2009－EC－12 (5)
2009／3／7

# IEEE CIG 2008に提出したMs Pacman コントローラーの概要と性能

徳山 超大 †, 松本 大志 ††, ラック ターウォンマット ‡

† 立命館大学情報理工学部

本稿ではIEEE CIG 2008のMs Pacmanコンテストに提出したICE Pumbush2という自動制御プログラムの概要とその性能について述べる．ICE Pumbush2は前版の画像処理及び経路探索を改良したモジュールを搭載し，4000点以上の性能向上に成功している．

# The Outline and Performance of ICE Pambush 2 － Submitted to IEEE CIG 2008 Ms Pac-Man Contest

Chouta TOKUYAMA † Hiroshi MATSUMOTO †† Ruck THAWONMAS ‡

† College of Information Science and Engineering, Ritsumeikan University

In this paper, we describe the outline and performance of ICE Pambush 2, a controller submitted to the Ms Pac-Man contest in IEEE CIG 2008. ICE Pambush 2 is equipped with improved image processing and path finding modules, leading to more than 4000 scores up compared to its predecessor.

## 1 はじめに

計算機械学習人工知能の技術を向上させるためにMs Pacmanの自動制御(コントローラー)で競い合う世界大会[1]がある．初めて大会が開催されたのはCEC 2007であり，我々はその次のWCCI 2008の大会から参加している．本稿では第三回大会となるIEEE CIG 2008に提出したコントローラーの概要と性能について述べる．

Ms Pacmanとは1981年にアメリカでPacmanの続編として作られたものであり，違いは主人公のパックマンにリボンがついているくらいで，ルールはほとんど同じである．Ms Pacmanのゲーム画面を図1に示す．Ms Pacmanに登場するオブジェクトはパックマン，ゴースト，ピル，パワーピル，アイテムである．主なルールはゲームマップ内のすべてのピルとパワーピルを食べるとステージクリア，初期ライフは3でゴーストに当たるとライフが1減る，パワーピルを食べると一定時間(ステージにより異なる)ゴーストがエディブルゴーストになり食べることができる(これにより高得点を狙える)，アイテムを食べると，それぞれ決められた得点(種類によって異なる)を得られる，1万点になると，ライフが1増えるなどである．

ゲームマップは，図2のようにステージ共通で$28 \times 31$マス，各1マスは$8 \times 8$ピクセルで構成されている．ステージ変化により，各オブジェクトの色や経路の形状が変更される場合がある．また移動するオブジェクトは，このマップ内の経路をピクセルレベルで動くことができる．

本コントローラーはループの最初にのみ実行する部分と一連の計算処理動作をループで繰り返す部分で構成されている．前者の部分は画像処理における部分であるので後ほど詳しく述べる．後者の部分は主に動作制御に関連するが，計算量が多くなりすぎると，パックマンの進行方向決定をマスごとに行えなくなってしまうので，どれだけ適切な経路であっても，その経路を辿ることができなくなってしまう．そのため，いかに計算量を抑え，その上でよりよい制御を行っていくかが重要となってくる．

本稿で提案するコントローラーは，前大会であるWCCI 2008に提出したものに大幅な改良をしてい

る．画像処理のモジュールは，大会のサイト[1]で提供されるサンプルコントローラーではなく，新しく独自で開発したものにし，動作制御のモジュールはコストベースに基づいた経路探索法の改良を行うことで，より優れた性能をもつコントローラーを実現した．2章では，ゲーム画面のオブジェクトの特定を行う画像処理を，3章ではパックマンの動作制御を行うルールや探索方法を，4章では結果を，5章では今後の課題を示す．

## 2 画像処理

### 2.1 マップ状況把握

画像処理のモジュールは，見つかったオブジェクトそれぞれの特徴を見つけ出すためにゲームマップを走査し，2次元配列のデータとして格納していく．前述しているように，各1マスは8×8ピクセルであるので，この広さごとに画像認識を行う．画像処理のうえで，見つけ出すべきオブジェクトは大きく分けて，”パックマン”，”ゴースト”，”ピル”，”パワーピル”，”道”，”壁” の6つが存在する．

まず，移動しないオブジェクトおよびマップの目印の画像処理について述べる．移動しないピル，パワーピル，道の3つを探し，その他を壁とすることでマップの形状を認識する．これらのオブジェクトの特徴として，ピル，パワーピル，道の特徴を図3に示す．マップの形状が決定した後，マップの目印となる曲がり角，ワープ地点，ゴーストの檻を探し出す．ここまでの認識は，一度認識するとステージが変更されるまで経路の形状が変更されることはない．また，ピルやパワーピルは，パックマンが通った時に”道”に変更すればよいので，再度走査し，オブジェクトを認識する必要はない．これにより，ゲームマップを認識する回数を削減することができる．

前述しているように，パックマンやゴーストのように移動するオブジェクトはピクセルレベルで移動するため，ピルや道のような認識ができないので，以下のように工夫を実施する．これらを正確に見つけ出すためには，大幅な時間がかかってしまうため，ピクセルレベルではなく1マスレベルでの近似探索を行うこととした．各マス内で，各オブジェクトの代表色をカウントし，その数が特定の数以上になればそのマスにオブジェクトが存在しているとしている．この工夫により，全体的な処理時間を大幅に短縮することに成功した．実際の座標による認識と比べて誤差は生じるが，動作制御を行う際に大きな障害とはならないため許容範囲であると判断した．

処理時間を前大会のコントローラーと比べてみたところ，前回のものでは1ループが約$60ms$であったものが今回は約$30ms$にまで短縮された．この結果は，本章でも説明したとおり，ゲームマップの走査回数を削減したこと，またオブジェクトの座標決定の工夫をした効果であると考えられる．これにより，動作制御にてより多くの時間が使えるため，より高度な制御システムが実装できるようになった．

### 2.2 ワープ地点を考慮した状況設定

ゲームマップの右サイドと左サイドにはワープ地点があり，その地点を通るとその逆サイドに移動できるので，以下はその活用法について述べる．前回のコントローラーでは，この地点の活用ができていなかった．今回はゲームマップ本来の横幅の2倍の配列を用意し，パックマン以外のオブジェクトを図4のように配置することで，ワープ地点の活用に成功した．これにより，経路探索を行うと必然的にワープ地点を利用することができる．データ量は多くなるが，走査回数は1回だけで，二次元配列の要素が多くなるだけなので大幅な時間的制約はかからず，動作制御時にこの計算を行わずにすむ．

## 3 動作制御

この章では動作制御に用いるルールと経路探索とコストについて述べる．まず，動作ルールで目標地点を決定し，その目標地点に向かう経路のうち最もコストの低い経路を選択する．次に，パックマンの移動方向を決定するにはパックマンの現在位置から次 (場合によっては次の次) の交差点または角までの進行方向に基づき，その進行方向のキー入力を行っている．具体的な動作ルール，探索方法およびコストは以下の節で示す．また，以後に示すマス距離とはマス同士の座標のマンハッタン距離を意味する．

### 3.1 動作ルール

目標地点およびその地点までの経路探索方法をマップ状況に応じた動作ルールを用いて決定する．

動作ルールは全部で7つあり，それぞれの動作ルールには優先順位が存在する．この優先順位は我々の戦略を有効に活用するためのものである．主な戦略とはパワーピル直前の交差点または角でゴーストが接近してくるのを待ち伏せ (キー入力を行っていない状態) し，エディブルゴーストを多く食べることによる点数向上を狙ったものである．その優先順位を上位から順番に並べたものを以下に示す．尚，特に明示しない限り，以下のルールで使われている距離とはパックマンを起点としたものとする．

- Rule1: 最も近いパワーピルとのマス距離がステージ1では5(ステージ2では3) 以下，かつ最も近いゴーストとのマス距離が4以上，かつ最も近いパワーピルとそれに最も近いゴーストとのマス距離がステージ1では6(ステージ2では4) 以上のときに待ち伏せ動作 (キー入力を行っていない状態) を行う．
- Rule2: パワーピルが存在していて，最も近いゴーストとのマス距離が8以下，かつ最も近いパワーピルとのマス距離が6以下，かつ最も近いパワーピルとのマス距離が最も近いパワーピルとそれに最も近いゴーストとのマス距離以下のときに最も近いパワーピルに向かう．
- Rule3: パワーピルが存在していて，最も近いゴーストとのマス距離が8以下，かつ最も近いパワーピルとのマス距離が最も近いパワーピルとそれに最も近いゴーストとのマス距離以下のときに最も近いパワーピルに向かう．
- Rule4: エディブルゴーストが存在していて，最も近いゴーストとのマス距離が8以下，かつ最も近いエディブルゴーストとのマス距離が8以下のときに最も近いエディブルゴーストに向かう．
- Rule5: 最も近いゴーストとのマス距離が8以下のときに最も近いピルに向かう．
- Rule6: エディブルゴーストが存在していて，最も近いゴーストとのマス距離が9以上，かつ最も近いエディブルゴーストとのマス距離が8以下のときに最も近いエディブルゴーストに向かう．
- Rule7: 最も近いゴーストとのマス距離が9以上のときに最も近いピルに向かう．

ただし，Rule1での待ち伏せはステージ1，2でのみ，Rule3，4，5で用いる探索方法はA*1動作，Rule2，6，7で用いる探索方法はA*2動作で行う．これらの経路探索方法は次の節で示す．

### 3.2 A*1とA*2アルゴリズム

今回用いた経路探索アルゴリズム [2] はA*1とA*2である．A*1では距離コスト，ゴーストコスト，ノードコストの3つのコストを用いて，設定された目標地点に経路上のコストが最も低い経路を探索するものである．それに対してA*2は距離コストのみを用いており，目標地点へのマス距離が最も近い経路探索を行うものである．使用用途は主にA*1はパックマンと最も近いゴーストがマス距離8以下にいる場合，A*2はパックマンと最も近いゴーストがマス距離9以上にいる場合である．

また，探索の範囲を狭めるために深さの概念を導入する．深さはパックマンの現在のマスから次の交差点または角が見つかるまでの地点を1深さとして，1深さ地点から次の交差点または角までを2深さ地点，これを繰り返しn深さ探索を行う．ただし，進行方向と逆方向の探索 (2深さ以降) は行わないものとする．今回用いた深さはA*1では3深さ，A*2では10深さである．目標地点がその深さ範囲外であれば，そのn深さまでのコストが最も低い経路を選択し，目標地点が範囲内にあれば，A*1ではその目標地点を挟む両方の交差点または角のコストまでを，A*2では目標地点の手前の交差点または角のコストまでを計算に含む．

そのほか，A*2では経路計算を減らすために，その経路のコストが一定以上 ($1000 \times (10 + 2 \times n$ 深さ)) になるとその経路を候補からはずし，次の経路計算に移るようにする．また，交差点や角でスムーズに曲がりきれるようにするために，前もって探索された経路上の2番目の進行方向を押しておく (ゲームシステム上，入力された方向が壁であれば，現在の進行方向を維持するため) ようにしている．まとめるとA*1では目標地点へのゴーストの位置を考慮に入れた最短経路を，A*2では目標地点への最短経路を導出している．

### 3.3 コスト生成法

前述したコストを適用する地点は交差点または角である．今回の動作制御に用いたコストは距離コスト，ゴーストコスト，角コストの3つであり，以下にそれらのコスト計算方法を示す．ただし，ゴーストコストはゴーストの現在のマスからの深さの概念を用いる．

- 距離コスト＝1000 × (対象地点のマスと目標地点のマスとのマス距離)+(パックマンの現在マスから対象地点までの経路上の移動にかかる実際のマス数) − (パックマンの現在マスと目標地点のマスとのマス距離)
- ゴーストコスト1＝500000/(ゴーストの現在マスからゴーストの対象地点のマスまでのマス距離)$^2$

  ゴーストコスト2＝1000000(ゴーストが交差点または角にいるとき)

  ゴーストコスト3＝600000(パックマンとゴーストが壁をはさまずに直線上に並んでいるときにゴースト側の交差点または角に)
- 角コスト＝5000(各角に)

このコスト設定の基準は，まず，距離コストとゴーストコストのバランスを考え，ゴーストにぶつからないぎりぎりの地点まで目標地点に向かえるようにする．このように設定するのは，各ゲームステージをクリアするのにすべてのピルをとらなければならず，ぎりぎりの設定にせずに逃げる動作を優先させると，最後に残った一塊のピルを取れずに角で挟みうちに合うなどしてやられる場合が多くなるためである．また，角コストを各角に5000ずつ与えているのは，角が存在する交差点間距離は基本的に長くなっており挟みうちに合いやすいので，目標地点までのマス距離がほぼ同じ場合は角の少ない経路を選択した方がよいと判断したためである．

コストを用いる範囲は距離コストはA*1とA*2の経路探索範囲の深さまで，ゴーストコストは各ゴーストを中心に2深さの地点までにコストを与える．角コストはマップ読み込み時に角であれば5000のコストを与えておき，変化はしない．

### 3.4 前回のコントローラーとの動作比較

WCCI 2008に提出した動作とIEEE CIG 2008に提出した動作の比較を表1に示す．待ち伏せ動作は以前の動作と比べると，無駄な動作を行っていないので，計算によるずれなどは生じない．また，経路探索方法は2種類の経路探索方法を使い分けることにより，ゴーストに食べられる確率かつ計算量の軽減につながった．さらに，2番目の進行方向入力も行えることになったことで，交差点で曲がり切れずに行きすぎてしまう現象が起こりにくくなった．しかし，コストは以前使用していたピルコストを導入し，ピルの存在する経路を優先的に選択した方がよいかもしれない．また，計算量は増えているがゴーストのコストは厳密に計算されているので，コストの改良によってゴーストから逃げる正しい経路探索が行えるようになった．

## 4 WCCI 2008との点数比較

IEEE CIG 2008に提出したコントローラーとその半年前に開催されたWCCI 2008に提出したコントローラーとの点数比較をデスクトップ(Core2 Duo Memory:3.5GB CPU速度:2.66GHz)，ノートPC(Core2 Duo Memory:2GB CPU速度:1.6GHz)，の2台で行った結果を表2に示す．また，それぞれのPCでの点数分布を図5に示す．表2内の数値は20回点数を計測した時の平均点である．

平均点はデスクトップとノートPCともにIEEE CIG 2008において大幅な平均点アップに成功した．これは，画像処理システムの計算量軽減や進行方向決定動作の効率化によって，新たな動作の追加やコストの精密計算が行えるようになったためであると考える．しかし，2つの異なるPCでの結果からもわかるように，このMs Pacmanの自動制御はPCの性能に大きく依存していることがわかる．

## 5 終わりに

ここまで，画像処理や動作制御の性能を述べ，点数を基に結果を示してきた．しかし，現在のコントローラーにも更なる性能向上のためのいくつかの改良点が残る．その代表的な改良点を以下に3つ挙げる．1つ目は，交差点間でゴーストに挟みうちに合いにくくするために，交差点の安全性を計算することである．2つ目は，パワーピルを取るタイミングをよりよくするために，パワーピルを挟む交差点間に入る条件を与える必要性である．そして最

Table 1　動作比較

| | WCCI 2008 | IEEE CIG 2008 |
|---|---|---|
| 待ち伏せ動作 | 進行方向と逆方向を交互に押す | 壁で停止 |
| 経路探索方法 | A*1 | A*1 と A*2 |
| コスト | ゴースト, ピル, 角 | ゴースト, 距離, 角 |
| 2 番目の進行方向 | × | ○ |

後は, 前章の終わりに述べたような性能による依存を動作制御の計算量の最適化によってなくしていくことである.

## 参考文献


1) http://dces.essex.ac.uk/staff/sml/pacman/PacManContest.html

2) Ian Millington, Artificial Intelligence for Games, The Morgan Kaufmann Series in Interactive 3D Technology, 2006

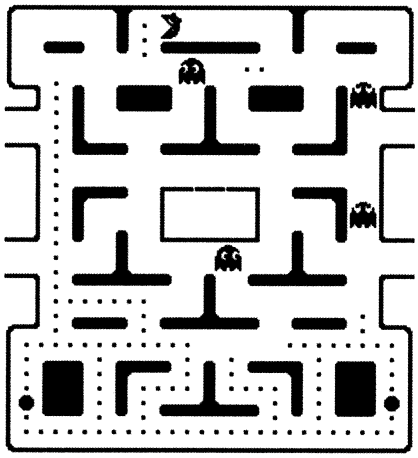



Fig. 1　ゲーム画面

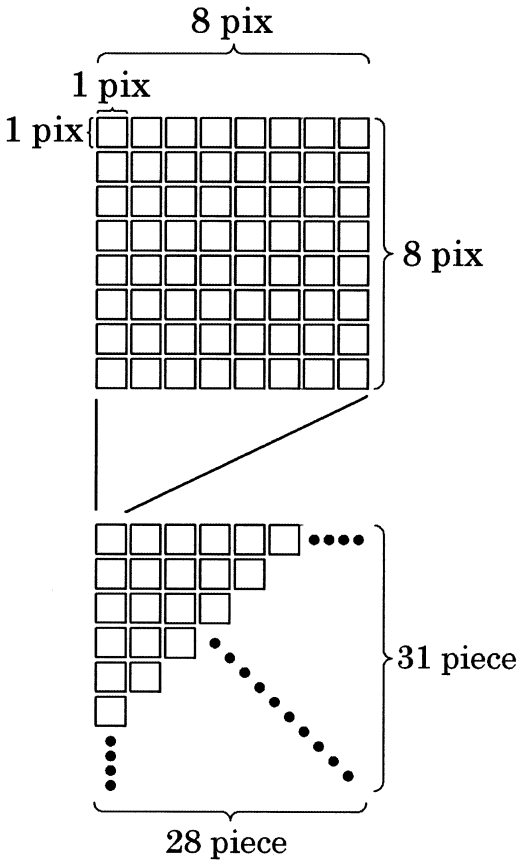
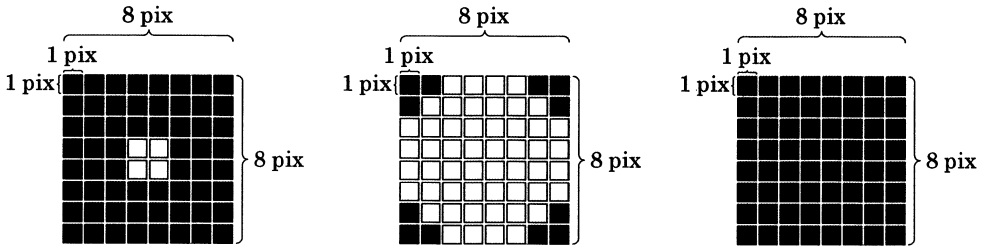



Fig. 2　マスとピクセル関係

Table 2　PC の違いでの点数比較

| | デスクトップ | ノート PC |
|---|---|---|
| IEEE CIG 2008 | 15607 | 11737 |
| WCCI 2008 | 9027 | 7216 |




(a) ピル　(b) パワーピル　(c) 道

Fig. 3　(a) ピル，(b) パワーピル，(c) 道

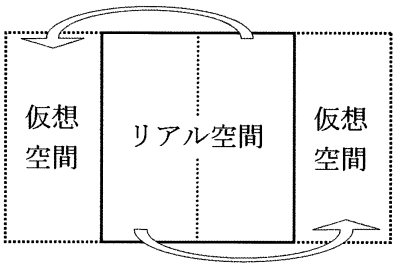
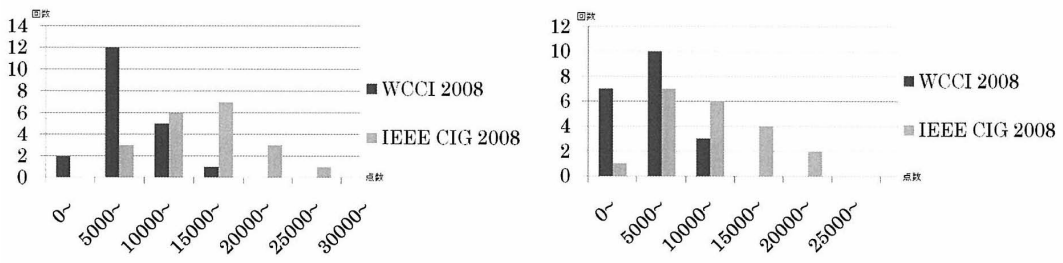



Fig. 4　ワープ地点を考慮にいれたマップ




(a) デスクトップ　(b) ノート PC

Fig. 5　(a) デスクトップ，(b) ノート PC