoshin Shikada,Nakaji Honda
電気通信大学 システム工学科
Dept. of Systems engineering, The University of Electro-Communications
toda@fs.se.uec.ac.jp,nishino@se.ues.ac.jp

Abstract

This paper shows an analysis result of human soccer forward player's behavior in simulated soccer; RoboCup. We developed an interface system called OZRP/Palm-system that enabled human pilots to dive into simulated socder field. Human players could play very well in spite of several constraints such as limited information and noise. We showed a priori cooperation abilities such as dynamic formation reconfiguration by means of statistical indices.

1 はじめに

本論文は、シミュレーションサッカーにおけるバーチャル環境下での、人間の協調的行動について計測、分析することを目的としている。さらに、この観測から得られた協調行動モデルを、協調環境にある人工システムに活かすことを目指している。

分散環境下におかれたロボットやエージェントシステムの協調においては互いの情報伝達が重要な役割を持つ。しかしながら電波や光、音声などの無線通信には、情報の欠落や不完全性、不規則な雑音が含まれ、疎な伝達しか行なえない。人間は、サッカーの試合での協調的プレイを頻繁に行なっている。その情報通路は単音節程度の簡易な音声や、簡単なジェスチャーといった疎な伝達のみで、高度な協調行動を実行している。

RoboCup サッカーシミュレーションに、人間が参加できるようにするユーザインタフェースシステム OZRP システムを用いて、人工チームとのシミュレーションサッカーの試合を行う。これまで、フォワードプレイヤについて統計的な指標に基づき分析した[5]。本論文では、プレイヤ相互のカバー行動について分析する。

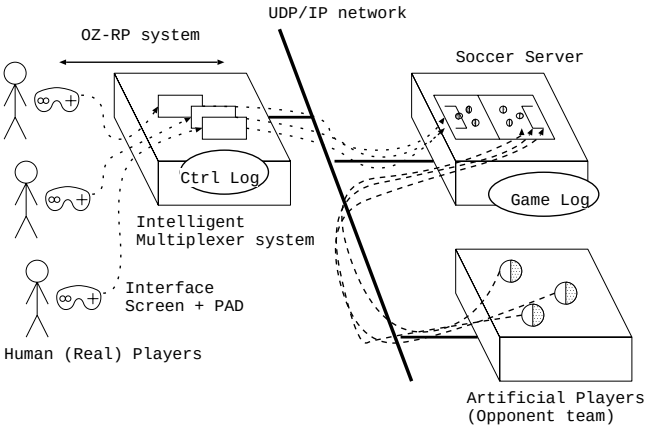


Figure 1: OZ-RP システムの構成：動作サポートは OZRP ローカルサーバで実現する

2 計測の方法

2.1 OZRP システム

OZ-RP[2, 4, 3]は、人工プレイヤどうしでおこなわれる RoboCup サッカーシミュレーションリーグ[1]の試合に、人間が参加するためのシステムであり、すでに様々なバージョンを構築し実験を進めている[6]。

システム構成の概念図を図 1 に、操作端末の例を図 2 にそれぞれ示す。OZRP_Palm システムは、ボール追従、目的位置へのパス、目的位置への移動というマクロレベルの指令を与えることができる端末である。人間に対して得られる情報も、各人が担当するプレイヤの視界に制限され、なおかつノイズを含んだものとなっている。これらの制約を満たすことで、人工プレイヤとフェアな条件で試合ができるように設計したものである。

2.2 実験試合

2002 年 10 月に名古屋工業大学で行なわれた RoboCup 秋のキャンプで対戦実験を行なった。11 プレイヤ中、フォ

Figure 2: OZ-RP_Palm システム：ユーザ端末の一例

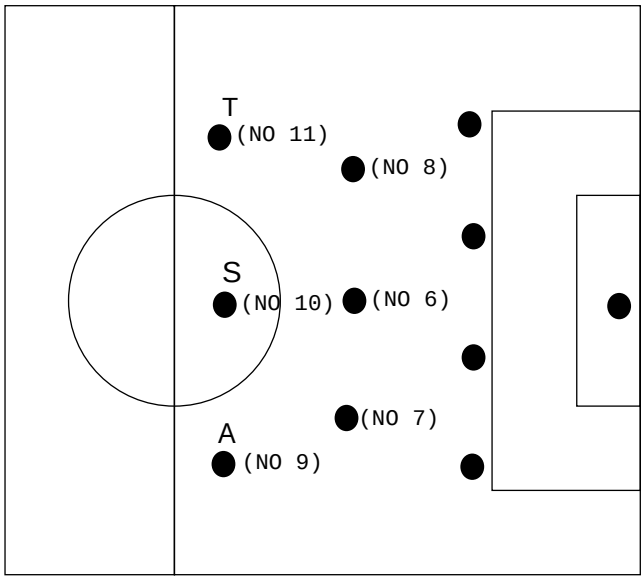


Figure 3: OZ-RP_Palm のフォーメーション

ワード 3 名を人間パイロット (A、S、T と呼ぶことにする) が操作し、残り 8 名を人工プレイヤが補助して 1 チームとした。補助人工プレイヤには、東京工業大学秋山氏作成によるチーム HELIOS のエージェントを採用した。試合中の人間プレイヤの位置を図 3 に示す。

実験中行なった試合から、チーム HELIOS が行なった試合を中心にして分析用にいくつか選択した。これは、本チームの守備に充当されている人工プレイヤ HELIOS との比較をすれば、人間 3 名以外の条件をそろえて比較ができるためである。選択した試合の一覧を表 1 に示す。

3 人間フォワードの行動の特徴

実験した試合結果それぞれに対し、フォワードプレイヤのフォーメーションの適応性を調べた。

Table 1: 分析に用いた試合

RoboCup Autumn Camp 2002			
対戦チーム			結果
OZ-RP_Palm	-	TsinghuAeolus	0-1
OZ-RP_Palm	-	nobisu	16-0
OZ-RP_Palm	-	NITsoccer02	25-0
OZ-RP_Palm	-	NITCalcio2002	6-0
HELIOS	-	TsinghuAeolus	0-1
HELIOS	-	nobisu	38-0
HELIOS	-	NITsoccer02	33-0
HELIOS	-	NITCalcio2002	26-0

3.1 フォーメーションの時間変化

人間行動における適応的変化を対象とするため、平均座標の時間変化を比較する。

一試合 6000 ステップを 500 ステップずつの時間帯セグメントに 12 分割し、おのおのの位置座標時間平均を算出した。図 4 に、人間チームが強いチーム (Thinghua) と戦った試合での、前半冒頭、後半 3 セグメント目の二つの時点でのフォーメーションを示す。

相手のチームが強いチームであることを、時間経過とともに把握し、フォワードプレイヤが 1 名中盤に移動していることがわかる。さらに、その空いたフォワードのスペースを埋めるために、残り 2 名のフォワードプレイヤがバランス良い位置に移動している。

これは互いに無言のまま創発した行動で、音声その他の方法で当座に相談して行なったのではなく、また事前の打ち合せがあったわけでもない。このようなアプリオリな行動シナリオをサッカーを知る人は持っており、暗黙の協調が生まれることを確認できた。また、この変化は常識的な観点で適切なものであったと言える。

同じ相手に対する人工プレイヤのみのチームの試合の様子を図 5 に示す。いずれの時点でも、フォーメーションが変化していないことが分かる。

3.2 人間の適応行動

次に、人間プレイヤがフォーメーションを変化させた様子をより詳しくみるため、一試合 6000 ステップを 100 ステップずつの時間帯セグメントに 60 分割し、おのおのの Y 方向の位置座標時間平均を算出した。さらに、人間プレイヤの Y 方向の平均座標 Y(t) の移動平均 M(t) を式 1 を用いて算出した。

$$M(t) = \frac{\Sigma_{i=0}^{4} Y(t-i)\alpha^i}{\Sigma_{k=0}^{4} \alpha^k} \tag{1}$$

ここで $\alpha = 0.8$ とする。

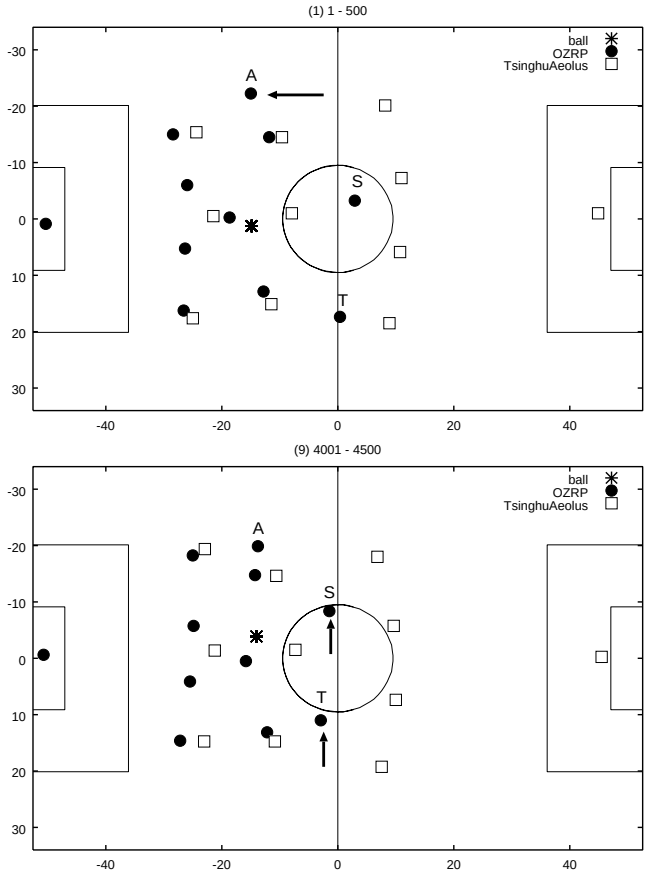


Figure 4: 人間のフォーメーションの時間変化

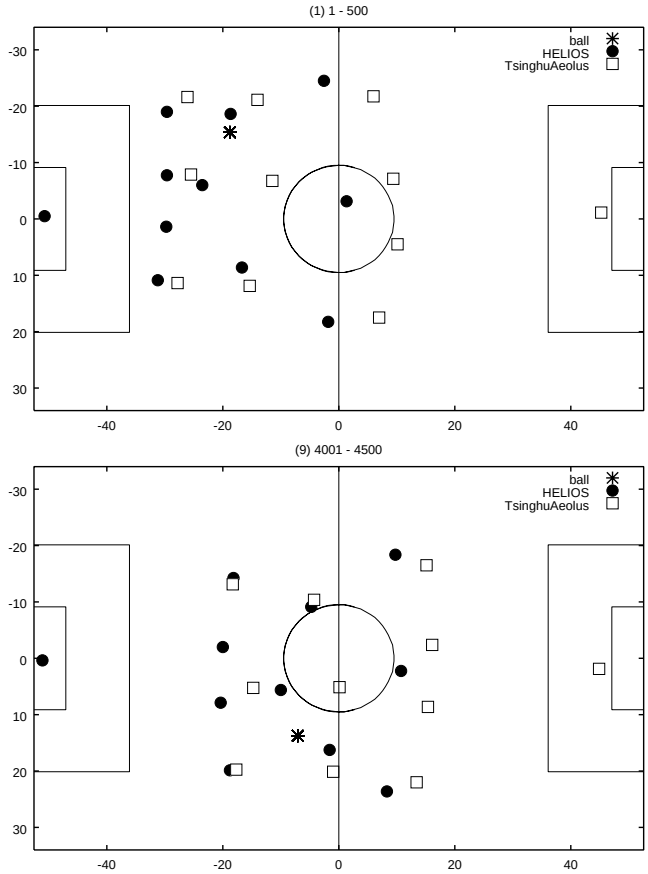


Figure 5: ロボットプレイヤ (HELIOS) のフォーメーションの時間変化

図 6 に、人間フォワードプレイヤの Y 方向の移動平均の時間変化を示す。

強い相手 (Tsinghu) との試合では、20 セグメントあたりから S と T は少しずつ位置が上方へと移動していることがわかる。また、A は他の試合に比べて少し中央によってはいるが、試合の中では大きな変化ない。

ポジションを移動したプレイヤ A とフォワードに残った 2 名 (S、T) のプレイヤとの Y 方向の平均座標の差を求め、式 1 と同様に移動平均を算出した。これにより、ボールの移動による全体の運動を除き、相対的なフォーメーションを示すことになる。S、T と A との Y 方向の距離を図 7、図 8 に示す。

人間プレイヤの 2 名と A との距離は、弱いチームとの試合では時間の経過とともに大きな変化はみられない。それに対して、強いチーム (Tsinghu) との試合では、A との距離が時間の経過とともに減少していることがわかる。

強いチームとの試合では、A は本来のフォワードのポジションから中盤へと移動している。そのため、A がいなくなり空いてしまった範囲を補うために、S は A の範囲も補うことができるような位置へと移動した。しかし、S が移動したことで、S の担当していた範囲を十分にカバーすることが出来なくなってしまう。そのため、T は S の範囲の一部を補うために S のポジションのほうへと移動したと考えることができる。

これは、人間が決められた環境のみだけではなく、状況の変化に応じて適切な行動をとることが出来ることを示している。

4 まとめ

シミュレーションサッカー環境における人間プレイヤの行動について、観測した記録にもとづいて分析した。フォーメーションの時間変化からは、あらかじめ準備されていない協調行動の発現が観測された。人間がもつサッカーにおけるシナリオ・チャンクの存在を示している。時間変動など統計的指標でも、ある程度の人間行動の特徴把握ができることが分かった。今回の知験をもとに、カバー行動に関する追加実験を行い、発現のプロセスをモデル化することが課題である。

参考文献

[1] P. Stone, T. Balch, and G. Kraetzschmar, editors. *Robocup 2000 : Robot Soccer World Cup IV*, volume 2019 of *LNAI*. Springer, 2001.

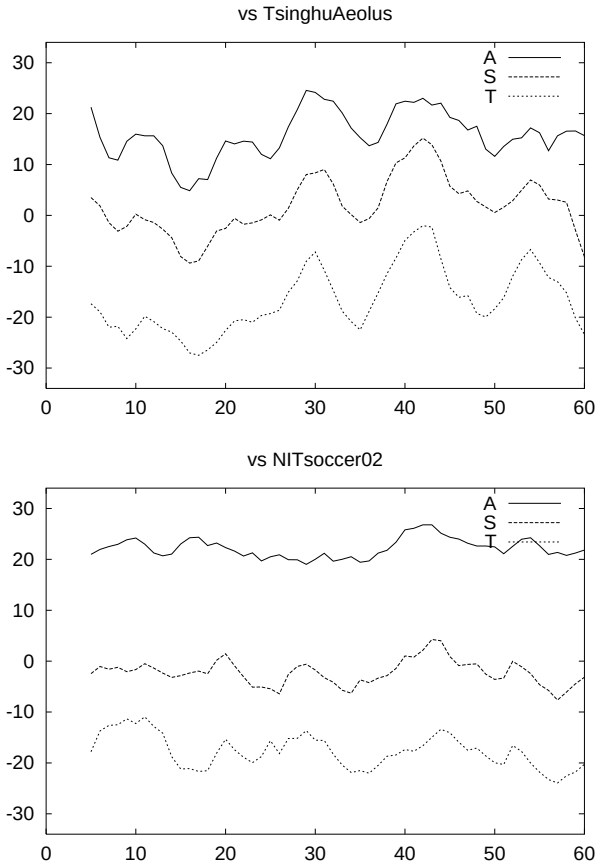


Figure 6: Y 方向の平均座標の時間変化

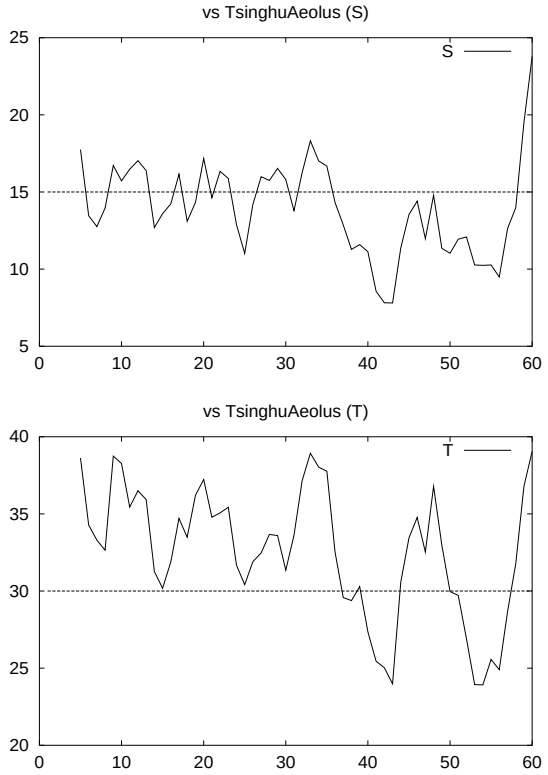


Figure 7: 強い相手との試合でのプレイヤ A に対する Y 方向の距離の時間変化

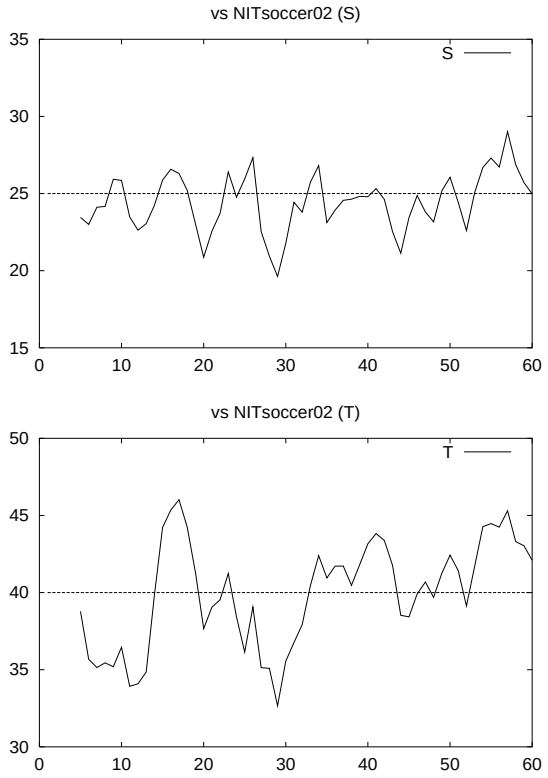


Figure 8: 弱い相手との試合でのプレイヤ A に対する Y 方向の距離の時間変化

[2] 秋田純一, 西野順二, 久保長徳, 下羅弘樹, and 藤墳到. 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. In *AI* **チャレンジ研究会第** *12* **回資料**, pages 23–28. 人工知能学会, 2001.

[3] 西野順二. ドリームチーム oz と人間チーム oz-rp の挑戦. **日本ロボット学会誌**, 20(1):39 – 40, 2002.

[4] 西野順二, 久保長徳, 秋田純一, and 下羅弘樹. Oz-rp システムを用いたバーチャルサッカーでの人間協調行動の観測. In **第** *11* **回インテリジェントシステムシンポジウム講演論文集**, pages 149–152, 2001.

[5] 西野順二, 戸田英治, 鹿田和之進, and 本多中二. バーチャルサッカーにおける人間 fw の特徴分析. **情報処理学会ゲーム情報学研究報告**, 2003-GI-9(35):1–6, 2003.

[6] 島涼平, 西野順二, and 本多中二. Oz-rp における協調行動の分析. **情報処理学会ゲーム情報学研究報告**, 2002-GI-7(27):9–16, 3 2002.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-7 (5/2)

A Hierarchical Multi-Module Learning System based on Self-Interpretation of Instructions by Coach

Yasutake Takahashi[1,2], **Koichi Hikita**[1], **and Minoru Asada**[1,2]
[1] Dept. of Adaptive Machine Systems,
[2] Handai Frontier Research Center,
Graduate School of Engineering, Osaka University
{yasutake,khikita,asada}@er.ams.eng.osaka-u.ac.jp

Abstract

We propose a hierarchical multi-module leaning system based on self-interpretation of instructions given by coach. The proposed method enables a robot (i) to decompose a long term task which needs various kinds of information into a sequence of short term subtasks which need much less information through its self-interpretation process for the instructions given by coach, (ii) to select sensory information needed to each subtask, and (iii) to integrate the learned behaviors to accomplish the given long term task. We show a preliminary result of a simple soccer situation in the context of RoboCup [1].

1 Introduction

Reinforcement learning (hereafter, RL) is a fascinating method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [2]. However, RL seems difficult to be applied to real robot tasks because the learning system needs a huge monolithic state/action space which consists of all kinds of sensory information and actuator commands, and such a big exploration space leads to scale the learning time exponentially. Therefore, it is hard to realize such learning systems from a practical viewpoint.

Fortunately, a long time-scale behavior might be often decomposed into a sequence of simple behaviors in general, and therefore, the searching space is expected to be able to be divided into some smaller ones. Connell and Mahadevan [3] decomposed the whole behavior into sub-behaviors each of which can be independently learned. However, task decomposition and behavior switching procedure are given by the designers. Morimoto and Doya [4] applied a hierarchical RL method by which an appropriate sequence of subgoals for the task is learned in the upper level while behaviors to achieve the subgoals are acquired in the lower level. In their system, the human designer has to define the subtasks based on their own experiences and insights. Doya et al. [5] have proposed MOdular Selection And Identification for Control (MOSAIC), which is a modular RL architecture for non-linear, non-stationary control tasks. However, the all learning modules share the same state space. Takahashi and Asada [6, 7] proposed a multi-layered RL system. The modules in the lower networks are organized as experts to move into different categories of sensor output regions and learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts which learn higher level behaviors using lower modules. However, this system tends to produce not only purposive behavior learning modules but also many non-purposive ones, and as a result, to require large computational resources.

When we develop a real robot which learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to the robot in order to accelerate the learning before it starts to learn. Whitehead [8] showed that instructions given by coach significantly encourages the learning and reduces the learning time. This method, called LBW (Learning By Watching), reduces the exploration space and makes learner have experiences to reach the goal frequently. Asada et al. [9] proposed a method, called LEM (Learning from Easy Mission). This basic idea is that a learning scheduling such as a robot starts to learn in easy situations to accomplish a given task at the early stage and learns in more difficult situations at the later stage accelerates the learning the purposive behavior. They applied this idea to a monolithic learning module. In order to cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the robot learns basic short term behaviors at the early stage and learns complicated long term behavior at the later stage based on instructions given by coach.

In this paper, we propose a behavior acquisition method based on hierarchical multi-module leaning system with self-interpretation of coach instructions. The proposed method enables a robot to

1. decompose a long term task into a set of short term subtasks,

2. select sensory information needed to the current subtask,

3. acquire a basic behavior to each subtask, and

4. integrate the learned behaviors to a sequence of the behaviors to accomplish the given long term task.

We show a preliminary result applied to a simple soccer situation in the context of RoboCup [1].

2 Basic Idea

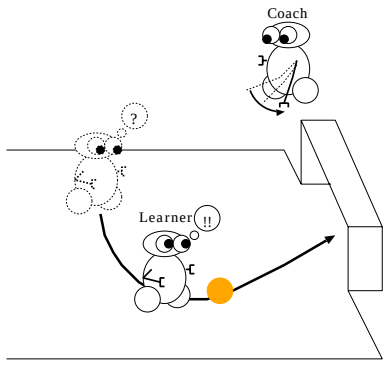


Figure 1: Basic concept: A coach gives instructions to a learner. The learner follows the instruction and find basic behaviors by itself.

There are a learner and a coach in a simple soccer situation (Figure 1). The coach has *a priori* knowledge of tasks to be played by the learner. The learner does not have any knowledge on tasks and just follows the instructions. After some instructions, the learner segments the whole task into a sequence of subtasks, acquire a behavior for each subtask, find the purpose of the instructed task, and acquire a sequence of the behaviors to accomplish the task by itself. It is reasonable to assume that the coach will give instructions for easier tasks at the early stage and give ones for complicated tasks at the later stage although it does not have any *a priori* knowledge about the learning system on the agent.

Figure 2 shows the perspective of development of the learning system through instructions given by coach at three stages. When the coach gives new instructions, the learner reuses the learning modules for familiar subtasks, generates new learning modules for unfamiliar subtasks at lower level and a new module for a sequence of behaviors of the whole instructed task at the upper level. After the learning at one stage, the learner adds newly acquired learning modules to the learning module database. The learning system iterates this procedure from easy tasks to more complicated ones.

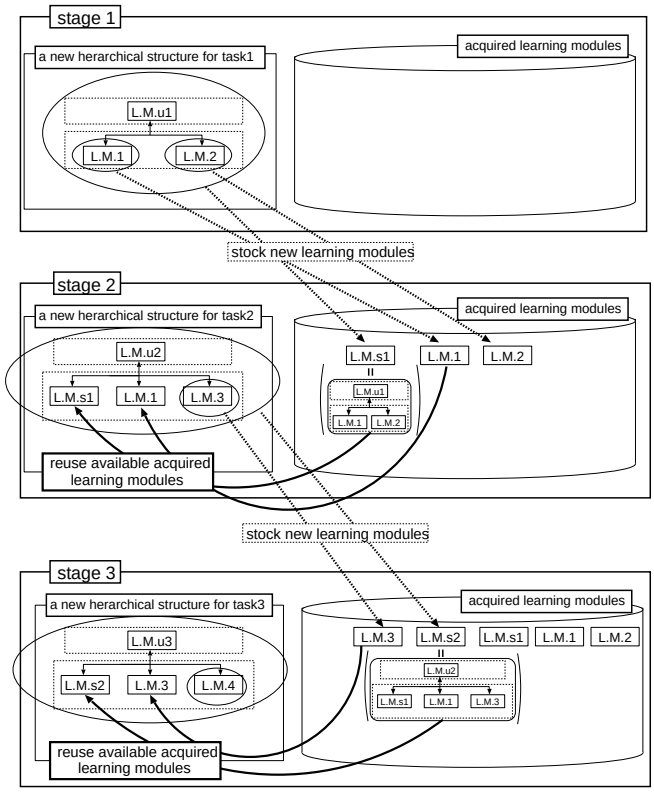


Figure 2: The perspective of of development of the learning system with staged instructions

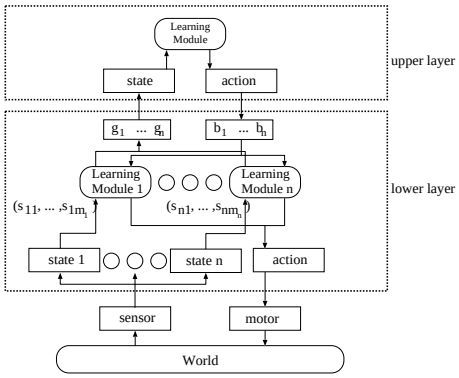
3 Hierarchical Multi-Module Learning System

3.1 Architecture

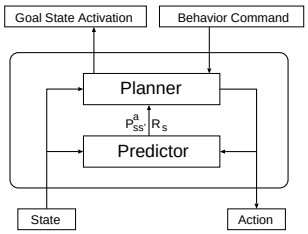
The basic idea of multi-layered learning system is similar to [6, 7]. The details of the architecture has been extended. The robot prepares learning modules of one kind, makes a layer with these modules, and constructs a hierarchy between the layers. The hierarchy of the learning module's layers can be regarded as a role of task decomposition. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (policy planner) which estimates the state-action value function based on the forward model in an RL manner (Figure 3(b)). The state and the action are constructed using sensory information and motor command, respectively at the bottom level.

The input and output to/from the higher level are the goal state activation and the behavior command, respectively, as shown in Figure 3. The goal state activation g is a normalized state value [1] , and $g = 1$ when the situation is the goal state. When the module receives the behavior command b from the higher modules, it calculates the optimal policy for its own goal, and sends action commands to the lower module. The action command at the bottom level is translated to an actual motor com-

[1] The state value function estimates the sum of the discounted reward received over time when the robot takes the optimal policy, and is obtained by Q learning.



(a) A whole system



(b) A module

Figure 3: A multi-layered learning system

mand, then the robot takes an action in the environment.

An approximated state-action value function $Q(s,a)$ for a state action pair (s,a) is given by

$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s',a') \right] , \quad (1)$$

where $\hat{\mathcal{P}}^a_{ss'}$ and $\hat{\mathcal{R}}^a_{ss'}$ are the state-transition probabilities and expected rewards, respectively, and the γ is the discount rate.

3.2 A Learning Procedure

The steps of the learning procedure are as follows:

1. Coach instructs some example behaviors to accomplish a task.
2. Learner evaluates the availability of learned behaviors to accomplish the task by watching the examples.
3. The learner segments the task into subtasks, and produces new learning modules at the lower layer if needed, and learns the behavior for each.
4. The learner produces a learning module at the higher layer and learns the whole behavior to accomplish the task.
5. Go to step 1.

3.3 Availability Evaluation

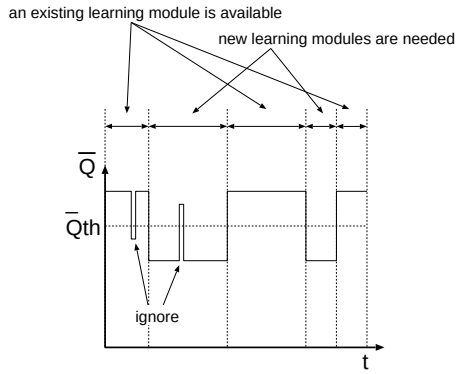


Figure 4: Availability identification during the given sample behavior

The learner needs to evaluate the availability of learned behaviors which help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired directly nor shows perfect example behavior from the learner's viewpoint. The learner should evaluate a module valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce $\overline{Q}$ in order to evaluate how suitable the module's policy is to the subtask as follows:

$$\overline{Q}(s,a_e) = \frac{Q(s,a_e) - min_{a'}\, Q(s,a')}{max_{a'}\, Q(s,a') - min_{a'}\, Q(s,a')} , \quad (2)$$

where a_e indicates the action taken in the instructed example behavior. $\overline{Q}$ becomes larger if a_e leads to the goal state of the module while it becomes smaller if a_e leaves the goal state. Then, we prepare a threshold $\overline{Q}_{th}$, and the learner evaluates the module valid for a period if $\overline{Q} > \overline{Q}_{th}$. If there are modules whose $\overline{Q}$ exceeds the threshold $\overline{Q}_{th}$ simultaneously, the learner selects the module which keeps $\overline{Q} > \overline{Q}_{th}$ for longest period among the modules (see Figure 4).

3.4 Generating new learning modules

If there is no module which has $\overline{Q} > \overline{Q}_{th}$ for a period, the learner creates a new module which will be assigned to the not-learned-yet subtask for the period. In order to assign a new module to such a subtask, the learner identifies the state space and the goal state. The following shows the steps briefly.

1. Prepare a set of state spaces $\boldsymbol{S}$ and, set their priorities as $\boldsymbol{S}_i : i = 1, 2, \cdots$.
2. For each state space $\boldsymbol{S}_i$,
   (a) Estimate a goal state space $\boldsymbol{G}$ in the state space $\boldsymbol{S}_i$ based on the instructed example behaviors.
   (b) If the estimated goal state space $\boldsymbol{G}$ covers all of the state space $\boldsymbol{S}_i$, increment i and goto step (a).
   (c) Construct a learning module and calculate Q values.
   (d) Check the performance of the learned behavior for the subtask. If the success rate is low, increment i and go to step (a).
3. Add a new module based on the state space $\boldsymbol{S}_i$ and the goal state space $\boldsymbol{G}$.
4. Check the availability of modules over the given task. If there is a period where there is no available module, go to step 1.
5. Exit.

3.4.1 State Variables Selection

If the number of state variables which construct the state space is large, the number of state scales up exponentially. We adopt heuristics, that is, only a few state variables are needed for all subtasks even if large number of state variables are necessary for the whole task: We limits the number of variables to only three in this study.

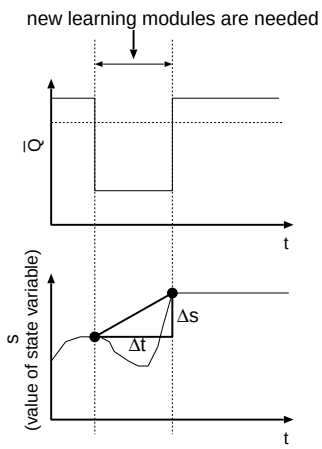


Figure 5: Gradient of a state variable

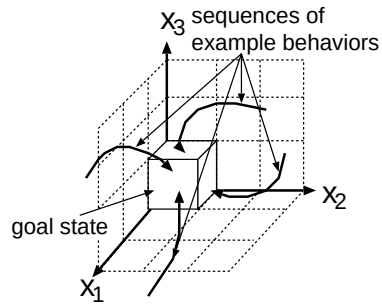
The system produces state spaces of all combinations of state variables and check their validities, however, this procedure takes much time. We introduce other heuristics and set priorities to the set of state spaces as follows:

1. Higher priority is assigned to the state variable which changes largely from the start to the end during the example behaviors because it can be regarded as an important variable to accomplish the subtask (see Figure 5).
2. Higher priority is assigned to the state space which has smaller average of entropy $H(s, a)$ (see equation 3) of the state transition probability $P^a_{ss'}$ for the experienced transition.

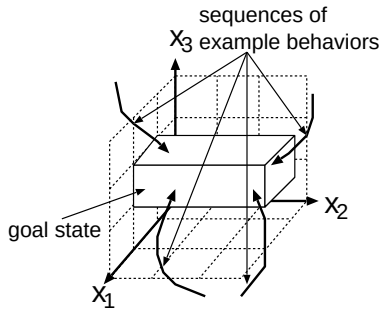
$$H(s,a) = -\sum_{s' \in S} P^a_{ss'}(s,a,s') log_2 P^a_{ss'}(s,a,s') \quad (3)$$

The reason is that the learning module acquires a more purposive behavior with more stable state transition probability which has lower entropy.

3.4.2 Goal State Space Selection



(a) goal state is specified by all variables x_1, x_2, x_3



(b) goal state doesn't care about variable x_2

Figure 6: Definition of goal state

It is hard to specify the goal state of the subtask with limited number of experiences of example behaviors. We need other heuristics here.

- A state variable of the goal state tends to be the maximum, the minimum, or the medium.
- If the value of a variable has no consistent one at the terminate state of the example behavior, the variable is independent of the goal state.

The system produce a reward model based on these heuristics.

3.4.3 Performance Evaluation

Even if we got a module with a proper policy based on the state transition model and reward one, there is no assurance that the module acquired a sufficient performance to the subtask. Before the system adds a new module to the available module database, it checks the success rate of the module. If the success rate is low, the system discards the module.

3.5 Learning behavior coordination

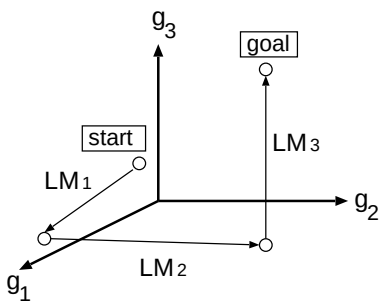


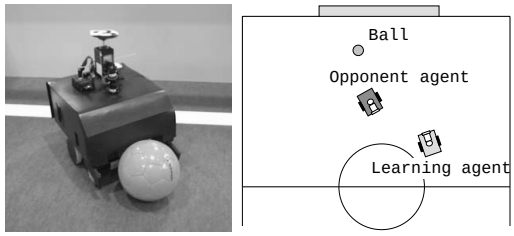
Figure 7: An example state transition on upper layer state space

After the procedures mentioned above, there should be necessary and sufficient modules at the lower layer, then the learning system puts a new learning module at the upper layer, and the module learns to coordinate the lower modules. The upper module has a state space constructed with the goal state activations of the lower modules. A set of actions consists of the commands to the lower modules. For example, there are three modules at lower level (say LM_1, LM_2, and LM_3), then the upper module has a state space based on their goal state activations (say g_1, g_2, and g_3). Figure 7 shows an example state transition on the upper layer state space. At the initial situation, all lower modules activate low. The system sends a command to the module LM_1, then the goal state activation of LM_1, that is g_1, goes up. After LM_1 finishes its own task, the upper module sends a command to the module LM_2, and accomplishes the whole task by activating LM_3 at last.

4 Experiments

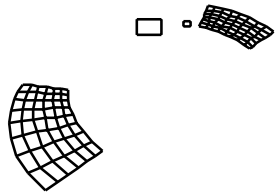
4.1 Setting

Figure 8 (a) shows a mobile robot we have designed and built. The robot has an omni-directional camera system. A simple color image processing (Hitachi IP5000) is applied to detect the ball area and an opponent one in the image in real-time (every 33ms). Figure 8 (b) shows a situation with which the learning agent can encounter and Figure 8 (c) shows the simulated image of the camera with the omni-directional mirror mounted on the robot. The larger and smaller boxes indicate the opponent and the ball, respectively. The robot has a driving mechanism, a PWS (Power Wheeled Steering) system.



(a) A real robot and a ball

(b) Top view of the field



(c) Simulated camera image with a omni-directional mirror

Figure 8: Real robot and simulation environment

The following state variables are prepared in advance:

A_i	area of object i on the image
θ_i	angle to object i from the center of image
A_{ij}	difference of areas between objects i and j
θ_{ij}	difference of angles between objects i and j

These variables are normalized to $[0:1]$. The area and the angle are quantized into 30 levels and 12 ones, respectively. The action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels.

4.2 Learning Scheduling and Experiments

The robot receives instructions for the tasks in the order as follows:

Task 1: ball chasing
Task 2: ball shooting into a goal without obstacles
Task 3: ball shooting into a goal with an obstacle

4.2.1 Task 1: ball chasing

First of all, the coach gives some instructions for the ball chasing task. There are the learner, a ball, the own goal, and the opponent goal in the environment. Figure 9 shows instructed behaviors for this task. According to the learning procedure mentioned in **3**, the system produce one module $LM[\boldsymbol{S}(A_b, \theta_b) : \boldsymbol{G}(Max, Front)]$, where $\boldsymbol{S}(A_b, \theta_b)$ indicates that the state space consists of area of ball A_b and angle of the ball θ_b from the center of the image, and $\boldsymbol{G}(Max, Front)$ indicates that the goal state is one where A_b is the maximum value and θ_b is the front of the robot. So this module acquired the behavior of ball chasing. Figure 10 shows the constructed system for this task 1.

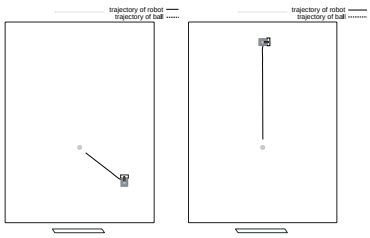


Figure 9: Example behaviors for task 1

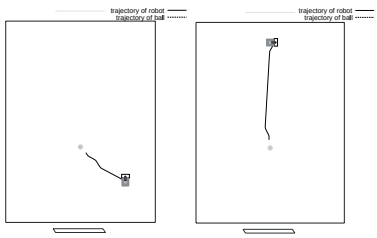


Figure 11: Learned behaviors for task 1

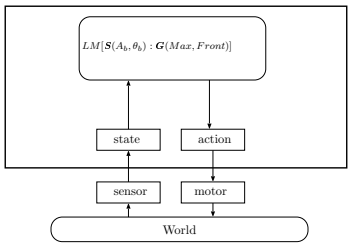


Figure 10: Acquired hierarchical structure (task 1)

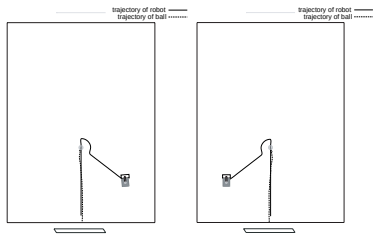


Figure 12: Example behaviors for task 2

4.2.2 Task 2: ball shooting into a goal without obstacles

At the second stage, the coach gives some instructions for the shooting task. Figure 12 shows example behaviors for this task, and Figures 13 and 14 show $\overline{Q}$s of the learning modules during the example behavior performance before and after the addition of new module, respectively. The arrows on the top of each series indicate the behavior of the instruction given by coach. There is no valid module during the time from 1.7 to 2.8 seconds. The learner produces another module $LM[\boldsymbol{S}(A_b, \theta_b, \theta_{bog}) : \boldsymbol{G}(Max, Don't\ care, Min)]$ during the period, where $\boldsymbol{S}(A_b, \theta_b, \theta_{bog})$ indicates that the state space consists of area of ball, angle of the ball from the center of the image, and difference between the angle of the ball and one of the goal, and $\boldsymbol{G}(Max, Don't\ care, Min)$ indicates that the goal state is one where A_b is the maximum value, θ_b is "*Don't care*", and θ_{bog} is the minimum value. This means that the module has a policy of going around the ball until the directions to the ball and the goal become same. Figure 15 shows the constructed hierarchical system for this task 2. The upper module coordinates these two modules to accomplish the shooting task. Figure 16 shows the acquired behaviors for the task, and Figure 17 shows the transitions of goal state activations and the selected learning module during the behavior performance.

4.2.3 Task 3: ball shooting into a goal with an obstacle

At the last stage, the coach gives some instructions for the shooting task with obstacle avoidance. Figure 18 shows example behaviors for this task. The learner produces another module $LM[\boldsymbol{S}(A_{op}, \theta_{op}, \theta_{ogop}) : \boldsymbol{G}(Max, Don't\ \ care, Max)]$, where $\boldsymbol{S}(A_{op}, \theta_{op}, \theta_{ogop})$ indicates that the state space consists of the area of opponent A_{op}, the angle of the opponent from the center of the image θ_{op}, and the difference between the angle of the opponent and the goal θ_{ogop}, and $\boldsymbol{G}(Max, Don't\ care, Max)$ indicates that the goal state is one where A_{op} is the maximum value, θ_{op} is "*Don't care*", and θ_{ogop} is the maximum value. Then this module acquired the behavior of going to the intersection between the opponent and the goal avoiding the collision. Figure 19 shows the constructed system for this task 3. The upper module enables the robot to shoot a ball into the goal avoiding the opponent. Figure 20 shows the acquired behaviors for the task, and Figure 21 shows the transitions of goal state activations and the selected learning module during the task accomplishment.

5 Conclusion

We proposed a hierarchical multi-module leaning system based on self-interpretation of instructions given by coach. We applied the proposed method to our robot and showed results of a simple soccer situation in the context of RoboCup.

Acknowledgment

This research was partially supported by the Japan Science and Technology Corporation, in Research for the the Core Research for the Evolutional Science and Technology Program (CREST) titled Robot Brain Project in the research area "Creating a brain."

References

[1] M. Asada, H. Kitano, I. Noda, and M. Veloso. Robocup: Today and tomorrow – what we have

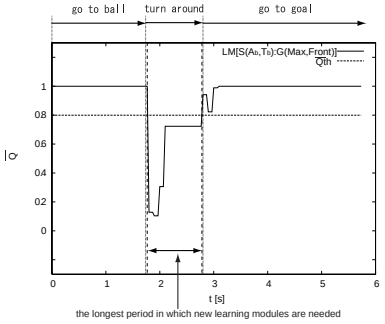


Figure 13: Availability evaluation : before the addition (Task 2)

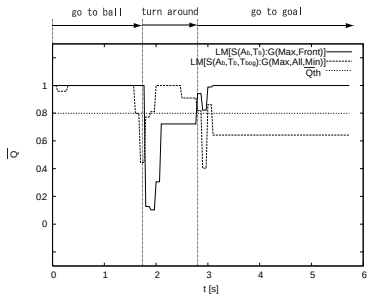


Figure 14: Availability evaluation : after the addition (Task 2)

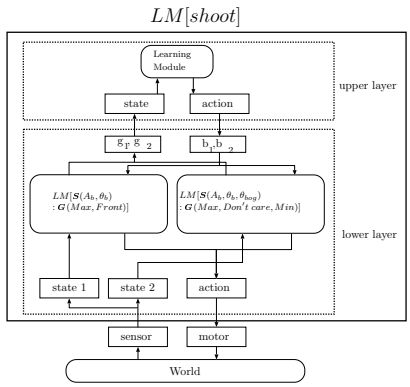


Figure 15: Acquired hierarchical structure (task 2)

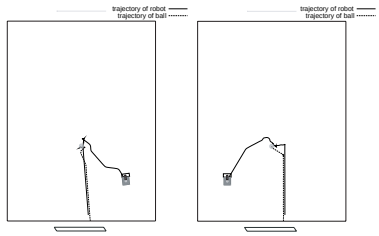


Figure 16: Learned behaviors for task 2

learned. *Artificial Intelligence*, pages 193–214, 1999.

[2] Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.

[3] Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*, chapter RAPID TASK LEARNING FOR REAL ROBOTS. Kluwer Academic Publishers, 1993.

[4] Morimoto J. and Doya K. Hierarchical reinforcement learning of low-dimensional subgoals and high-dimensional trajectories. In *The 5th International Conference on Neural Information Processing*, volume 2, pages 850–853, 1998.

[5] Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.

[6] Y. Takahashi and M. Asada. Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 395–402, 2000.

[7] Y. Takahashi and M. Asada. Multi-controller fusion in multi-layered reinforcement learning. In *International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI2001)*, pages 7–12, 2001.

[8] Steven D. Whitehead. Complexity and cooperation in q-learning. In *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pages 363–367, 1991.

[9] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Vision-based reinforcement learning for purposive behavior acquisition. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 146–153, 1995.

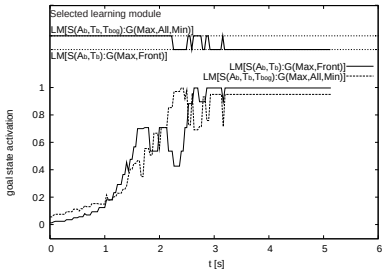


Figure 17: Transitions of goal state activations and the selected learning module (task2)

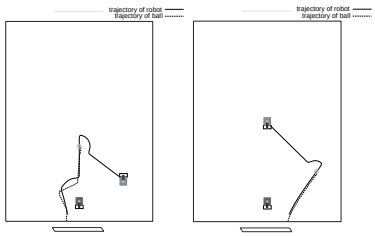


Figure 18: Example behaviors for task 3

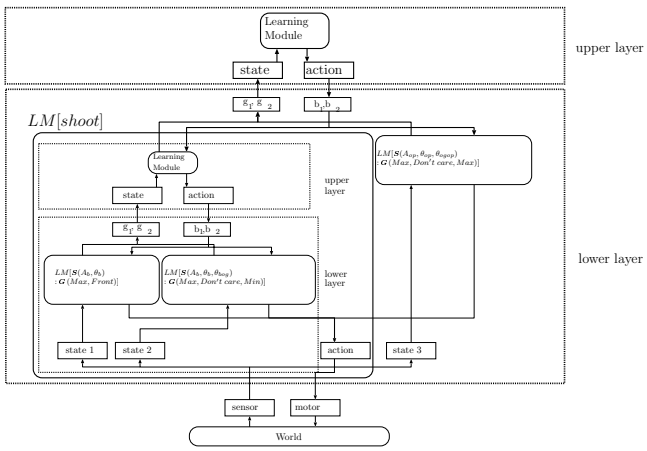


Figure 19: Acquired hierarchical structure (task 3)

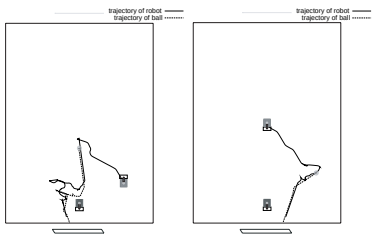


Figure 20: Learned behaviors for task 3

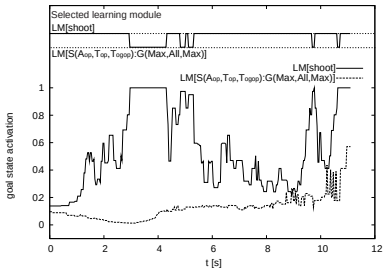


Figure 21: Transitions of goal state activations and the selected learning module (task 3)

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-8 (5/2)

Reinforcement Learning of Parameters for Humanoid Rhythmic Walking based on Visual Information

Masaki Ogino[1], Yutaka Katoh[1], Minoru Asada[1,2] and Koh Hosoda[1,2]
[1]Dept. of Adaptive Machine Systems, [2]HANDAI Frontier Research Center,
Graduate School of Engineering, Osaka University
{ogino, yutaka}@er.ams.eng.osaka-u.ac.jp, {asada, hosoda}@ams.eng.osaka-u.ac.jp

Abstract

This paper presents a method for learning the parameters of rhythmic walking to generate a purposive motion. The controller consists of the two layers. Rhythmic walking is realized by the lower layer controller which adjusts the speed of the phase on the desired trajectory depending on the sensor information. The upper layer controller learns (1) the feasible parameter sets that enable a stable walking for a robot, (2) the causal relationship between the walking parameters to be given to the lower layer controller and the change of the sensor information, and (3) the feasible rhythmic walking parameters by reinforcement learning so that a robot can reach to the goal based on the visual information. The method was examined in the real robot, and it learns to reach the ball and to shoot it into the goal in the context of RoboCupSoccer competition.

1 Introduction

Recently, a number of humanoid projects have started and various kinds of humanoid platforms have been developed. The typical method for real robot adopted in those platforms is planning the desired trajectory of each joint based on Zero Moment Point [3, 14]. In that method, the ZMP trajectory for not falling down is planned and the trajectory of each joint is calculated based on the ZMP trajectory. This method needs very precise dynamics parameters for the robot and much calculation time for planning.

The other method to realize bipedal walking is rhythmic-walking-based approach. This method doesn not use the precise structural parameters of a robot. Instead, the controller adjusts its inherent frequency depending on the sensor information so that the entrainment between dynamics of the controller and those of environment takes place. Taga et al. proposed the model of CPG (Central Pattern Generator) system [2] for human walking based on the nonlinear dynamics equations [11]. The network system changes its frequency depending on the sensor information. In the simulation experiment, this model realizes the stable walking under the various kinds of disturbances [6]. In the original Taga's CPG model, the output value of each neuron is used as a reference of torque applied to a corresponding joint. While almost all of the currently existing humanoid robots are driven by high gain PD controllers, instead of torque control. Therefore, it is difficult to apply Taga's CPG model to real robots directly. However, even such a robot with high gain PD controllers can realize the stable walking with a controller which utilizes sensor information properly. Pratt [9] realizes the energy efficient walking in real robot with a controller which consists of state machines. The state transition of the controller occurs when the swing leg touches the ground. Tsuchiya et al. [13] realized stable walking based on a method in which a trajectory controller determines the shape of the trajectory, and a phase controller changes the speed of the desired angle on the trajectory. In this controller the phase speed is adjusted by the sensor information.

In rhythmic walking, the control parameters are found heuristically, not by planning as ZMP approach. This makes it difficult to construct the upper layer controller to control the movement of a robot because the walking parameters such as walking step are not found until the robot interacts with the real world. Taga [12] and Fukuoka et al. [4] constructs the upper layer controller which gives the control parameters to the lower CPG controller depending the visual information so that the robot can avoid obstacles or climb over a step. In these methods, the adjusting parameters were given by the designer in advance. However, for making a more adaptive robot to the dynamic situations, it is necessary that the relationship between the parameters of the lower rhythmic walking controller and the resultant change of the environment should be learned.

In this paper, the layered controller is introduced, in which the lower controller realizes rhythmic walking based on the controller proposed by Tsuchiya et al. [13] and the upper controller learns the parameters of the

controller of the lower layer based on the visual information. There are three points in learning of the upper layer controller. (1) In the first stage, it learns the feasible parameters of the lower layer controller which enables a robot to walk. (2) To accelerate a learning process, the upper layer controller learns the model of the world : the relationship between the control parameters given to the lower rhythmic walking controller and the change of the visual sensor information. (3) The upper layer controller learns what parameters should be given to reach a goal by the reinforcement learning.

The rest of this paper is organized as follows. First, the lower controller which enables a rhythmic walk is introduced. Next, we describe the upper layer controller in which the parameters of the lower controller is learned by reinforcement learning. Then, the suggested controller is applied to the RoboCupSoccer task [8], "approaching to a ball", and experimental results are shown. Finally, conclusions are given.

2 Rhythmic walking controller

2.1 Biped robot model

Fig. 1 shows a biped robot model used in the experiment which has one-link torso, two four-link arms, and two six-link legs. All joints are single DOF rotation ones. Each foot has four FSRs to detect reaction force from the floor and a CCD camera with a fish-eye lens is attached at the top of the torso.

2.2 Rhythmic walking controller based on CPG principle

Here, we build a lower-layer controller based on the controller proposed by Tuchiya et al. [13]. The proposed controller consists of two sub-controllers: *a trajectory controller* and *a phase controller* (Fig. 2). The trajectory controller outputs the desired trajectory of each limb depending on the phase which is given by the phase controller. The phase controller consists of four oscillators, each of which is responsible for movement of each limb (Fig. 4). Each oscillator changes its speed depending on the touch sensor signal, and the effects reflected on the oscillator in each limb. As a result, the desired trajectory of each joint is adjusted so that global entrainment between dynamics of the robot and those of the environment is realized. In the following, the details of each controller are explained.

2.2.1 Trajectory controller

The trajectory controller calculates the desired trajectory of each joint depending on the phase given by the corresponding oscillator in the phase controller.

Here, the trajectory of each joint is characterized by four parameters as shown in Fig. 3. For joints 3, 4 and 5, of which axes coincide with pitch axis, the desired trajectory is determined so that in the swing phase the foot trajectory draws a ellipse that has the radiuses, h in vertical direction and β in horizontal direction, respectively. For joints 2 and 4, of which axes coincide with roll axis, the desired trajectory is determined so that the leg tilts from $-W$ to W relative to the vertical axis. The desired trajectory of joint 1 is determined by the amplitude of the oscillation, α. The desired trajectories are summarized as following functions,

$$\theta_1 = \alpha \sin(\phi) \quad (1)$$
$$\theta_2 = W \sin(\phi) \quad (2)$$
$$\theta_i = f_i(\phi, h, \beta) \qquad (i = 3, 4, 5) \quad (3)$$
$$\theta_6 = -W \sin(\phi). \quad (4)$$

The detail of f_i is explained in Appendix. Among four parameters described above, α, which determines the walking step length, and β, which determines the walking direction are selected as rhythmic parameters of walking. Although these parameters characterize approximate direction and step length, resultant walking is not as precisely determined by those parameters because of the slips between the support leg and the ground. These parameters are learned in the upper layer learning module, explained in **3**.

2.2.2 Phase controller

The phase which determines the desired value of each joint is given by the phase controller. The phase controller consists of two oscillators, ϕ_R for right leg and ϕ_L for left leg. The dynamics of each oscillator is determined by basic frequency, ω, the interaction term between two oscillators, and the feedback signal from sensor information,

$$\dot{\phi}_L = \omega - K(\phi_L - \phi_R - \pi) + g_L \quad (5)$$
$$\dot{\phi}_R = \omega - K(\phi_R - \phi_L - \pi) + g_R. \quad (6)$$

The second term of RHS in above equations keeps the phases of two oscillators in opposite. The third term, feedback signal from sensor information, is given as follows:

$$g_i = \begin{cases} K' Feed_i & (0 < \phi < \phi_C) \\ -\omega(1 - Feed_i) & (\phi_C \leq \phi < 2\pi) \end{cases} \quad (7)$$
$$i = \{R, L\},$$

where K', ϕ_C and $Feed_i$ denote feedback gain, the phase when the swing leg contacts with the ground, and the feedback sensor signal, respectively. $Feed_i$ returns 1 if the FSR sensor value of the corresponding leg exceeds the certain threshold value, otherwise 0. The third term enables that the mode switching between the free leg phase and the support one happens appropriately according to the ground contact information from the FSR sensors. In this paper, the value of each parameter is set as follows; $\phi_C = \pi$, $\omega = 5.23$[rad/sec], $K = 15.7$, $K' = 1$.

3 Reinforcement learning with rhythmic walking parameters

3.1 Principle of reinforcement learning

Reinforcement learning has recently been receiving increased attention as a method for robot learning with little or no *a priori* knowledge and higher capability of

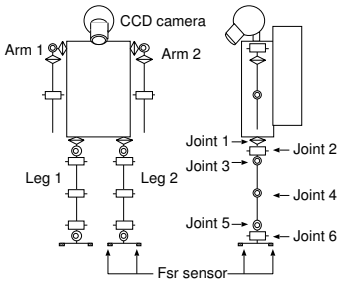


Figure 1: Model of biped locomotion robot

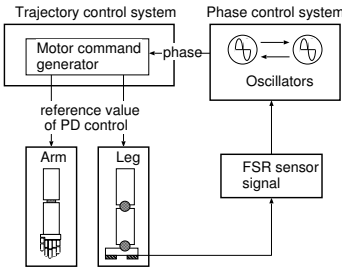
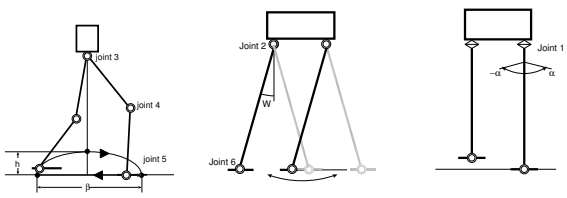


Figure 2: Walking control system



(a) Joint angles around pitch axis

(b) Joint angles around roll axis

(c) Joint angle around yaw axis

Figure 3: Joint angles

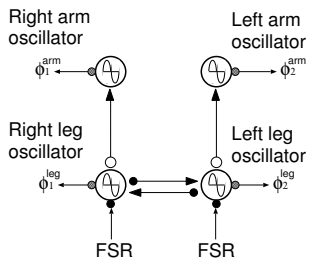


Figure 4: Phase control system

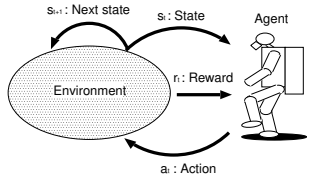


Figure 5: Basic model of agent-environment interaction

reactive and adaptive behaviors. Fig. 5 shows the basic model of robot-environment interaction [10], where a robot and environment are modelled by two synchronized finite state automatons interacting in a discrete time cyclical processes. The robot senses the current state $s_t \in \boldsymbol{S}$ of the environment and selects an action $a_t \in A$. Based on the state and action, the environment makes a transition to a new state $s_{t+1} \in \boldsymbol{S}$ and generates a reward r_t that is passed back to the robot. Through these interactions, the robot learns a purposive behavior to achieve a given goal. In order for the learning to converge correctly, the environment should satisfy the Markovian assumption that the state transition depends on only the current state and the taken action. The state transition is modelled by a stochastic function $\boldsymbol{T}$ which maps a pair of the current state and the action to take to the next state ($\boldsymbol{T} : \boldsymbol{S} \times \boldsymbol{A} \rightarrow \boldsymbol{S}$). Using $\boldsymbol{T}$, the state transition probability $P_{s_t,s_{t+1}}(a_t)$ is given by

$$P_{s_t,s_{t+1}}(a_t) = Prob(\boldsymbol{T}(s_t, a_t) = s_{t+1}). \tag{8}$$

The immediate reward r_t is given by the reward function in terms of the current state by $R(s_t)$, that is $r_t = R(s_t)$. Generally, $P_{s_t,s_{t+1}}(a_t)$ (hereafter $\mathcal{P}^a_{ss'}$) and $R(s_t)$ (hereafter $\mathcal{R}^a_{ss'}$) are unknown.

The aim of the reinforcement learner is to maximize the accumulated summation of the given rewards (called *return*) given by

$$return(t) = \sum_{n=0}^{\infty} \gamma^n r_{t+n}, \tag{9}$$

where $\gamma\,(0 \leq \gamma \leq 1)$ denotes a discounting factor to give the temporal weight to the reward.

If the state transition probability is known, the optimal policy which maximize the expected *return* is given by finding the optimal value function $V^*(s)$ or the optimal action value function $Q^*(s,a)$ as follows. The derivation of them can be found elsewhere [10].

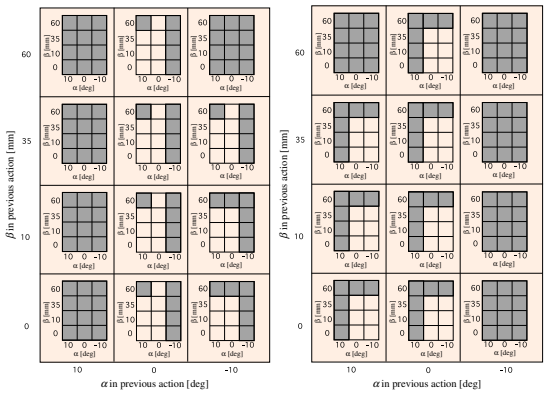
$$\begin{aligned} V^*(s) &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} \\ &= \max_a \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma V^*(s')\right] \end{aligned} \tag{10}$$

$$\begin{aligned} Q^*(s,a) &= E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\} \\ &= \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right] \end{aligned} \tag{11}$$

3.2 Construction of action space based on rhythmic parameters

The learning process has two stages. The first one is to construct the action space consisting of feasible combinations of two rhythmic walking parameters (α, β). To do that, we prepared the three-dimensional posture space s_p in terms of the forward length β (quantized into four lengths: 0, 10, 35 60 [mm]), the turning angle α (quantized into three angles: -10, 0, 10 [deg]) both of which mean the previous action command, and the leg side

(left or right). Therefore, we have 24 kinds of postures. Firstly, we have constructed the action space of the feasible combinations of (α, β) excluding the infeasible ones which cause collisions with its own body. Then, various combinations of actions are examined for stable walking in the real robot. Fig. 6 shows the feasible actions (empty boxes) for each leg corresponding to the previous actions. Due to the differences in physical properties between two legs, the constructed action space was not symmetric although it should be theoretically.



(a) left leg (b) right leg

Figure 6: Experimental result of action rule

3.3 Reinforcement learning with visual information

Fig. 7 shows an overview of the whole system which consists of two layers: adjusting walking based on the visual information and generating walking based on neural oscillators. The state space consists of the visual information s_v and the robot posture s_p, and adjusted action a is learned by dynamic programming method based on the rhythmic walking parameters (α, β). In a case of ball shooting task, s_v consists of ball substates and goal substates both of which are quantized as shown in Fig. 8. In addition to these substates, we add two more substates, that is, "the ball is missing" and "the goal is missing" because they are necessary to recover from loosing their sight.

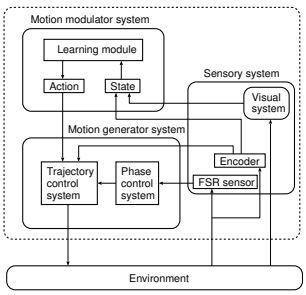
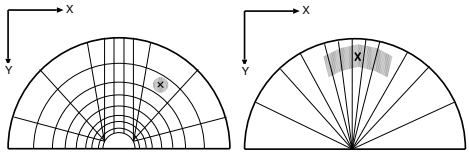


Figure 7: Biped walking system with visual perception

Learning module consists of a planner which determines an action a based on the current state s, a state



(a) State space of ball (b) State space of goal

Figure 8: State space of ball and goal

transition model which estimates the state transition probability $\mathcal{P}^a_{ss'}$ through the interactions, and a reward model (see Fig. 9). Based on DP, the action value function $Q(s,a)$ is updated and the learning stops when no more changes in the summation of action values.

$$Q(s,a) = \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}_s + \gamma \max_{a'} Q(s',a')], \qquad (12)$$

where $\mathcal{R}_s$ denote the expected reward at the state s.

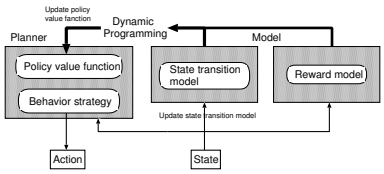


Figure 9: Learning module

4 Experiments

4.1 Robot platform and environment set-up

Here, we use a humanoid platform HOAP-1 by Fujitsu Automation LTD. [7] attaching a CCD camera with a fish-eye lens at the head. Figs. 10 and 11 show a picture and a system configuration, respectively. The height and the weight are about 480[mm] and 6[kg], and each leg (arm) has six (four) DOFs. Joint encoders have high resolution of 0.001[deg/pulse] and reaction force sensors (FSRs) are attached at soles. The colour image processing to detect an orange ball and a blue goal is performed on the CPU (Pentium3 800MHz) under RT-Linux. Fig. 12 shows an on-board image.

The experimental set-up is shown in Fig. 13 where the initial robot position is inside the circle whose center and radius are the ball position and 1000 [mm], respectively, and the initial ball position is located less than 1500 [mm] from the goal of which width and height are 1800 [mm] and 900 [mm], respectively. The task is to take a position just before the ball so that the robot can shoot a ball into the goal. Each episode ends when the robot succeeds in getting such positions or fails (touches the ball or the pre-specified time period expires).

4.2 Experimental results

One of the most serious issues in applying the reinforcement learning method to real robot tasks is how

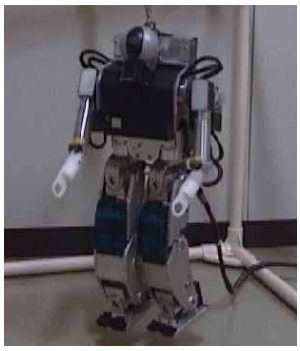


Figure 10: HOAP-1

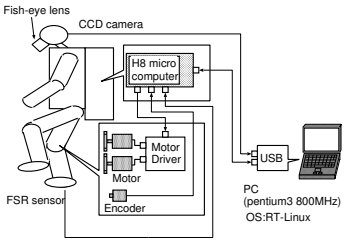


Figure 11: Overview of robot system

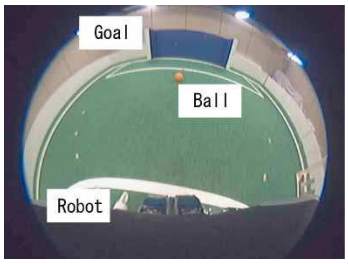


Figure 12: Robot's view (CCD camera image through fish-lens)

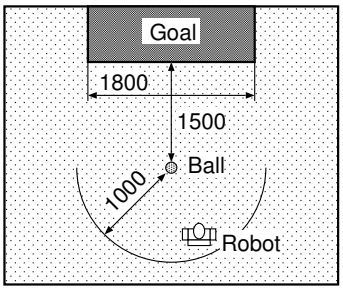


Figure 13: Experimental environment

to accelerate the learning process. Instead of using Q-learning that is most typically used in many applications, we use a DP approach based on the state transition model $\mathcal{P}^a_{ss'}$ that is obtained separately from the behavior learning itself. Further, we give the instructions to start up the learning, more correctly, during the first 50 episodes (about a half hour), the human instructor avoids the useless exploration by directly specifying the action command to the learner about 10 times per one episode. After that, the learner experienced about 1500 episodes. Owing to the state transition model and initial instructions, the learning converged in 15 hours, and the robot learned to get the right position from any initial positions inside the half field.

Fig. 14 shows the learned behaviors from various initial positions. In Fig. 14, the robot can capture the image including both the ball and the goal from the initial position while in Fig. 14 (f) the robot cannot see the ball or the goal from the initial position.

(a) Result 1 (b) Result 2 (c) Result 3

(d) Result 4 (e) Result 5 (f) Result 6

Figure 14: Experimental results

5 Concluding remarks

A vision-based behavior of humanoid was generated by reinforcement learning with rhythmic walking parameters. Since the humanoid generally has many DOFs, it is very hard to control all of them. Instead of using these DOFs as action space, we adopted rhythmic walking parameters, which drastically reduces the search space and therefore the real robot learning was enabled in reasonable time. In this study, the designer specified the state space consisting of visual features and robot postures. State space construction by learning is one of the future issues.

Acknowledgments

This study was performed through the Advanced and Innovational Research program in Life Sciences from the Ministry of Education, Culture, Sports, Science and Technology, the Japanese Government.

References

[1] A. Fujii, A. Ishiguro. Evolving a cpg controller for a biped robot with neuromodulation. In *Climbing and Walking Robots* (2002), pp. 17–24.

[2] S. Grillner. Neurobiological Bases of Rhythmic Motor Acts in Vertebrates. *Science*, Vol. 228, (1985), pp. 143-149.

[3] S. Kajita and K. Tani. Adaptive Gait Control of a Biped Robot based on Realtime Sensing of the Ground Profile. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1996), pp. 570-577.

[4] H. Kimura, Y. Fukuoka, and H. Nakamura. Biologically inspired adaptive dynamic walking of the quadruped on irregular terrain. In *Proc. of 9th International Symposium of Robotics Research* (1999), pp. 271-278.

[5] M. Laurent and J. A. Thomson. The role of visual information in control of a constrained locomotor task. *J. Mot. Behav*, Vol. 20, (1988), pp. 17–37.

[6] S. Miyakoshi, G. Taga, Y. Kuniyoshi, and A. Nagakubo. Three Dimensional Bipedal Stepping Motion using Neural Oscillators –Towards Humanoid Motion in the Real World–. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1998), pp. 84-89.

[7] Y. Murase, Y. Yasukawa, K. Sakai, etc. Design of a Compact Humanoid Robot as a Platform. In *19th conf. of Robotics Society of Japan*, (2001), pp. 789-790.

[8] H. Kitano, M. Asada The RoboCup humanoid challenge as the millennium challenge for advanced robotics. In *Advanced Robotics*, vol. 13, (2000), no. 8, pp. 723-736.

[9] J. Pratt Exploiting Inherent Robustness and Natural Dynamics in the Control of bipedal Walking Robots Doctor thesis, MIT, June. (2000).

[10] Richard S.Sutton and Andrew G.Barto. "Reinforcement learning:An Introduction", MIT Press/Bradford Books, March, (1998).

[11] G. Taga, Y. Yamaguchi, H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, Vol. 65, (1991), pp. 147–159.

[12] G. Taga. A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance. *Biological Cybernetics*, Vol. 78, (1998), pp. 9–17.

[13] K. Tsuchiya, K. Tsujita, K. Manabu, S. Aoi. An emergent control of gait patterns of legged locomotion robots. *IAV2001*, pp. 271-276, (2001), pp. 271-276.

[14] J. Yamaguchi, N. Kinoshita, A. Takanishi, and I. Kato. Development of a Dynamic Biped Walking System for Humanoid –Development of a Biped Walking Robot Adapting to the Human's Living Floor–. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems* (1996), pp. 232-239.

Appendix: planning the reference trajectory around the pitch axis

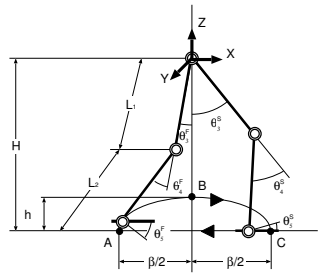


Figure 15: Joint angles and the reference trajectory of the foot

The reference trajectories of joints 3, 4 and 5 are determined by the position of the foot. Let x and z be the position of the foot in the plane XZ which is perpendicular to the pitch axis, the reference trajectory of the foot is given by,

$$
\begin{aligned}
x_F &= \frac{\beta}{2}\cos(\phi^F), \\
z_F &= -H + h\sin(\phi^F), \\
x_S &= -\frac{\beta}{2}\cos(\phi^S), \\
z_S &= -H,
\end{aligned}
$$

where (x_F, z_F) and (x_S, z_S) are the positions of the foot in the free and support phase, respectively, H is the length from the ground to the joint 3, β is the step length, and h is the maximum height of the foot from the ground (Fig. 15). When the position of the foot is determined, the angle of each joint to be realized is calculated by the inverse kinematics as follows,

$$
\begin{aligned}
\theta_3 &= \frac{\pi}{2} + atan2(z, x) - atan2(k, x^2 + z^2 + L_1^2 - L_2^2) \\
\theta_4 &= atan2(k, x^2 + z^2 - L_1^2 - L_2^2) \\
\theta_5 &= -(\theta_3 + \theta_4),
\end{aligned}
$$

where k is given by the following equation,

$$
k = \sqrt{(x^2 + z^2 + L_1^2 + L_2^2)^2 - 2\{(x^2 + z^2)^2 + L_1^4 + L_2^4\}}.
$$

In this research, the value of each parameter is set as follows; $H = 185$[mm], $h = 8$[mm], $W = 13$[deg], $L_1 = 100$[mm], $L_2 = 100$[mm].

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-9 (5/2)

ロボカップ小型リーグにおける協調プレー実現の一手法について

Realization of Cooperative Soccer Play by Small-Size Robots

村上和人 [1] 日比野晋也 [2] 児玉幸治 [2] 加藤恭佑 [2] 鈴木英之 [1] 山本浩司 [1] 成瀬 正 [1]
Kazuhito Murakami [1], Shinya Hibino [2], Yukiharu Kodama [2], Kyosuke Kato [2],
Hideyuki Suzuki [1], Koji Yamamoto [1] and Tadashi Naruse [1]
1:愛知県立大学 情報科学部 2:愛知県立大学大学院情報科学研究科
1: Faculty of Information Science and Technology, Aichi Prefectural University
2:Graduate School of Information Science and Technology, Aichi Prefectural University
apurobo@ist.aichi-pu.ac.jp

Abstract

One of the typical cooperative actions is the pass play in RoboCup small-size league. This paper presents three technical key features to realize robust pass play between robots. First one is the high resolution image processing to detect the positions and orientations of the robots. Second one is the control algorithm to move and adjust the robots for the pass play. Third one is the mechanism to catch the ball moving at high speed. This paper discusses these methods and shows the effectiveness of the methods by experimental results.

1. はじめに

ロボカップ小型リーグでは、実機ロボットを用いてゲームを行う[1-5]。パスプレーは単に得点の機会を増やすだけではなく、ロボット間の協調動作を実現する観点からも、その実現が期待されている。

ロバストなパスプレーを実現するためには、幾つかの技術要素がある。第一に、ソフトウェアの観点からは、ロボットの正確な位置・方向を検出する高精度な画像処理手法は欠かせない。というのも、これらの情報を検出できない、あるいは、できたとしても時間がかかるようでは、決められた位置でボールを受け取ることができなくなるためである。第二に、相手ロボットに正確にボールを蹴り出すためには、ロボットの位置・方向制御アルゴリズムも重要である。第三に、高速で移動するボールを受け取るためには、キャッチ機構も必要である。

本論文では、パスプレーを実現するための諸技術、すなわち、ロボットとボール追跡が約 1.4msec で実現可能な探索領域削減手法を組み込んだ画像処理手法、パスプレーを行う 2 台のロボットの制御アルゴリズム、および、あそびを持たせることにより保持率を高めたボールキャッチ機構について述べる。以下、２．～４．で各々、画像処理手法、協調動作アルゴリズム、および、ロボットのハードウェアメカニズムについて述べる。

2. 画像処理システム

2. 1 認識対象物

チームを識別するために、ルールでは黄または青の直径 4cm のチームマーカーをロボット上面に装着することになっている。ロボット同士の区別、あるいは、ロボットの向きの検出のために、必要ならば規定の範囲内で補助マーカーの装着が許されている。相手チームに影響がなければ色に関する制限はない（ただし、白、黒は使用可能）。数、配置なども自由である。各チームとも形状、色、配置に工夫を凝らしているが、筆者らのチームでは Figure 1 に示すような長方形パターン（105mm×16.5mm）をロボットの方向検出に、また、小さな円形パターン（1～4 個、直径 8.5mm）をロボットの個体識別に利用している。画像処理システムは、これらのパターン検出を目的とする。

2. 2 画像処理システム

PentiumXeon 2GHz、Windows2000 の PC を使用している。高速に動くロボットやボールを検出するために、プログレッシブ方式のカメラ（SONY, DXC-9000）を用いている。

カメラはフィールドの上方 3.0m に設置されているが、ゴール部分まで含めると 3.2m の範囲をカバーする必要がある。このため、広角レンズアタッチメントを装着して対処している（Fujinon, WCV-65, ×0.75）。画像入力ボード（Matrox, GEN/X/00/STD）より取り込まれる画像のサイズは 640(W)×480(H)であり、5mm/pixel の分可能である。

Figure 1: Overview of our robot team

2．3　画像処理アルゴリズム

画像処理システムでは、チームマーカーを利用してロボットの位置検出を行い、ロボットの個体（ID）識別ならびにロボットの向きの検出にはサブマーカーを利用している。画像処理の流れを Figure 2 に示す。処理概略は次のとおりである。なお、処理の詳細は文献[6-8]を参照されたい。

Step-1(Input Image)：プログレッシブスキャンカメラから RGB 画像を入力する。入力画像の一例を Figure 3 に示す。Figure 3(b)はロボットの周辺を拡大したものである。

Step-2(Calculation of Search Range)：前フレームの結果の信頼度に基づいて探索領域を決定する。信頼度は画像処理結果を特性づける重要なパラメータであるので、詳しくは後で述べる。Figure 4 の白線で囲った長方形領域は探索範囲である。この範囲内において、RGB 画像を YUV 表色系に変換する。探索領域の限定によって計算時間が大幅に短縮されている。

Step-3(Segmentation of Colored Region)： CMU の方法[9]を拡張した 9 色（最大 32 色まで可能）のカラー照合を Step-2 と同じ範囲で YUV 画像に対して同時的に行う。実際に対戦が行われる会場の照明は非均一なことが多く、明るさもスペクトラムも同一ではないため、色に幅をもたせている。具体的には、ボールに 2 色、それぞれのチームカラーに 2 色ずつ、フィールドから除外したい色に 3 色割り振っている。この手法により、明るさの他、フィールド上における色あいの違いも吸収するようにしている。

Step-4（Labeling）：伝播処理を利用したラベリングを行っているが、その際、ラベリング処理の最初の伝播を対角近傍にのみ行うようにしている[8]。Figure 5 に処理結果を示す。

Step-5（Selection of Objects）：ID マーカーと長方形のサブマーカーを検出する。長方形の ID マーカーと円形のサブマーカーの区別は面積によって容易に判別可能である。Figure 6 に処理結果の一例を示す。

Step-6（Calculation of ID’s and Directions）：円形サブマーカーの個数と位置を利用して ID を識別し、ロボットの大まかな向きを計算する。精度は約 8°である。

Step-7（Calculation of Modified Directions）：Figure 7 に示すように、長方形パターンの長辺のエッジ点列に最小二乗法を適用し、ロボットの詳細な向きを計算する。精度は約 1°に向上する。

Step-8（Generation of Position Record）：現在のロボットの位置情報を記録する。

Step-9（Object Position with ID and Direction Information）：カメラ画面の座標からワールド座標に変換する。画像処理の最終結果の一例を Figure 8 に示す。

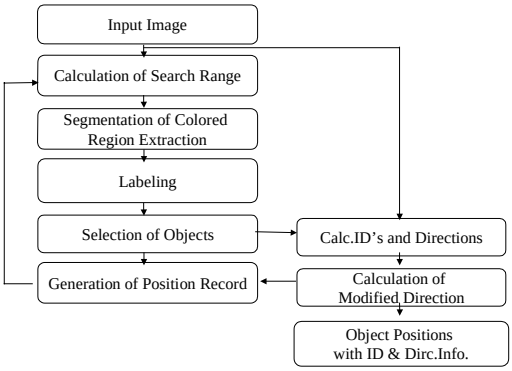
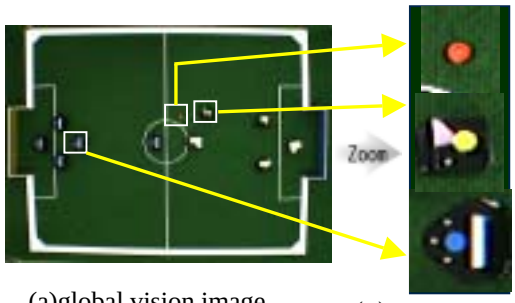


Figure 2: Flowchart of image processing



(a)global vision image　　（b）enlarged image

Figure 3: An example of grabbed image from a global vision camera

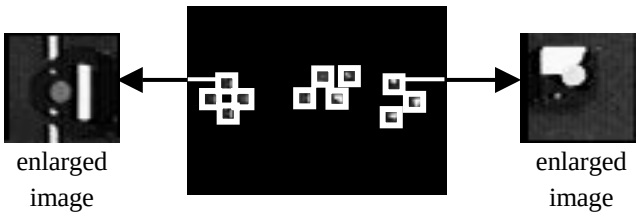
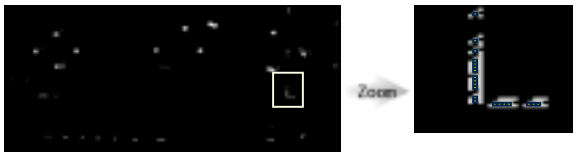


Figure 4: Restricted region to be searched (whitened area) and enlarged images



Since the noise like a straight line of width 1pixel will be recognized as separated objects, it is easy to remove them by simple area thresholing.

Figure 5: An example of line noises to be deleted by the proposed labeling algorithm

Step-2 と Step-3 において信頼度という指標を用いたが、これは画像処理結果の信頼性をパラメータ化表現したものである。つまり、もし、前フレームの処理における対象物検出（追跡）の信頼度が低い場合は、次のフレームでは探索領域は広めにとる必要がある。逆に、信頼度が高ければ、

次のフレームでは、Figure4 に示したような限定領域内だけ探索すれば十分に対象物を検出（追跡）できることになる。

筆者らのシステムでは、{non, low, intermediate, high}の 4 段階の信頼度レベルを定義し、フレームレート（1/60sec ごと）に更新するようにしている。基本的には、もし対象物検出に成功すれば 1 レベル上げ、そうでなければ 1 レベル下げる。

信頼度が下がると広い探索領域から探さなければならず、計算時間を要する。しかし、情報管理システムは過度に下がるのを防いでいる。例えばボールが隠蔽状態になったとき、このルールでは信頼度が下がりつづけるが、隠蔽状態のときにはレベル 2（intermediate-level）になるように補正している。具体的な探索領域サイズは、レベル 1（high-level）のとき 20×20pixel、レベル 2（intermediate-level）のとき 30×30pixel、レベル 3（low-level）のとき 60×60pixel であり、20×20pixel は実際のフィールドでは 10cm×10cm に相当している。本手法では 3m/sec までのボール追跡が可能である。

(a) the result of marker extraction　　(b) enlarged image

Figure 6: The result of marker extraction

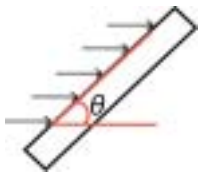


Figure 7: Calculation of modified direction by LMS

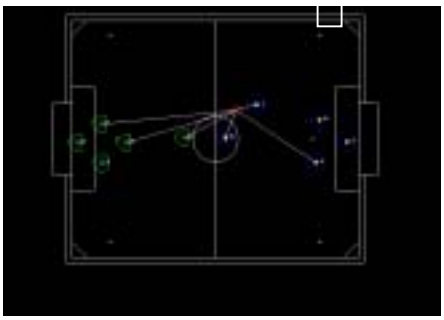


Figure 8: Final result of image processing for the game field

3. 協調動作アルゴリズム

筆者らのシステムが備える協調プレーには、例えば、パス、ディフェンス、アシスト、などがある。本稿では、紙面の制約により、パスプレーのアルゴリズム概要を説明する。パスの出し手と受け手のそれぞれのアルゴリズムは次のとおりである。

まず、変数、フラグを次のように定義する。パスの出し手と受け手のロボット中心を結ぶ線を $\mathrm{Line_A}$ とする。また、出し手ロボットの前方に伸びる直線を $\mathrm{Line_B}$ とする。そして、$\mathrm{Line_A}$ と $\mathrm{Line_B}$ のなす角を θ とする。方向フラグ、安定フラグを各々、Dir、PassCounter とする。

[Initializing]

Step 1. もし $|\theta|$ が 20 °より大きければ、回転速度 v_r を $\theta/20+\theta/|\theta|\times 9$ (cm/sec)とする。そうでなければ $\theta/2$ (cm/sec)とする。回転を開始する。

Step 2. ドリブル機構を作動させる。

Step 3. もし $\mathrm{Line_A}$ 上に障害物が存在するならば、障害物の中心が $\mathrm{Line_A}$ のどちら側か計算する。もし、それが出し手ロボットから見て左側にあるならば、Dir を 1 とする。そうでなければ、Dir を-1 とする。もし、$\mathrm{Line_A}$ 上に何も障害物が無ければ、Dir を 0 とする。

[Pass Algorithm]

Step 1. [Initializing]を行う。

Step 2. もし Dir が 0 でないならば、 PassCounter を 0 とし、Step 5 へ。

Step 3. $\mathrm{Line_B}$ に沿って 5 cm/sec で移動する。

Step 4. もし $|\theta|$ が 3°より小さく、かつ、ボールとパスの出し手ロボットの距離が 10cm より小さいならば、PassCounter を 1 増やす。もし、 PassCounter が 20 より大きく、かつ、 $|\theta|$ が 1°より小さいならば、キック装置を起動させる。そして PassCounter をリセットする。

Step 5. 次のタイムステップを待つ。そして Step 1 へ。

[Receive Algorithm]

Step 1. [Initializing]を行う。

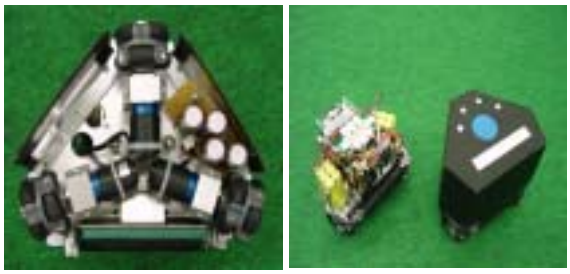
Step 2. もし Dir が 0 でないなら、 $\mathrm{Line_B}$ から Dir × 70 °の方向へ、60 と 7 ×（ボールと $\mathrm{Line_A}$ の間の距離）の小さい方の値（cm/sec）で進む。

Step 3. 次のタイムステップを待つ。そして Step 1 へ。

4. ロボットメカニズム

4．1　ロボットメカニズムの概要

連携プレーを行うためには、メカニズムの面でも、移動、シュート、ドリブルなどの命令を確実に実行できる構造になっていなければならない。Figure 9 に筆者らのロボットシステムを示す。ロボットは 3 つのオムニホイールを持ち､全方向に移動可能である。Figure 9(a)に示すように、3 つのエンコーダー付き DC モータ（FAULHAVER）を用いている。ギア比は 9.7:1 である。キック装置にはソレノイドを利用している（Figure 9(b)）。最大約 3m/sec のシュートが可能である。カバーを外した Figure 9(b)に見られるように、ドリブルは回転ローラーを利用してボールにバックスピンを与えることによって実現している。



(a) omni-directional drive　　(b) appearance of our robot

Figure 9: Our robot system

パスプレーを実現するためには、技術的に達成すべき 2 つの大きな課題がある。一つは、ロボットとボールの正確な位置・方向の検出である。もう一つは、ボール捕捉のための機械的な仕組みである。前者については、２．で述べた高速かつ高精度な画像処理システムによって解決している。後者については、ドリブル機構にショック吸収機構を組み込むことで解決している。コーネル大学の BigRed チームのように、スプリングを用いた方法は自然であるが、筆者らのチームでは別の簡単な手法で解決を図っている。

ドリブル装置は、Figure 9(b)に示すように、ロボットの前面に取り付けられている。ゴム製のローラーが回転し、ボールにバックスピンを与えることによりボールを引き付けている。ゴム製であるが、ローラーはボールを噛まない程度に硬い。硬過ぎるとボールは簡単に跳ね返ってしまう。そこでローラー取り付け部にあそびを持たせ、上方に僅かに動くようにしている。単純な機構であるが十分に衝撃を吸収できている。しかし、吸収の程度は床材にも依存するので、あそびの量は調節できるようにしておく必要がある。

4．2　キャッチ機構のモデリング

一般的に、衝撃吸収メカニズムをモデル化するためには様々な要因があり容易ではない。筆者らのシステムでは、ドリブル装置にあそびを持たせることによって衝撃を吸収している。このメカニズムを、次の 3 つの状態、すなわち、(a)衝突の瞬間、(b)定常状態になるまでの遷移過程、および、(c)定常状態、に分類してモデル化し、解析を試みる。

(a) 衝突の瞬間

Figure 10 に示す X－Y 座標系において、ローラーとボールの中心を結ぶ線分と水平軸がなす角を θ、ローラー、ボールの質量と各々の重心の速度を m_R 、 m_B、 v_R 、 v_B とする。また、ローラーとボールの衝突点に水平、垂直にとった x－y ローカル座標系における v_R と v_B の成分を、各々、$(v_{R,x}, v_{R,y})$、 $(v_{B,x}, v_{B,y})$ とする。

ローラーとボールの回転がない場合の衝突をモデル化すると、運動量保存則より、y-軸方向に、

$$v'_{B,y} - v'_{R,y} = -e_y (v_{B,y} - v_{R,y}) \qquad (1)$$

$$m_B v_{B,y} + m_R v_{R,y} = m_B v'_{B,y} + m_R v'_{R,y} \qquad (2)$$

が成り立つ。ここで、「 ´ 」がついた変数は衝突後の値を表す。e_y は y-軸方向の跳ね返り係数である。 x-軸方向についても同様に、

$$v'_{B,x} - v'_{R,x} = e_x (v_{B,x} - v_{R,x}) \qquad (3)$$

$$m_B v_{B,x} + m_R v_{R,x} = m_B v'_{B,x} + m_R v'_{R,x} \qquad (4)$$

が得られる。これらは、ボールとローラーの衝突を質点と見なした場合のモデルである。

実際にはローラーもボールも回転しているので、角運動量についても考慮する必要がある。そこで、慣性モーメントと角速度を各々 I 、 ω とすると、角運動量保存則より、

$$I_B \omega_B + I_R \omega_R = I_B \omega'_B + I_R \omega'_R \qquad (5)$$

が成り立ち、また、衝突点では、

$$r_B \omega_B = v_{B,x}\ ,\ r_R \omega_R = v_{R,x}\ ,$$

$$r_B \omega'_B = v'_{B,x}\ ,\ r_B \omega'_R = v'_{R,x} \qquad (6)$$

となる。ここで、r_B, r_R は各々、ボールとローラーの半径である。ローラーはゴムでできた円筒状なので、慣性モーメント I_R と I_B は、 $I_R = \frac{1}{2} r_R^2 m_R$ 、 $I_B = \frac{2}{5} r_B^2 m_B$ となる。衝突の瞬間の $v'_{R,x}$, $v'_{R,y}$, $v'_{B,x}$, $v'_{B,y}$ の解は、式 (1)～(6)を解いて得られる。

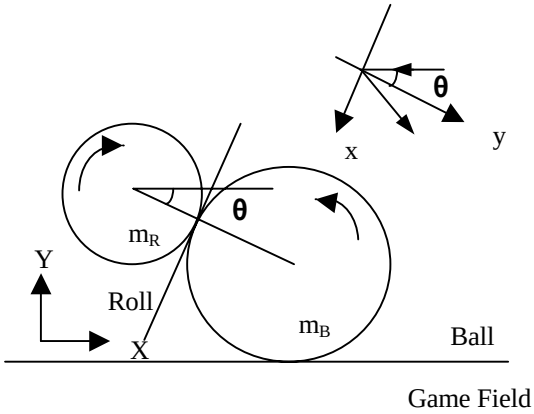


Figure 10: Physical situation at a the impact

(b) 定常状態までの遷移過程

衝突の後、ボール（の重心位置）は Figure 11 に示したように反射を繰り返す。もし、定常状態になるまでの遷移過程においてθが僅かしか変化しないならば、重心の動きのみを考慮すれば十分である。詳細な解析とモデル化は今後の課題であるが、もし、$x-y$ ローカル座標系において $e_x \cong e_y$ が成り立つならば、モデル化は容易である。

Figure 11 に示すように、反射する角度はθ, 2θ, 3θ, ..., nθ,.....のように跳ね返る毎に増える。n回の反射の後に $\pi/2$ を超え、ボールはもと来た方向に返ることになる。Figure 12 はθ＝$\pi/4$ の場合である。このときは、ボールは短時間の間に跳ね返り、もしローラーが衝撃を吸収するために図の矢印で記した方向に動けば、定常状態に移るまでの遷移過程において角度関係は保たれる。

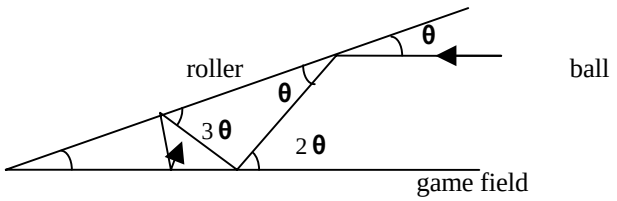


Figure 11: Reflections of ball in the transition

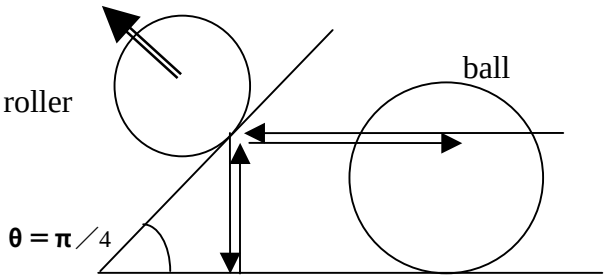


Figure 12: An example of shock absorption at θ＝π/4

(c) 定常状態のモデリング

Figure 13 に示すように、垂直抗力を N_B、 N_R 、動摩擦係数を μ_1、 μ_2 とすると、ボールの運動方程式は、

$$\text{X-direction: } -\mu_1 N_B + N_R\cos\theta - \mu_2 N_B \sin\theta = 0 \qquad (7)$$

$$\text{Y-direction: } N_B - m_B g - N_R \sin\theta - \mu_2 N_R \cos\theta = 0 \qquad (8)$$

となる。式(7)、(8)を解くと N_B, N_R は次のように得られる。

$$N_R = \frac{\mu_1 m_B g}{-(\mu_1+\mu_2)\sin\theta + (1-\mu_1\mu_2)\cos\theta} \qquad (9)$$

$$N_B = \frac{(\cos\theta - \mu_2 \sin\theta) m_B g}{-(\mu_1+\mu_2)\sin\theta + (1-\mu_1\mu_2)\cos\theta} \qquad (10)$$

したがって、ボールのバックスピンによる力 f は、

$$f = \mu_1 N_B = \frac{\mu_1(\cos\theta - \mu_2\sin\theta) m_B g}{-(\mu_1+\mu_2)\sin\theta + (1-\mu_1\mu_2)\cos\theta} \qquad (11)$$

$$= \frac{\mu_1(1-\mu_2\tan\theta) m_B g}{1-\mu_1\mu_2 - (\mu_1+\mu_2)\tan\theta}$$

となり、$\frac{\partial f}{\partial\theta} > 0$ および $f\big|_{\theta=0} > 0$ より、θが大きくなるにつれて、より大きな力 f が得られることが分かる。

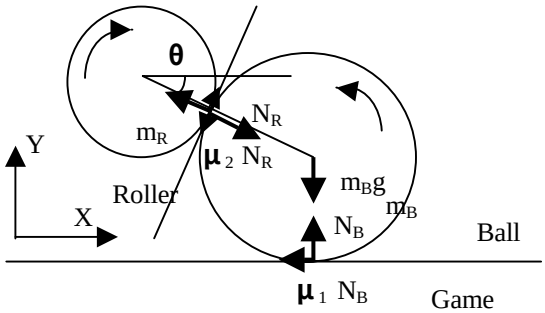


Figure 13: Stable model by dribbling device

5. 考察

メカニカルな機構の観点から、ドリブル機構とキック機構の有効性を確認した。フィールドのカーペットの動摩擦係数が大きい場合、およびフィールドの壁の辺りを除き、ドリブル機構は有効に作動した。

画像処理の観点からは、実験的に精度を確認した。実験結果の一部を Table 1 に示す。実験は蛍光灯で照明された実験室環境のフィールド上に白熱灯を 2 基追加して実際の試合会場に近い環境を作って行なった（結果的にフィールド面上の照度は 450～800Lux）。

静止 ID 検出率取得実験では、ロボット 5 台をフィールド上のフリーキックマーカ及びセンターに配置し、静止状態

で 10,000 回各ロボットの ID を取得した。実際に配置した位置から 10cm 以上離れていれば誤認識とみなして処理結果を整理した。動的 ID 検出率取得実験では、ID0 のロボットにリサージュ曲線を描かせ 10,000 回ロボットの位置を取得し、そのデータを解析することによって認識率を計算した。位置取得精度測定実験では、フィールド上の各フリーキックマーカ、センター、ペナルティーキックマーカの 9 点において、静止状態でロボットの位置を取得し、その変位から最大誤差を得た。角度取得精度測定実験では、上述した 9 個所の地点において、それぞれの点でロボットを 0°、45°、90°、135°のそれぞれの方向に向け、それぞれ 1,000 回ずつ計測を行い、その変位から最大角度誤差を得た。ここで、フィールド上の位置や角度による特異な差は特に認められなかった。処理速度測定実験では、フィールド上にロボット 10 台、ボール 1 個がある状態で画像処理システムのみの速度を 1,000 回計測し、その平均を得た。

ボール、チームカラーマーカー、サブマーカーなどの検出対象物の認識率は、実際のゲーム中のデータを基に評価した。ルールでは 700Lux～1000Lux と規定されているが、現実的にはこの値から外れる場合もある。また、フィールド上の位置によっても変わり得る。筆者らのシステムは、200Lux の照明環境下でも動作することが確認できている。

画像処理システムは、ゲーム開始直後、ある期間完全に隠蔽状態になった場合、およびボールのシュート速度が 5m/sec を超えた場合に全画面探索を行う。全画面探索を行った場合でも、処理時間はおよそ 20msec/frame である。探索範囲限定を行う場合はおよそ 1.4msec/frame である。したがって、ロボットの陰からボールがシュートされるような場合にも十分に対処可能である。

Table 1 Experimental results of image processing performance

Experimental item	Planned	Result
ID recognition ratio in a still mode [%]	100.00	99.68
ID recognition ratio in a dynamic mode [%]	100.00	99.61
Maximum deviation for the position [cm]	0.50	0.097
Maximum deviation for the angle [deg]	1.00	0.78

6. まとめ

本稿では、ロボカップ小型リーグで複数のロボットが協調動作を行う例としてパスプレーを取り上げ、それを実現するためのロボットの制御アルゴリズム、キック機構とドリブル機構、および、高速かつロバストな画像処理アルゴリズムについて紹介した。ロボットに対する画像フィードバックシステムの動作を 1/60 秒ごとに行うことにより、3.0m/sec のボールに反応できる高速な画像処理を実現した。

画像処理の観点からは、自然光のように幅広い光スペクトルが含まれている場合、境界部分に色にじみが発生する問題があるため、光学的な色収差によるオブジェクトの誤検出への対処が必要となる。また、スポットライトのように、明るさのダイナミックレンジが広すぎると画像処理が破綻するため、今後は明るさの変化や種々の光源に対してよりロバストな画像入力システムに改良する必要がある。

行動計画の観点からは、前もって敵の戦略を知ることができないため、短時間で敵の行動パターンに柔軟に対応できる戦略が必要となる。敵の行動パターンを学習して弱点を分析するシステムの組み込みが課題である。

ソレノイドを用いたキック機構は、現在、一定の強さのシュートしか放てない。パスを確実にするためにも、キックの強さを可変にすることが必要である。今後、ソレノイドに与えるエネルギーを変化させることによって、キックに強弱をつける方式に改良する予定である。

現在、筆者らのロボットの最高速度は 1.2m/sec であるが、4 輪に増やしてロボットの高速化を図っている。障害物回避、経路生成法、行動生成法などのソフトウェア的な改良、および、ハードウェアとソフトウェアのバランスの検証も今後の課題である。

参考文献

[1] http://www.robocup2002.org/

[2] M.Veloso, E.Pagello, and H.Kitano (Eds.)：”RoboCup-99：RobotSoccerWorldCupIII”, Springer (June 2000).

[3] P.Stone, T.Balch and G.Kraetzschmar (Eds.)：”RoboCup 2000:Robot Soccer WorldCupIV”, Springer (March 2000).

[4] A.Brik, S.Coradeschi, and S.Tadokoro (Eds.)：”RoboCup 2001:Robot Soccer WorldCup V”, Springer (March 2002).

[5] G.A.Kaminka, P.U.Lima and R.Rojas (Eds.)：”RoboCup 2002:Robot Soccer WorldCup VI”, The 2002 International RoboCup Symposium Pre-Proceedings, Fukuoka, (June 2002).

[6] S.Hibino, Y.Kodama, T.Iida, K.Kato, S.Kondo, K.Murakami, and T.Naruse：”System configuration of RoboDragons team in RoboCup small size league”, Proc. of SI2002, Kobe, (Dec.2002).

[7] Y.Kodama, S.Hibino, K.Murakami, and T.Naruse：”Small Robot Detection by Using Image Processing and its Application to Action Planning and Action Analysis”, Proc.of MIRU2002, Vol.1, pp.223-228, Nagoya, (July 2002).

[8] S.Hibino, Y.Kodama, Y.Nagasaka, T.Takahashi, K.Murakami, and T.Naruse：”Fast image processing and flexible path generation system for RoboCup small size league”, The 2002 International RoboCup Symposium Pre-Proceedings, pp.45-57, Fukuoka, (June 2002).

[9] Bruce, J., Balch, T. and Veloso, M.: “Fast and Inexpensive Color Image Segmentation for Interactive Robots”, Proc. of IROS ’00, pp. 2061-2066 (2000).

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-10 (5/2)

An architecture that unifies virtual and real environment

Kentaro Oda, Takeshi Kato, Toshiyuki Ishimura, Takeshi Ohashi

Dept. of Artificial Intelligence, Kyushu Institute of Technology
ken@mickey.ai.kyutech.ac.jp

Abstract

Development in real multi-agent system like robotic soccer in the RoboCup leagues is inefficient because of real world uncertainty. In this paper, we introduce an architecture we developed in the Sony Four-Legged League 2002. This architecture designed to unify the real and virtual environment to accelerate multi-agent system development and scientific research for the communities not only for the legged league one. The architecture consists of the abstraction layer, common communication protocol, communication hub, and simulator named open robot simulator. We implemented it with standards Java/XML technologies.

1 Introduction

Development in real multi-agent system like robotic soccer in the RoboCup leagues is inefficient because of real world uncertainty. Considering environment changes, for example, the changes of the lighting condition affect robot's behavior seriously in general. In this case, it is difficult to find out where the problem occurred - problems in vision module or planning module and so on. In order to locate the problem, generally, we debug the target system with the introduction of monitoring facility like logging. After fixing the system, But how we can verify whether the problem is fixed. Reproducing the exactly same situation is almost impossible in real world because of sensory noise. In multi-agent system, obviously this problem becomes more complex. To focus on scientific problem, we must distinguish a discovered problem is just a programming error or an inherent one. As a matter of the fact, we experienced a situation in which a programmer discovered an odd robot behavior and convinced the problem is fixed with consuming 10 hours, but actually the modified source code had not been compiled at all. This clearly shows that how real world development is unreliable and uncertain one. In the previous case, later we found the problem is in the vision code through real-time monitoring. AI oriented simulators like SoccerServer [Noda, 1998] may not help in this case. Here, it is necessary that a development environment unify both simulation and real environment. With unified environment, not only we can run the same robot program on a real robot and virtual simulated robot, the same monitoring tool can be used for both environments. This may accelerate the development and scientific research.

2 Architecture Design

The main goal of proposed architecture is to unify the two environments - real and virtual (simulated) environment. With this abstraction, the same robot program can be run under the simulated and real environment. To introduce this abstraction, we prepared the environment abstraction layer on the top of robot dependent API such as OPEN-R and our virtual robot API. Before illustrating a specific part of proposed architecture, we give the policy of overall our architecture as follows.

- Openness - each part of the system must be expandable to introduce new robots, new planning strategy, new color models, new collision detection module, new image processing module, etc.
- Distribution in a Network - Since individual robots exist distributed in a physical space, the architecture must support distribution in a network.
- Rich Information - Provide rich information not only for programmers but robots as much as possible in arbitrary abstraction level, to facilitating machine learning, debug, coordination strategy etc. For example, information from a joint angle to an exact global position (x, y, θ) in a virtual soccer field is necessary.
- Rich Debugging Facility - debug supporting functionalities such as taking a snapshot of virtual / real environment which contains overall information, no interruption of disconnection/connection of agents etc are necessary.

- Heterogeneous Architecture - dependency on specific OS, programming languages, communication protocols must be minimized.

- Minimize the Requirements - requirements migrating existing programs to this architecture must be minimized.

Under this policy, we developed the architecture and implemented it with standard/open technologies such Java and XML.

3 Proposed Architecture

The main idea to unify the both real and virtual environment is the introduction of an environment abstraction layer, a common communication protocol, commucation hub and a vision-based simulator. We also developed the monitoring tool to visualize local information acquired from virtual and real robots. Fig. 1 shows an example of configuration both virtual and real robots resides on at the same time. In this configuration, the virtual robot can be simulated in the simulator, but also with the simulator, the real robot can be visualized so as to its estimated position is reflected to the virtual environment. This allows us simulate the environment based on synthesized values and real-time sensory values. After this section, we illustrate the each part of the architecture.

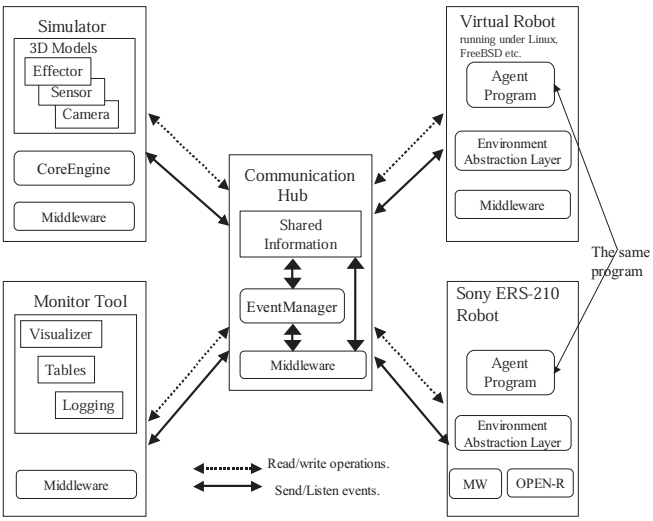


Figure 1: The Architecture Overview

3.1 Communication Hub

The communication hub holds whole environmental information including from each robot's estimated position in the soccer field to camera images generated / acquired through the simulator or a real robot camera.

- Tree structured: Environmental information is provided as external representation so as to programmer can get information user friendly: it is tree structured as shown in Fig. 2.

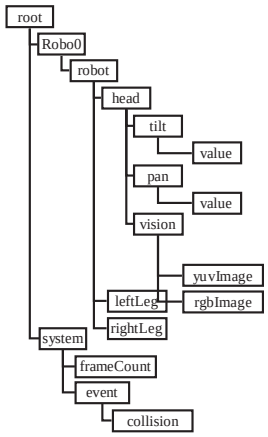


Figure 2: A Tree Representation Example of Environmental Information

```
<?xml version="1.0" encoding="UTF-8" ?>
<Environment version="0.1" xmlns="http://www.asura.ac/xethereal">
  <Leaf name="/Robo0/robot/head/tilt/value" type="double" scalar="true">
   <i>32.5</i>
  </Leaf>
  <Leaf name="/Robo0/robot/head/pan/value" type="double" scalar="true">
   <i>11.4</i>
  </Leaf>
  <Leaf name=" /Robo0/robot/leftLeg" type="double" scalar="false" length=3>
   <i>1.5</i> <i>131.4</i> <i>231.1</i>
  </Leaf>
  ....
</Environment>
```

Figure 3: An Example XML Output of Environmental Information

- Snapshot: The hub also can take a snapshot of its contents. The snapshot is XML formatted so that programmers can modify and export. Fig. 3 gives an example of the XML output result.

- Dynamic connection/disconnection of clients: This is one reason why we introduce the hub. In development cycle, we frequently kill/run the clients. Using this hub, the system can continue the execution regardless of disconnection because the last state of a robot is still hold even if it was crashed.

- Dual Communication: We provide two communication mechanisms - one is synchronous operations: `read()` and `write()` operations to the tree. These operations are fairly simple. To read value on the path name of /robot1/camera/yuvImage is `read("/robot1/camera/yuvImage")`. It returns a byte array container. To gain the performance, bulky versions of these operations are provided. The other is an asynchronous event mechanism which supports send and listen event operations. All events must be sent with a string label to address the contents. The event receiver can set a filter to receive interested events and reduce bandwidth by specifying a regular expression. Only events matches the regular expression can be received.

A communication pattern may be used in common is like that 1) a client receives an update event, and 2) is-

sues read() operations to get its interested nodes, 3) send an finish read event. 4) After receiving the finish read event, a client who wants to write data start to write the nodes through write() operation. After that, 5) it sends an update event. The tree held in the communication hub works as a shared memory. Asynchronous event mechanism can be used as a synchronization mechanism between data producer and consumer.

3.2 The Environment Abstraction Layer

The Environment Abstraction Layer (EAL for short) is provided for two purposes. One is to hide details on a underlying layer and the other is to give an illusion of a modeled environment. An example of the modeled environment is the Sony four-legged league soccer environment. Fig. 4 shows an example of classes for which an agent program interacts with the modeled environment. This layer may be designed in an application specific way. It lies on the top of the platform dependent API layer such as OPEN-R API and middleware for communication. The responsibility of this layer is to correctly map the operations of a modeled environment to one in its underlying layers. For example, an implementation of the EAL in the simulated agent may become a set of code invokes the communication hub with read()/write() and send() event operations to reflect its environment information correctly. On the other hand, while providing the same interface for an agent program, the EAL in Sony ERS-210 actually responsible for getting information through image processing, self localization, locomotion modules etc. Thus, the EAL definitely defines the mapping of a modeled environment to actual information sources/consumers. Implementation of this layer is application specific and user responsibility.

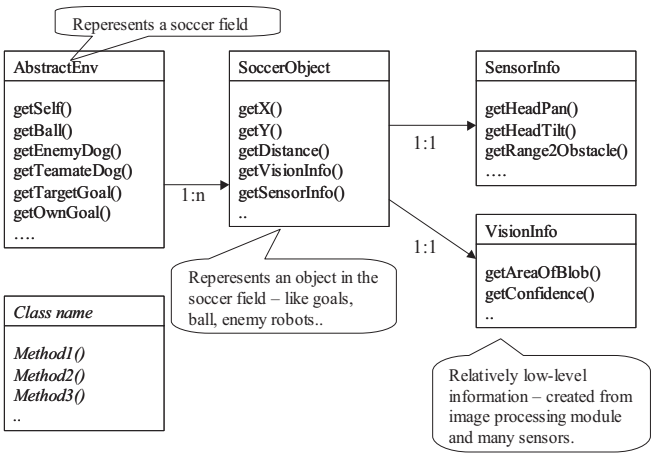


Figure 4: An Example Classes Diagram of The Environment Abstraction Layer

3.3 Middleware

The middleware provides the interface of read()/write() and sendEvent()/receiveEvent() operations. The middleware implementation encapsulates floating point representation and byte ordering. At this time, Java and C++ version of middleware is implemented.

3.4 Simulator

The simulator is designed as vision based simulation. This means that the simulator takes the several control values as inputs and generates a synthesized image as an output. These input and output comes in/out from the communication hub through accessing middleware. Fig. 5 shows a snapshot of running the application. It consists of global, command line, tree, and local camera views. In our implementation, the simulator takes five odometory control values like forward[cm/cycle], left[cm/cycle], rotate[degree/cycle], and tilt-head[degree/cycle] and pan-head[degree/cycle] from an agent. Virtual robot acquires the synthesized image from the communication hub and sends back these control values. Accordingly, the simulator, communication hub and virtual robot forms loop in simulation. This simulator has several useful features - plugin support, lisp like simple scripting, several views etc.

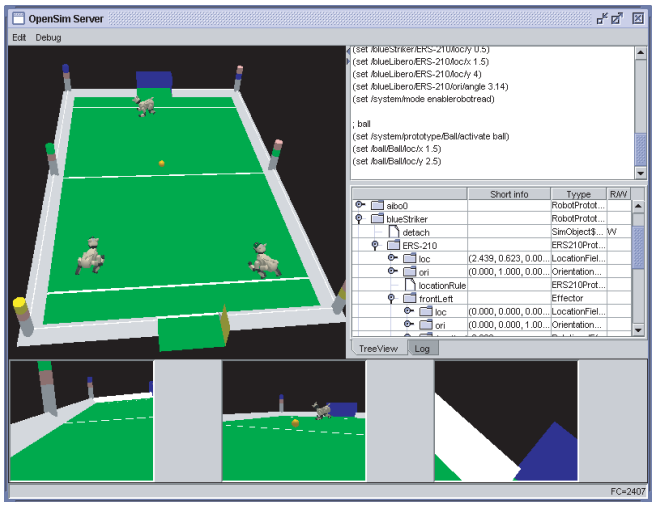


Figure 5: A Screen Snapshot of The Simulator

4 Implementation

To implement the communication hub, simulator and monitoring tool, we use Java, Java Advanced Imaging API and Java 3D API. For generating/parsing XML file in the communication hub, we use the SAX parser. For an agent programming we use C++ and partially scheme, a lisp dialect. The agent program can be run both Sony ERS-210 platform and Linux and FreeBSD with gcc.

5 Conclusion

With this architecture, we convince successfully introduce heterogeneity by using standard Java, distribution in a network with TCP/IP, minimized the requirements through simple communication operations, providing rich information through the tree structured external representation, openness with communication hub and plugin facility in the simulator and rich debugging facility with the communication hub.

In the four-legged league in RoboCup 2002, our system had not been implemented. Now we have a good fundamental to experiment new planning and coordination strategy and so on. At this time, the primary implementation has been done. As a future work, we have to evaluate the system in practical. In particular, the effectiveness of combining information real and virtual robot have to be evaluated.

The authors would like to express our gratitude to the Information-technology Promotion Agency (IPA), Japan for sponsor of this project and Professor Takaichi Yoshida for supporting.

References

[Oda, 2002] Kentaro Oda, Takeshi Ohashi, Takeshi Kato, Toshiyuki Ishimura, Yuki Katsumi: The Kyushu United Team in the Four Legged Robot League in RoboCup 2002 Robot Soccer World Cup VI, page. 452, 2002.

[Noda, 1998] Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, Ian Frank: Soccer Server: a tool for research on multi-agent systems, Applied Artificial Intelligence 12, pp.233-250, 1998.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-11 (5/2)

An Open Robot Simulation Environment

Toshiyuki ISHIMURA, Takeshi KATO, Kentaro ODA, Takeshi OHASHI
Dept. of Artificial Intelligence, Kyushu Institute of Technology

isshi@mickey.ai.kyutech.ac.jp

Abstract

Nowadays various kinds of robots are publicly available such as AIBO, ASIMO and so on. However, the development of robots is still difficult because of their complexity, continual changes of environments, limitation of resources and so on. To solve this problem, robot developers often use the simulator that allows to program and test robots' program effectively in the ideal environment where specified conditions can easily be reproduced. It is still difficult to realize the simulator regardless of its usefulness, because the cost of simulator implementation seems the unexpected cost in the development of robots. To overcome this problem, it is need to realize the open robot simulation environment in which any kind of robots can be simulated. This paper focuses on vision-based robot simulation environment and describes a method to construct it. Finally, we implemented a simulator for Robocup Sony 4-Legged League by this method.

1 Introduction

Nowadays various robots are publicly available such as AIBO, ASIMO and so on. However, the development of robots is still difficult because of their complexity, continual changes of environments, limitation of resources and so on. Considering environment changes in vision-based robot, for example, the changes of the lighting condition affect robot's behavior seriously. To clear problems the robot's strategies in the real environment, it is need to check strategies in exactly same environment, because in each testing time sensory values such as camera images and the effectors will change.

To solve this problem, there are the robot simulators that are categorized two types; one aims to simulate the robot mechanical behavior with accurate robot model data, the other aims to simulate the vision of the robot in order to test robot strategy. The simulator allows developers to program and test robots' program effectively in the ideal environment where specified conditions can easily be reproduced. It is still difficult to realize the simulator regardless of its usefulness, because the cost of simulator implementation can be seen the unexpected cost in robot development. To overcome this problem, it is need to realize the open robot simulation environment in which any type of robots can be simulated. This paper focuses on vision-based robot simulation environment and describes a method to construct it. The next section describes concept of open robot simulation environment. Section 3 shows design of architecture. Section 4 shows an implementation of the simulator for Robocup Sony 4-Legged League. Finally, section 5 concludes with a discussion of our method.

2 What is Open Robot Simulator

The purpose of our simulation environment is to improve the efficiency of the development of vision-based robots' strategies in particular. When testing the robot strategy by simulation, each developer would prepare the simulator in which accommodate a single kind of robot. However, each simulator has something in common such as synthesis of the camera image, effector simulation, 3D object management and so on. Thus, we provide a general simulation environment to simulate any kind of robots and customize according to any testing condition. To archive this environment, we give the policy as follows:

- Openness that means the simulator reveals itself to robot developer
- Reproducibility of simulation
- Minimum modification of robot program
- Useful test and monitor tool

The first one is our main subject. With high openness, developers can manipulate the simulation environment easily to treat various kinds of robots and customize it according to real environment. Second, reproducibility of the simulation environment is very important in order

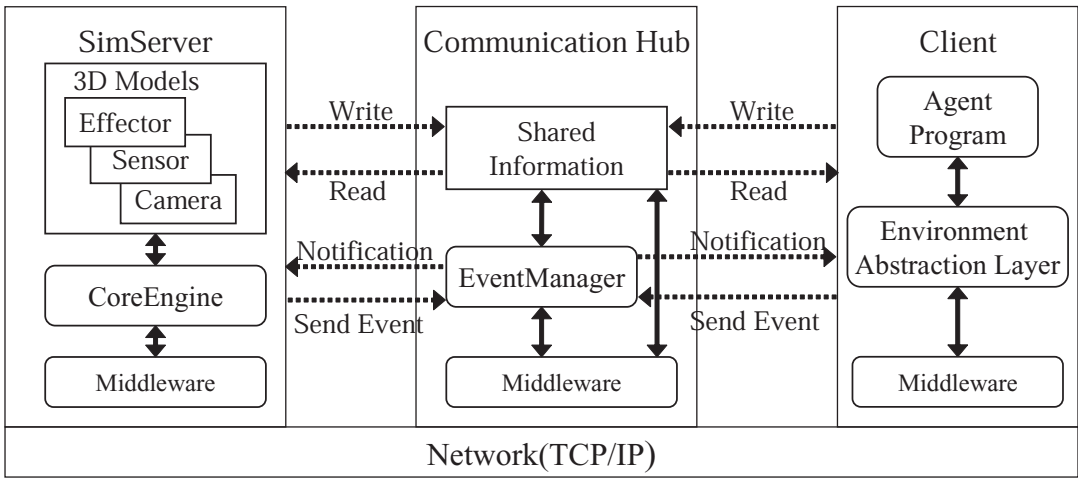


Figure 1: The Simulation Environment Architecture

to test the robot strategy correctly because real robot always affects the change of the environmental condition such as lighting condition. In development cycle of robot strategy, robot developers would repeat coding and testing of programs. When developers find out problems in certain situation, the simulator must support to reproduce that situation in short time. Third, to reduce the cost of simulation, the modification of robot's agent program must be minimized when migrating between real robot and the simulator. Finally, it is important that robot developers are able to monitor any information in the simulator to test the robot strategy effectively.

3 System Design

Figure 1 illustrates our system architecture, which adopts client/server model and introduces two servers as follows:

SimServer manages all objects in the virtual environment and synthesizes virtual camera images and simulates the robots' effector.

Communication Hub manages communication between the *SimServer* and clients and can capture messages through it.

In this model, multiple heterogeneous robot agents implemented in various program languages can participate to this environment because each client lies on the common communication middleware and connects the *Communication Hub* via the TCP/IP network. If the simulation environment communicates with real robots, it can accommodate both real and simulated robots in the same simulation environment. To reduce the cost of realization of the simulator, our simulation environment provides a class library of fundamental component as follows:

Robot holds cameras, effectors and its position and orientation.

Camera holds parameters such as view angle, resolution and so on, synthesized images by the Core Engine.

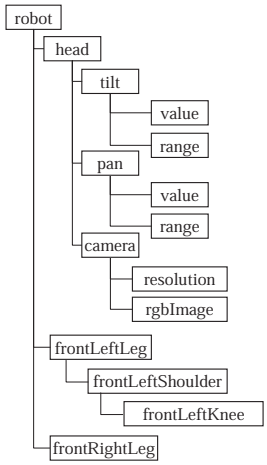


Figure 2: An example of object structure

Effector keeps current value and the range of value. It also can hold any sub-effector and camera as children.

They are implemented with opened interface in order to operate easily. The *SimServer* always watches information of each object then updates the virtual environment. This library allows users (i.e. robot developers) to construct the virtual robot by few steps; preparation of robot 3D shape model data, combination of several objects according to each real robot, adjustment of object parameter such as camera resolution and the range of effectors, and corresponding to model data.

The *SimServer* manages all objects in tree structure and provides name space according to that structure. The robot's agent program and developers can access all objects and its all attribute information by that name. Figure 2 shows a part of object structure that represents Sony ERS-210. In this structure, an agent program accesses the head camera image to indicate its name likes `/robot/head/camera/rgbImage`. This increases openness of the simulator and allows developers to test the robot strategy with the ideal information such as the location of robot. To customize the simulation environ-

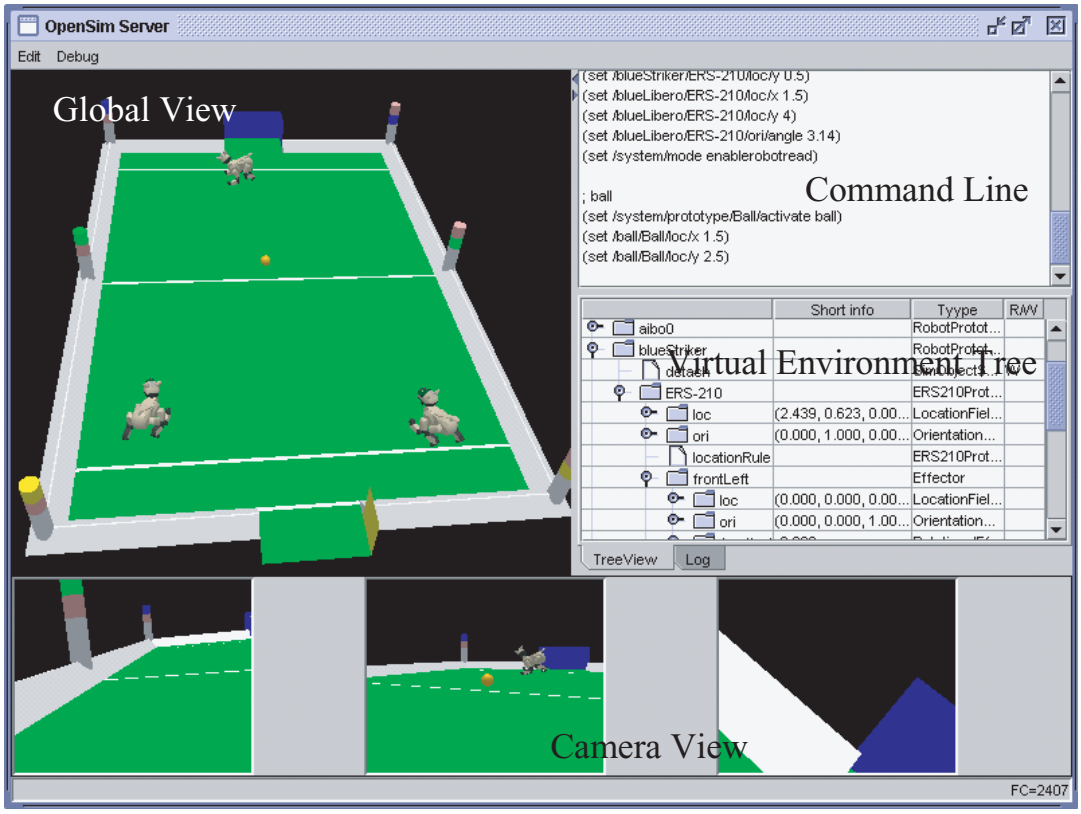


Figure 4: The overview of the *SimServer*

```
(let
  (set /Ball/loc/x
    (plus /ribo0/loc/x
      (/ (dist /robo0 /robo1) 2)
    )
  )
  (set /Ball/loc/y
    (plus /ribo0/loc/y
      (/ (dist /robo0 /robo1) 2)
    )
  )
)
```

Figure 3: An Example of script language

ment dynamically, any object in the *SimServer* can be appended and removed in runtime.

The *Communication Hub* supports synchronous and asynchronous communication to provide flexible communication. The *SimServer* and clients usually communicate by asynchronous operation (i.e. `read`/`write`) to improve communication performance. The event manager provides synchronous commutation by event sending and notification if needs to synchronize any information. The clients can connect and disconnect to the *Communication Hub* at any time in order to help development of strategy program and to provide fault-tolerance. Even if the client is disconnected, the *Communication Hub* and the *SimServer* keeps all information about the virtual robot. Consequently, developers can continue to examine the robot program in the same situation after fix problems.

Functionality The following functionalities enhance our simulation environment: plug-in user module, a script language, persistence of simulation environment state, communication among the agents and visualization of view frustum.

First, developers can insert fragment of program as plug-in module to the virtual robot in order to customize its behavior. This allows us to realize strange camera image and reduction of effector speed. Naturally, these modules can be appended and removed in runtime.

Second, to provide interaction between developers and the simulation environment, a script language likes the S-Expression is introduced. By using this language, developers can access overall object and its attribute information. It is possible to adjust the simulation environment such as lighting condition, simulation speed and so on. Figure 3 shows an example code to place a ball between two robots (named `robo0` and `robo1`). This functionality allows us to test the robot's strategy in the exactly same environment because the simulator can reproduce it repeatedly.

Third, the *Communication Hub* can store simulation environment state to a file in order to reproduce a certain situation. To this, the *Communication Hub* watches any information through it. After storing the file, developer ever reproduces that situation in any time.

Forth, the *Communication Hub* provides communication among the agents. With this functionality, it is possible to develop a strategy to collaborate the agents without using the real robots.

Fifth, when using the active camera, enhancement of camera motions is important to recognize the virtual environment. The simulator provides functionality to visualize the view frustum that each of the cameras is now seeing. This visualization is useful in order to tune-up camera motion.

4 Implementation

First, we implemented the *SimServer* and the *Communication Hub* in Java and Java3D[Java3D]. The *SimServer* shown Figure 4 consists of four components; global view, command line panel, the tree view, camera views. On the global view, user can change own view by mouse operations. The command line panel allows us to interact with the simulator by the script language described in the above. The tree view shows information about all objects in the simulation environment. We also implemented a library for communication between the server and the client, which supports C++ and Java.

Second, to evaluate our method, we implemented a simulator for the Robocup Sony 4-Leggued League on this environment. Then we succeed in migration from real robot's agent program [Oda,2002] to the simulator with a little bit modification. By using robot communication functionality of the *Communication Hub* , this simulator can simulate communication between robots instead of wireless LAN.

5 Conclusion

On the vision-based robot simulator, it is important to reduce cost of robot strategy programming. To achieve this, we proposed the open robot simulation environment to accommodate any kinds of robots. In this paper, it has been yet considered physical effects in simulation, nevertheless, we plan to introduce simple method by using collision detection.

The authors would like to express our gratitude to the Information-technology Promotion Agency (IPA), Japan for sponsor of this project and Professor Takaichi Yoshida for supporting.

References

[Oda,2002] Kentaro Oda, Takeshi Ohashi, Takeshi Kato, Toshiyuki Ishimura, Yuki Katsumi, The Kyushu United Team in the Four Legged Robot League, in Robocup-2002: Robot Soccer World Cup VI, page.452, 2002

[Java3D] Java 3D(TM) API Home Page. http://java.sun.com/products/java-media/3D/.

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-12 (5/2)

並列シナリオ記述を用いた RoboCup-Rescue Civilian の動作記述

篠田 孝祐 [1,2], **野田 五十樹** [1,2,3], **國藤 進** [2]
[1] サイバーアシスト研究センタ, 産業技術総合研究所,
[2] 北陸先端科学技術大学院大学
[3] さきがけ研究 21, 科学技術振興事業団.

Abstract

A framework to describe behaviors of civilian agents in RoboCupRescue simulations is proposed. In a disaster situation, it is hard to model civilian's behaviors as logically. Furthermore, we model the behaviors by "Scenario". In the proposed framework, the behaviors are divided into multiple scenarios for each goal, in which behavior-rules are grouped based on situations where the rules are active.

1 大規模社会シミュレーションにおけるエージェントモデル

大規模社会シミュレーションは，災害救助や経済活動など問題空間を論理的かつ明示的に表現することが困難な領域を対象としている．そのような領域を分析するための実証手法として，エージェントベースシミュレーションは一般的になりつつある[3, 10]．社会シミュレーション上にて動作するエージェントを作成する場合，その社会を構成する要素のモデルおよび振舞いの定義は重要な問題である．特に，主要素である人間の行動は多様であるだけでなく，集団となることでより複雑な様相を示す．エージェントの振舞いを規定する行動記述モデルは，シミュレーションの価値を左右する問題と言える．

社会シミュレーションにおけるマルチエージェントの意義とは，単なる一要素でしかない個々のエージェント同士の相互作用から創発されるマクロ事象を再現する点にある．従来の人間やロボットなどの行動記述に関する研究 (Soar[16, 13], ACT-R[1, 2], RAPs[4, 5]など) では，特定の行動目標に対する効率的な問題解決プロセスに焦点がおかれてきた．したがって，そこで開発された記述言語では，明確に定義された問題空間における最適な行動モデルの獲得や専門家の振舞いを獲得する過程の分析等が中心的課題であった．このような行動記述モデルは，災害時の避難経路演習シミュレーション等のように合目的的な最適行動プロセスの解明やエージェントの行動記述を対象としている．しかしながら，社会シミュレーションでは一見無駄に思える寄り道したという行動が自身や他の主体に影響を与え新たな現象を誘発する可能性がある．よって，社会シミュレーションにおけるエージェントには，最適行動プロセスの表現と同時に，合目的的には整理できない多様な振舞いも容易に記述できる行動記述の枠組みが必要とされる．

また，人間の振舞いは多様であり，それを再現するエージェントの行動は複雑になりやすく，必要とされる行動ルールの数は増加する．行動ルール数の増加はエージェントの行動決定に必要な計算時間コストの増加もたらしやすい．この計算時間コストの増加は，現実的な時間でのシミュレーションの観点から重要な問題である．さらに，社会シミュレーションにおけるエージェントの振舞いは．エージェントの本体の設計とは別に行動設計者としてのシナリオライタにより設計されることが多い．シナリオライタによるエージェントの行動設計には，より簡単な行動を表現するルールからテストを重ね，徐々に大きなルールを作り上げていけるプロトタイピングの形が適切だと考える．よって，エージェントの行動に関してプロトタイプイングのための変更・追加が容易な行動記述が必要である．

2 並列シナリオ行動記述

2.1 行動記述モデル

人間の行動は，知的であり多様であるだけでなく，集団となることでより複雑な様相を示す．そして，その振舞いは必ずしも合目的的に整理できるとは限らない．社会シミュレーションにおいて人間が示す様々な現象を適切に再現するためには，人間の多様な振舞いを的確にそして容易に設計できるエージェントフレームワークが必要である．大規模社会シミュレーションにおいて，人間の行動をモデル化するためのエージェントフレームワークに求められる要素として，我々は以下の 3 点に着目する．

(1) **熟考型行動と反射型行動の両方の特性を持つ行動モデル**
(2) **軽量実行**
(3) **変更，追加が容易な行動記述モデル**

これらの点を実現するために，本論文ではシナリオと呼ぶ行動ルールの集合を導入する．

2.2 シナリオベース行動記述

(1) の行動モデルに関して，社会シミュレーションにおけるエージェントの行動は，環境の変化に応じた反射型行動から，長期的なプランをもつ熟考型行動まで多岐にわたる．エージェントの行動モデルは，従来，反射型プランニング[4, 2]や熟考型プランニング[13]として表現されてきた．一方で，社会シミュレーションで扱う人間の行動は，長期的プランをともなう行動であっても熟考の上での行動というよりは，特定の目標の達成に必要な行動プランや習慣としての行動パターンをあらかじめ知見として持ち，それを状況に合わせて再現しているとみなせる．よって，社会シミュレーションにおけるエージェントの行動をモデル化には，長期間にわたる一貫性を持った行動を表現が容易である熟考型と，多様な環境への対処が容易である反射型の両者の特徴をもつ行動モデルが適していると考える．本研究では，行動プランや習慣としての行動パターンを**シナリオ**として記述し，人間の行動をシナリオに基づいてモデル化することで人間の多様な振舞いを表現する．このシナリオに基づく行動記述を**シナリオベース行動記述**と呼ぶ．

シナリオベース記述では，特定の目的の解決や習慣を表わす一連の行動をシナリオとして表現する．そのシナリオでは，Figure1 が示すように目標の達成に至る行動の段階を状況で切り分け，それら状況の遷移という形で表現する．

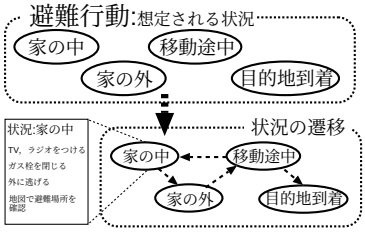


Figure 1: The example of scenario-chart

また，エージェントの行動全体については複数の目的の達成や習慣を同時に行なう様子は，複数のシナリオを同時に実行することで実現する．本研究では，このように複数のシナリオの並列実行を目的とする行動記述の枠組みを**並列シナリオ記述**と呼ぶ．

2.3 シナリオによる実行及び記述の効率化

(2) の軽量実行に関してシナリオベース記述では，シナリオにて長期的プランに基づく行動を状況の時間的遷移として事前に展開することで表現する．これにより，熟考に要する環境の同定やプランの展開などにかかる計算時間コストは最小限に押さえられ，行動決定を一段の推論として行なうことができる．さらに，並列シナリオ記述の枠組みにおける**状況遷移の直接的操作**をシナリオ作成の段階から行なうことで状況に適したシナリオのみが起動される．この操作により，エージェントの行動決定に必要な評価の対象となる行動ルール数を小さく保つことができ，結果として行動決定モジュールは軽量に実装することが可能である．

さらに，(3) の行動記述モデルに関しては，状況遷移を意図するシナリオの追加や削除の操作を明示的に記述することにより，新たな行動の追加や変更を行なうことが容易になると考える．エージェントに新しい行動を追加するには，シナリオとして作成した後，避難行動のシナリオに組み込む．そして適切な状況にてシナリオを起動するように追加する．

3 エージェントフレームワーク

3.1 Posit: Part Of SITuation

Posit は，特定の目的を解決するシナリオの途中段階を状況として切り出したものである．この *Posit* 間の遷移により，シナリオにおける目的達成の過程を表現する．各 *Posit* は，対応する状況で適用可能

な行動ルールの集合として定義される．

エージェントの行動設計者であるシナリオライタは，特定の目標を解決する過程をシナリオとし，切り分けられた個々のシナリオを状況および状況の遷移から問題解決のプロセスを構築する．このとき，切り分けられたある状況下において実行可能，もしくは実行すべき行動ルールを *Posit* に記述する．また，シナリオにおける状況の時間的遷移である *Posit* の遷移は，行動ルールのなかに明示的に記述される．

Posit は，具体的には．Figure2 に示す文法にしたがって記述する．*Posit* を構成するルールは，ルール名と環境条件 (Condition)，活性式 (Activation)，行動 (Action) の要素から成り立つ．各ルールは，Condition において内部情報を含めた環境情報に対する観測内容を記述する．その際，評価の対象となるルールに優先度はなく，対象となったすべてのルールの Condition は同列に評価される．Activation は活性式を表わし，Condition の観測内容を満たしたルールは Activation によりその活性値を決定する．エージェントは，この活性値を基にルールの選択を行なう[1]．Activation は単純な数値だけでなく，変数を含む関数として表現できる．つまり，シナリオライタは Action に示された動作に対する重要性を Activation を通じて環境状態を反映させることができる．Action には，シナリオの条件下 Condition の条件下でのエージェントの振舞いを記述する．この時，Action の具体的動作は *Posit* にて定義されるものではなく，エージェントにて定義されるものとする．

3.2 シナリオ，並列シナリオ処理

特定の目的解決プロセスにおける状況および状況の遷移により表わされるシナリオは，特定の状況を意味する *Posit* の集合と，それらの時間的遷移から構成される．エージェントの行動全体は，複数のシナリオの時間的重ね合わせとして表現し，様々な環境の変化への対応を図る．その際，シナリオ内における *Posit* の遷移，シナリオの起動および終了は，以下に述べる Posit oprator [2] によって行動ルールの Action に記述される．*Posit* におけるシナリオの操作は，以下の様な種類が用意される．

- (start_scenario S): シナリオ S の開始
- (end_scenario S): シナリオ S の終了
- (transit_posit A B): シナリオ内の *Posit*A から *Posit*B への遷移

並列シナリオ記述では，シナリオの起動・終了，そしてシナリオ内部での遷移の操作を Posit operator を用いて明示的に行なうことで表現する．これにより，複数の行動タスクを同時に実行する．我々はエージェントの行動全体を**並列シナリオ**として設計することで，対象とする現象に対する人間の適応的行動に関係する知識を最大限に活用する．そして，特定の目標に対する行動プロセスをシナリオベースとして表現することで，エージェントの複雑な行動をボトムアップに構築する．

3.3 PS^2:Parallel Scenario Problem Solver

PS^2 は，**並列シナリオ記述**の概念に基づいて記述されたエージェント行動を，エージェントの環境に基づいてシナリオを活性化，そのシナリオの流れにそって適切なルールを評価・選択する行動決定エンジンである．Figure3 は，PS^2 の情報とコントロールのフローを含めた PS^2 の基本サイクルを示している．

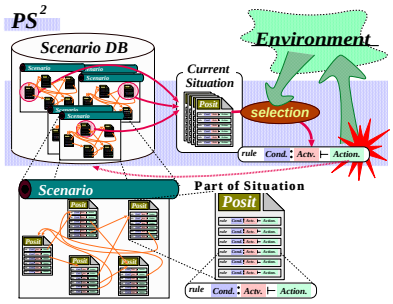


Figure 3: Process flow of PS^2

PS^2 のサイクルは，次のようになっている．

1. エージェントの起動時，すべてのシナリオは **Scenario DateBase(SDB)** に格納．
2. SDB 内にある起動シナリオの初期 *Posit* を **Current Situation(CS)** に格納．
3. CS 内のすべての行動ルールに対する評価
   (a) 行動ルールの Condition と入力された環境情報とを比較．

[1]現時点では Activation が最大となるルールを選択する方式を採用しているが，ルーレット選択など確率を用いるなどいくつかの選択方式が考えられる．

[2]Posit operator のプリミティブ操作は，各 *Posit* の活性化，不活性化を行なう add_posit，remove_posit である．上記のオペレータは (start_scenario S) {=(add_posit S:init)(end_scenario S) {=(remove_posit S:*) (transit_posit A B){=(add_posit B:init)(remove_posit A;*)} として実現される．このとき S:init は，シナリオ S の起動時の初期状態 init を，S:* は．終了時の *Posit* を意味する．

```
Posit      ::=  (defposit PositName Rule * )
Rule       ::=  (defrule RuleName  :condition  CondiForm
                                   :activation ActivForm
                                   :action     ActioForm )
CondiForm  ::=  ([LogicOP] CondiForm *)|(InSensor ...)
ActivForm  ::=  CalcForm
ActioForm  ::=  ([LogicOP] ActioForm *)|(OutSensor ...)
LogicOP    ::=  and | or | not | progn
InSenor    ::=  ⟨input sensor name with prefix "?"⟩
OutSenor   ::=  ⟨output sensor name with prefix "!"⟩
CalcForm   ::=  ⟨number⟩|Var|(CalcOP CalcForm CalcForm)
CalcOP     ::=  + | − | * | / |%
Var        ::=  ⟨variable name with prefix "*"⟩
```

Figure 2: Syntax of Posit

(b) 次に 3(a) を通過した行動ルールの Activation を評価しそれぞれの評価値を格納

4. 任意の選択ポリシーに基づいて実行ルールを選択
5. (4) で選択された行動ルールの Action を実行.
6. (5) によって SRS が変化した場合，適応可能な行動ルールがなくなるまで (3)-(5) まで繰り返す．[3] CS にルールがあり，適応可能な行動ルールが無くなった場合には，(3) に移動後，次の環境からの入力を待つ．もし CS 内に評価の対象となる *Posit* が無くなった場合には，エージェントの実行を終了する．

この PS^2 では，シナリオにて表現されたエージェントの行動を再現することで，シナリオの実行や *Posit* の活性化の前提となる環境の同定を事前に排除している．また状況の変化などにしたがって，シナリオ等の活性化をコントロールするために行動決定の対象となる行動ルールは全体のルールの一部にすぎない．

4 評価実験

我々は提案した並列シナリオ記述によるエージェントの行動決定への効果を評価するあたり一般市民エージェントを実装した．以下に示す Figure4 もっとも簡単な振舞いのルールをもとに振舞いを組み立ててある[4]．

Figure4 では，避難行動シナリオと避難所探索シナリオの２つのシナリオが用意され，避難所探索シナリオは避難行動 Posit が起動されたときに同時に起動される．なお，検証実験の環境は Table1 に示す通りである．

[3] ただし，エージェントの処理系によりルールの適応回数はリソースの制限が設けられる場合が考えられる．

[4] このルール記述の中では，細かい表記を省略し，文章にて表現している

```
((defposit "初期 Posit"
    (defrule "初期状態"
      :condition (?know "震災時室内にいる")
      :activation 1     :action "自分の家の場所を確認")
    (defrule "避難できるか?"
      :condition (?know "怪我がない")
      :activation 10
      :action "避難所探索，避難行動に遷移")
  (defposit "避難行動"
    (defrule goto_refuge1
      :condition (?know "避難所の場所")
      :activation 1     :action "避難所の場所に移動")
    (defrule "避難所に到着"
      :condition (?know "現在地が避難所")
      :activation 2     :action "避難行動を終了"))
  (defposit "避難所探索シナリオ"
    (defrule "近隣の避難所を探索"
      :condition (?know "近隣にある避難所の場所")
      :activation (- 50 *time)
      :action "避難所の場所を記憶し，避難所探索を終了"))
```

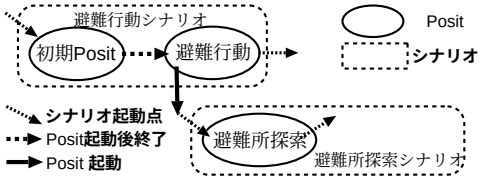


Figure 4: Example of Behavior:Simple Action

Table 1: 実験環境

OS	VineLinux 2.5CR
言語	gcc 2.95.3
CPU	PentiumXeon 2.4GHz x 2
Memory	2GB

4.1 行動ルール数とエージェント数の計算時間コストに与える影響

Figure5 は，シナリオのなかにあるルール数の違いによる 1 エージェントの計算時間コストの最大値の違いと，エージェント数による変化を示したグラフである．x 軸は同時に実行したエージェント数，y 軸は計算時間コスト [nsec] を示している．

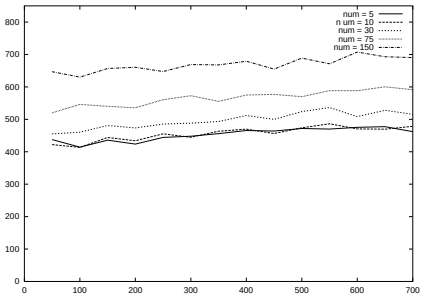


Figure 5: Difference of calculation time cost for each agent

Figure5 のグラフから，エージェント数が増加しても，1 体当りの計算時間コストはほとんど変わらないことが分かる．よって，ルール数の増加によるシステムとしての計算時間コストへの影響は微少である．

4.2 実行状態の比較

Table2 は，我々の実装した市民エージェントと同じフィールド上で動作するレスキューエージェント [14]のプロセスとの実行状態の比較である．表が示すように，互いのエージェント数は異なるが，CPU 占有率，平均処理時間ともにレスキューエージェントより軽量である．メモリの消費量に違いがあることを含め，プログラムの実装に依存するところも大きいが，この結果から市民エージェントは十分軽量で動作すると考える．

Table 2: 救助エージェントとの実行状態の比較

	CPU 占有率 (%)	メモリ占有率 (%)	処理時間 (msec)
A	5.5	10.2	344.5
B	52.3	0.1	439.0

A:一般市民 (150 ルール,500 体) B:救助部隊 (10 体)

5 考察

我々が提案した**並列シナリオ記述**は，2.1 にて大規模社会シミュレーションにおけるエージェント行動記述モデルに対する要求として以下の点をあげた

(1) **熟考型行動と反射型行動の両方の特性を持つ行動モデル**

(2) **軽量実行**

(3) **変更，追加が容易な行動記述モデル**

これらに対して

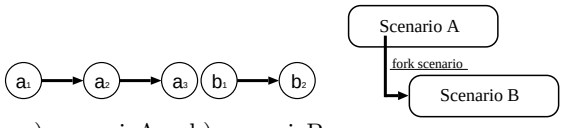
(a) **シナリオベース行動記述**

(b) **シナリオと状況によるルールの切り分けおよび状況の直接的操作**

(c) **並列シナリオとインタプリタ実行**

の 3 点を実現することで，社会シミュレーションにて動作するエージェントの行動を表現し実行する．

具体的には，(1) に対して (a) では，熟考型行動における一貫性のある行動プランをシナリオにおける状況の時間的変化として，反射型行動における状況変化への対応を並列シナリオとルールの並列評価による柔軟な制御により対応している．そして (2) に対して (b) により 4 で示したように，各エージェントの行動決定における計算時間コストの軽量化を図り，多数のエージェントが現実的な計算時間で動作することを確認した．最後に (3) に関しては (c) により，シナリオライタは Figure6 の様に，個々の行動目標に対するエージェントの振舞いをシナリオとして独立に設計可能できる．そして，Figure6c) の様に新たな *Posit* やシナリオを追加することでエージェントの行動を必要に応じて変更できるだけでなく，エージェントの動作を決定する為に評価する行動ルール候補を状況ごとに絞ることで，ルール評価プロセスにかかる計算時間コストの軽量化が行えるなどの利点を与える．また，*Posit* は特定の状況下での利用可能なルールの集合であるために，他のシナリオでの再利用可能などの利点がある．



a) scenarioA b) scenarioB c) add scenarioA to scenarioB on posit

Figure 6: Compound Sceanrio by *Posit*

6 まとめ

本論文では，大規模社会シミュレーションにおけるエージェントの行動記述に対して，**並列シナリオ記述**を導入した．並列シナリオ記述では，特定の行動目標の解決を目的とする行動プランや習慣などをシナリオと定義し，各エージェントの状況依存的振舞いをシナリオの重ね合わせとして表現する枠組みである．

並列シナリオ記述では，シナリオの起動及び終了はシナリオの記述の段階で明示的に記述する．これ

により，シナリオライタは設計するエージェントの行動を，行動目標ごとにシナリオを作成できるだけでなく，個々のシナリオの起動・終了を意味するシナリオ間の遷移を追加・削除するだけでエージェントの振舞いを変更することができる．さらに，並列シナリオ記述に基づいて記述されたエージェントの行動を，解釈し行動を選択するのが，行動決定エンジンである PS^2 である．PS^2 では，シナリオの起動・終了，そしてシナリオ内の遷移にしたがって，行動決定に必要な行動ルールの候補を小さくたもつことで，個々のエージェントの意思決定にかかる計算時間コストの削減を意図している．

この並列シナリオ記述に基づく大規模社会シミュレーションのためのエージェント行動記述への有効性の検証の一つとして，行動ルール数ならびにエージェント数に対する計算時間コストのシミュレーションに与える影響を実験した．実験では，エージェント数の増加による影響はエージェント数に対して単純増加であり，行動ルール数の増加の影響は少ないと示された．また，同じシミュレーション上における他のエージェントとの比較を行なった結果，多数のエージェントが動作しても現実的な時間で行動決定ができることが示された．よって，我々は**並列シナリオ記述**によるエージェントの動作記述は，大規模社会シミュレーションに適していると考える．

本研究は以下の点で改善が必要であると考える．

1. 状況の変化に対するシナリオの整合性の管理
2. ビジュアルプログラミングなどによるシナリオ設計環境

(1) に関しては，環境の変化に対する行動をシナリオ内にてあらかじめ展開するために場合によっては十分な対応が取れなくなり，状況と実行しているシナリオとの間に整合性が取れない可能性がある．よって，本研究で提案するフレームワークにおいても，GAEA にて開発されている状況の整合性チェックをする方法[11]を導入する必要がある．

(2) では，研究が対象とする社会シミュレーションでの利用を目指すのであれば，容易にエージェントのシナリオが設計・管理できることが理想である．それには，Squeak[6]などに代表されるビジュアルプログラミング環境が適切であると考える．以上の点を今後の課題として検討する．

参考文献

[1] ACT-R. *Web Page* `http://act.psy.cmu.edu/`.

[2] John R. Anderson, Michael Matessa, and Christian Lebiere. Act-r: A theory of higher level cognition and its relation to visual attention. *HUMAN-COMPUTER INTERACTION*, 12:430–462, 1997.

[3] Joshua M. Epstein and Robert Axtell. *Growing Artificial Societies*. MIT Press, 1996.

[4] Robert James Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, 1989.

[5] Robert James Firby. Task networks for controlling continuous processes. In *the Second International Conference on AI Planning Systems*, 1994.

[6] Dan Ingalls, Ted Kaehler, John Maloney, Scott Wallace, and Alan Kay. Back to the future –the story of squeak, a practical smalltalk written in itself. In *OOPSLA'97*, 1997.

[7] Interaction Design Language Q Web Page. `http://www.digitalcity.jst.go.jp/research/Q/`.

[8] Toru Ishida. Q: A scenario description language for interactive. *IEEE Computer*, 10 2002.

[9] Toru Ishida and Masahito Fukumoto. Interaction design language q : The proposal(in japanese). In *Proceeding of JSAI'01*, 2001.

[10] Hiroaki Kitano. Robocup rescue : A grand challenge for multi-agent system. In *Proceedings of the Third International Conference on Multi-Agent Systems(ICMAS-2000)*, pages 5–11, 2000.

[11] Hideyuki Nakashima and Itsuki Noda. Dynamic subsumption architecture for programming intelligent agents. In *Proceedings of International Conference on Multi-Agent Systems 98*, pages 190–197. AAAI Press, 1998.

[12] Hideyuki Nakashima, Itsuki Noda, and Kenichi Handa. Organic programming language gaea for multi-agents. In *Proceedings of International Conference on Multi-Agent Systems 96*, pages 236–243. AAAI Press, 1996.

[13] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[14] Rescue HP. `http://www.r.cs.kobe-u.ac.jp/robocup-rescue/`.

[15] RoboCup WebPage. `http://www.robocup.org/`.

[16] Soar Web Page. `http://ai.eecs.umich.edu/soar/`.

社団法人 人工知能学会
Japanese Society for
Artificial Intelligence

人工知能学会研究会資料
JSAI Technical Report
SIG-Challenge-0317-13 (5/2)

ヒューマノイドロボットの動的起き上がり運動の理解と実現

Dynamic Stand up Motion of a Humanoid

大賀 淳一郎 * **荻野 正樹** * **細田 耕** *† **浅田 稔** *†

Jun'ichiro OOGA*, **Masaki OGINO***, **Koh HOSODA***†, **and Minoru ASADA***†

* 大阪大学大学院工学研究科, † 阪大フロンティア機構

***Graduate School Engineerging, Osaka University, † HANDAI Frontier Research Center**

{ooga, ogino}@er.ams.eng.osaka-u.ac.jp, {hosoda, asada}@ams.eng.osaka-u.ac.jp

Abstract

In this paper, dynamic stand-up motion is realized by a humanoid robot. To avoid hard impact with the ground, and to overcome the limits of the joints, a round back is attached to the robot. Parameters of the back is determined by simulations since the motion analysis is quite difficult because of the nonlinearity, friction, and position control. An experimental result is shown that a real humanoid robot HOAP–1 equipped with the designed back can stand up dynamically.

1 はじめに

ヒューマノイドロボットは人間の生活環境においてさまざまな活動を行う存在としての期待が高まっている．人間の代わりとして，または人間と共同して仕事を行うためには安定した 2 足歩行をすることが求められ，これまでの研究では，倒れずに歩き続けることが注目されてきた．しかし不確実性の高い実世界においては，倒れないことよりも，倒れても起き上がることができることに注目すべきであると考えられる．

起き上がり行動を実現するために，静的につりあいをとりながら起き上がることも考えられる[1]が，動特性を生かして起き上がることができれば，より少ない自由度あるいはモータ出力で，起き上がり行動ができると期待される．動的な起き上がりに関しては，階層型強化学習による 3 リンク 2 関節ロボットの起立行動の獲得[2]や，脚の振り降ろしを利用した 4 リンク 3 関節の一脚ロボットによる起き上がり運動[3]などが報告されている．これらの方法は，平坦な背中での起き上がりを考えており，起き上がりの際に生じる衝突によるエネルギー損失が大きくなることが予想される．

本報告では，人間型ロボットの動的起き上がりを，平坦な背中のままで実現可能かどうかをシミュレーションによって実験し，起き上がり運動に必要な条件について考察する．さらに制限条件によって平坦な背中のままでは起き上がることのできないロボットに対しては，ロボットに丸い背中を実装することによって起き上がり運動を実現する．これは，丸い背中を装備することによって着地時の衝撃を減らすことができるため，より少ないエネルギーで起き上がることができると考えられるからである．背中の形状をシミュレーションによって検討し，実際のロボットに実装して，起き上がりを実現できることを実験的に確認した．

2 平坦な背中での動的起き上がり

起き上がり運動には多くの種類が存在するが，本報告では脚を振り降ろすときのモーメントを利用する動的な起き上がり運動の実現を目指す．

2.1 ロボットのモデル

ほとんどのヒューマノイドロボットの背中は平坦であるので，**Fig. 1** のようにヒューマノイドロボットをモデル化する．

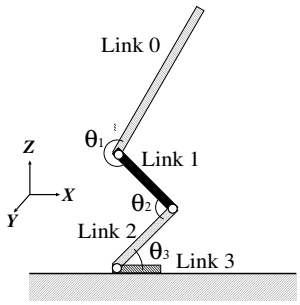


Figure 1: Robot model configuration. θ_i (i=1,2,3) indicates joint angles for hip, knee, and ankle, respectively.

ここで $\theta_i (i = 1, 2, 3)$ はそれぞれ腰，膝，足首関節の関節角である．また，進行方向を X 軸，鉛直方向を Z 軸としている．

2.2 制御上の仮定

起き上がるための条件を考察するに当たり，以下のような仮定をおくことにした．

1. 矢状面内の運動に限定する．
2. それぞれの脚のモータは PD 制御でコントロールし，速度台形則で関節角軌道を算出する．速度台形則では速度関数が連続しているが，少なくとも加速，等速，減速 3 つの場合分けが必要であるので，計算量を減らすために加速時間と減速時間を同じ長さに設定する．駆動時間 T は

$$T = \frac{b - a}{\dot{\theta}_{max}} + \Delta T \tag{1}$$

となる．ただし，初期時刻における関節角度を a，終端時刻における関節角度を b，最大角速度を $\dot{\theta}_{max}$，加速時間 (=減速時間) を ΔT とする．

3. モータの最大角速度は $\dot{\theta}_{max}$ であり，予め分かっている．ここで加速時間を決めると，駆動時間 T は初期時刻の関節角度と終端時刻の関節角度の変化量に依存する．

2.3 起き上がり運動のための条件

上体が起き上がるときの回転中心は腰関節であるが，足先が床面と衝突した後の回転中心は足裏に移動する．この回転中心の移動距離が小さいほど，衝撃時に散逸するエネルギー E_{imp} を抑えることができる．回転中心の移動距離を小さくするには

- 関節角の可動範囲を広くすることで，しゃがんだ姿勢のときの足先位置を腰関節に近づける
- 足先と床面の衝撃を小さくするため，$l_1 \geq l_2$ とする．(ただし l_k は Fig. 1 における Link$k$ の長さ)

とすればよいと考えられる．

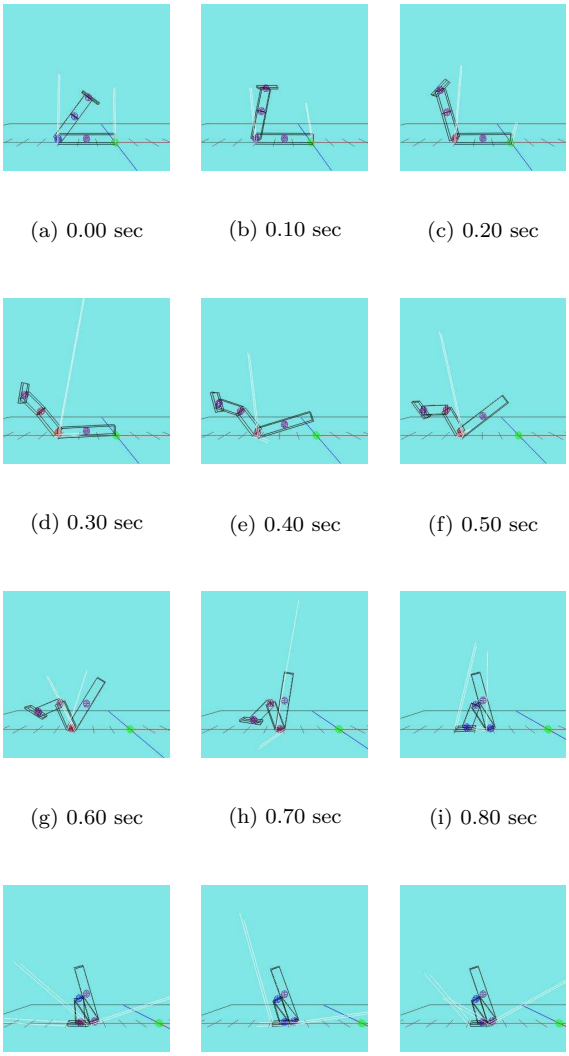
2.4 シミュレーションによる実験

起き上がり運動が実現できるかどうかをシミュレーションで検証した．用いたシミュレーションでは，リンクのダイナミクス計算，床との摩擦，衝突計算，それぞれの関節のサーボの計算を行うことができる．シミュレーションに用いるロボットのモデルのパラメータを **Table 1** に示す．また，各関節可動範囲はそれぞれ次のようにした．シミュレーション上において各関節に設計された軌道を追従させて，起き上がり運動を実現できたようすを **Fig. 2** に示す．Fig. 2 を見ると約 1 秒で脚を振り降ろししゃがんだ

Table 1: Parameters of the simulation robot

link	length [m]	weight [kg]	inertia [kg m^2]
Link0	0.31	0.76	6.09×10^{-3}
Link1	0.15	0.16	3.00×10^{-4}
Link2	0.15	0.38	7.13×10^{-4}
Link3	0.01	0.11	8.33×10^{-8}

joint	range[rad]
hip	2.95〜5.94
knee	0.35〜3.33
ankle	1.57〜4.71



(a) 0.00 sec (b) 0.10 sec (c) 0.20 sec
(d) 0.30 sec (e) 0.40 sec (f) 0.50 sec
(g) 0.60 sec (h) 0.70 sec (i) 0.80 sec
(j) 0.90 sec (k) 1.00 sec (l) 1.10 sec

Figure 2: A stand up motion in the simulation

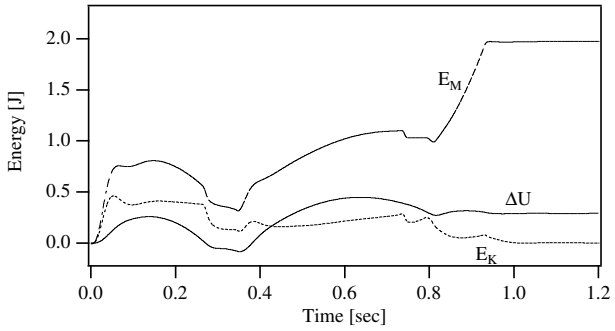


Figure 3: Energy change during stand up motion in the simulation

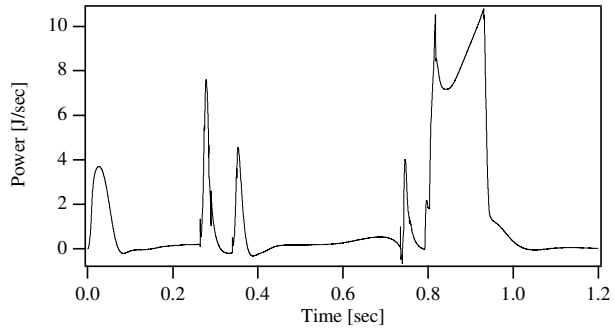


Figure 4: Power change during stand up motion in the simulation

姿勢で脚を深く曲げて，足裏に体重が載っているのが分かる．また，シミュレーションで得られたエネルギーと仕事率の時間変化をそれぞれ **Fig. 3, 4** に示す．ここで，Fig. 3 の ΔU はロボットのリンクの持つ現在のポテンシャルエネルギーから初期状態のポテンシャルエネルギーを引いたもの，E_K はリンクの重心の並進運動エネルギーと重心回りの回転エネルギーの和，E_M は以下のようにして計算されるモータによって与えられたエネルギーである．

$$E_M = \int_0^t \tau\dot{\theta}dt \tag{2}$$

また，Fig. 4 は

$$Power = \frac{d}{dt}\left(E_M - \Delta U - E_K\right) \tag{3}$$

を表している．

Fig. 2 と合わせて見てみると，足先が床面と衝突する瞬間以降 ($t = 0.8$〜[sec]) から運動エネルギーが減少し始めているのが読みとれる．このときポテンシャルエネルギーはさほど大きく変化していないことから，衝撃によってエネルギーが散逸したと考えられる．

また，$0.0 < t < 0.1$ や $t = 0.3$ 付近などで $Power$ が増加しているのはリンクを動かし始める時と止める時に急激に加速，あるいは減速するために衝撃が生じているためである．

2.5 平坦な背中による起き上がり運動の考察

今回のシミュレーションでは関節可動範囲がかなり大きいので，足先と腰関節の距離を小さくすることができ，スムーズに回転中心を移動させることができた．しかし，既に完成されたロボットにおいては，関節可動範囲を簡単に大きくすることができないので，この問題を解決することは非常に難しいと思われる．そこで次に丸い背中をロボットに装着した場合の動的起き上がりについて考察し，丸い背中を設計して実機を使って実験を行った．

3 丸い背中での動的起き上がり

3.1 丸い背中の利点

ロボットに丸い背中を装着することによって，床面との衝撃を抑えて E_{imp} が小さくすることができれば，より少ないエネルギーによって起き上がりを実現できると期待される．

ロボットに丸い背中をつける利点として以下の点があげられる．

- 上体が回転している時に床面との摩擦で失われるエネルギーが腰関節の一点で回転させるときよりも小さくなる．
- しゃがんで脚を曲げた状態のときに足先ができるだけ背中と一続きになるように設計すれば，回転中心の移動が滑らかに行われるので，足先と床面の衝突時の散逸エネルギーの量を減少させることができる．ただし，その場合に脚の各関節の可動範囲を考慮に入れなければならない．
- 非常に簡単な関節角軌道を関節角に与えるだけで起き上がることができる．

3.2 制御上の仮定

2.2 で述べた制御上の仮定に以下の点を加える．

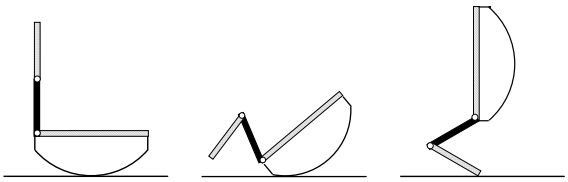
- 運動の過程を以下の 3 つの状態に区分し，それらを経て立ち上がるものとする．

  状態 A 脚を伸ばした状態で上方に持ち上げる．

  状態 B 足裏が床面に接するまで，膝を曲げた状態で素早く脚を振り降ろし始める．このとき関節の角速度は最大になっている．

  状態 C 重心が足裏にうまく載ると，直立姿勢がとれるように膝を伸ばし起立する．

  ロボットに丸い背中を装着したときのそれぞれの状態を **Fig. 5** に示す．状態 A から状態 B へと遷移する間に，ロボットの脚と胴体に運動エネルギを蓄え，これを位置エネルギーに変換することによって，起き上がり運動が実現される．



(a) State A (b) State B (c) State C

Figure 5: Key postures in stand up motion by attaching a round back

本報告では **Fig. 6** のように，円弧状の背中をロボットに取り付けることとした．ここで，設定すべきパラメータ

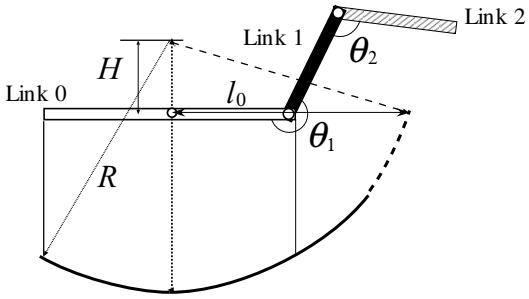


Figure 6: Robot model with a round back

は，ロボットの重心に対する円弧の中心の位置 H と円弧の半径 R である．また，脚を曲げた状態で足先ができるだけ背中と一続きになる，すなわち円周上に足先が来るように設定し，H と R は

$$R^2 = H^2 + l_0^2. \tag{4}$$

という関係を充たすようにした．

3.3 シミュレーションによるパラメータの検討

ロボットに取り付ける背中のパラメータ H，R を，ダイナミックシミュレーションプログラムによって求めた．

シミュレーションに用いるロボットのモデルのパラメータを Table 2 に示す．重量バランスができるだけ HOAP-1 に近付くように設定した．

シミュレーションにおいては $R = 0.2$[m]，$H = 0.0$[m] の背中によって起き上がりを実現できた．シミュレーション上において HOAP–1 の各関節に設計された軌道を追従させて，起き上がり運動を実現できた様子を **Fig. 7** に示す．

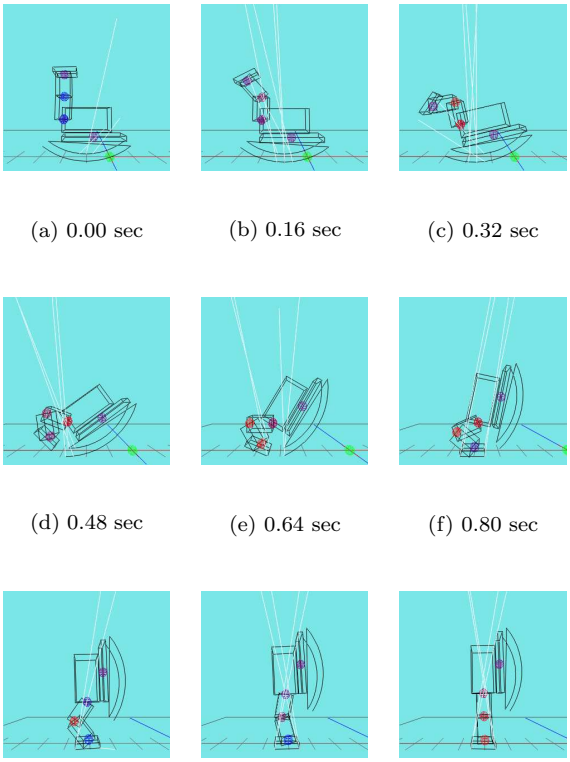
3.4 実ヒューマノイド HOAP での実装と実験

実験においては，富士通で開発された HOAP–1 を用いる．高さがおよそ 0.48[m]，重量はおよそ 6.0[kg] であり，片脚 6 自由度，片腕 4 自由度を備えている．装備されている腕

Table 2: Parameters of the simulation robot

link	*length* [m]	*weight* [kg]	*inertia* [kg m^2]
body	0.20	3.55	1.32×10^{-2}
thigh	0.10	0.7	5.83×10^{-4}
shank	0.10	1.18	9.83×10^{-4}
foot	0.098	0.50	1.67×10^{-5}

joint	range[rad]
hip	1.92〜5.15
knee	0.94〜3.14
ankle	2.10〜4.18



(a) 0.00 sec (b) 0.16 sec (c) 0.32 sec

(d) 0.48 sec (e) 0.64 sec (f) 0.80 sec

(g) 0.96 sec (h) 1.12 sec (i) 1.28 sec

Figure 7: A stand up motion in the simulation

はトルク不足で，腕を用いた静的な起き上がりを実現することができない．HOAP-1 の各関節はそれぞれローカルに位置制御されているので，予め設計された軌道に追従させることとした．

3.5 背中の実装と起き上がり

シミュレーションによって得られたデータを基に，丸い背中を設計し，ロボットに実装した．**Fig. 8** に実装した様子を示す．

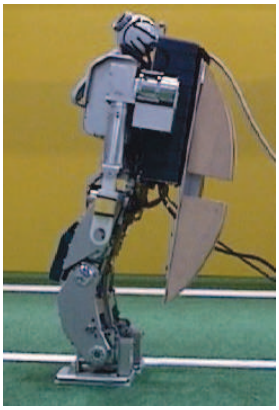


Figure 8: HOAP–1 with a round back

HOAP–1 の各関節に設計された軌道を追従させると，起き上がり運動を実現することができた．この様子を **Fig. 9** に示す．

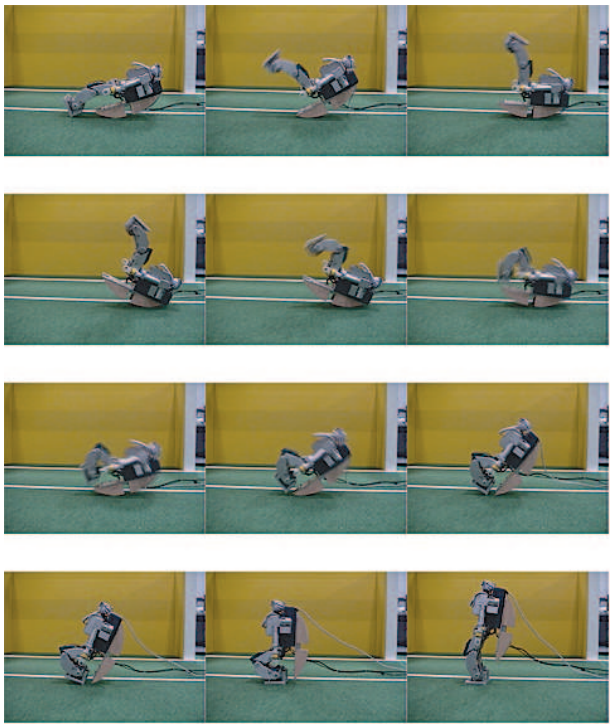


Figure 9: A stand up motion

4 おわりに

本報告では，人間型ロボットの脚の振り降ろしを利用した動的な起き上がり運動について考察し，ロボットに丸い背中を設計，実装することで床面との衝突時におけるエネルギー損失を減少させ，実現させることに実験的に成功した．

今後はより安定して起き上がらせるために，着地時における制御を工夫したり，センサによるロボット自身の状態認知によって脚の振り上げ角の制御をオンラインで行うことにより，さまざまな床面で起き上がり運動が実現できるかどうかを考えていきたい．

参考文献

[1] 金広文男, 稲葉雅幸, 井上博充. 倒れても起き上がることのできる人間型ロボット. 第 13 回日本ロボット学会学術講演会, Vol. 1, No. 1, pp. 195–196, 1995.

[2] 森本淳, 銅谷賢治. 階層型強化学習を用いた 3 リンク 2 関節ロボットによる起立運動の獲得. 日本ロボット学会誌, Vol. 19, No. 5, pp. 574–579, 2001.

[3] 伊藤聡, 阪圭央, 川崎晴久. 脚の振り降ろしを利用した起き上がり運動について. 第 20 回日本ロボット学会学術講演会予稿集, pp. CD–ROM 1I34, 2002.

〒162 東京都新宿区津久戸町 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 査
奥乃 博
京都大学大学院 情報学研究科 知能情報学専攻
〒606-8501 京都市左京区吉田本町
075-753-5376 Fax: 075-753-5977
okuno@i.kyoto-u.ac.jp

幹 事
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
武田 英明
国立情報学研究所 知能システム研究系
樋口 哲也
独立行政法人 産業技術総合研究所
田所 諭
神戸大学 工学部 情報知能工学科

Executive Committee
Chair
Hiroshi G. Okuno
Dept. of Intelligence Science and Technology,
Gradulate School of Informatics
Kyoto University
Yoshida-Honmachi Sakyo, Kyoto 606-8501 JAPAN

Secretary
Minoru Asada
Dept. of Information and Intelligent Engineering
Graduate School of Engineering
Osaka University
Hideaki Takeda
National Institute of Informatics
Tetsuya Higuchi
National Institute of Advanced Industrial Science and Technology
Satoshi Tadokoro
Dept. of Information and Intelligent Engineering
Kobe University

SIG-AI-Challenges home page (WWW):
`http://winnie.kuis.kyoto-u.ac.jp/SIG-Challenge/`