将棋を対象とした画像情報を用いた自動局面認識手法

栗田哲平[†]　三輪誠[††]　近山隆[†,††]

将棋における画像情報を用いた精度の高い局面の自動認識は，現実の対極での棋譜の効率的取得およびユーザ支援をするにあたって重要な処理である．精度の高い自動認識をするためには，その時の状況に因ってパラメータを変え，ロバストに将棋盤の認識と局面状況の認識を行う必要がある．本研究では将棋を対象とし盤の桝目の認識と，差分情報を用いた局面の自動認識を精度高く行う事を目的としている．駒の動きは事前に得ておいた訓練例を用いて決定木を構成し分類を行い，打った駒は初期盤面の駒情報を訓練例とし逐次最小最適化法を用いた Support Vector Machine による分類によって，その種類の判断を行った．結果として，今回の実験例では 640 × 480 の解像度を持つカメラからの画像情報を用いて，桝目の認識はパラメータを調整しないで 97.0%，桝目属性の分類に 99.7%，打った駒の判断を 100%の正解率で分類を行う事が出来た．

Automatic position recognition method using image information in shogi

Teppei KURITA,[†] Makoto MIWA[††]
and Takashi CHIKAYAMA[†,††]

Automatic position recognition with high accuracy is a critical process for automatic capture of game records in shogi. For high accuracy recognition, recognition of shogi board and game situation is desirable without parameter adjustment. In this paper, we recognize a shogi board and its position situation based on image difference information. The action of pieces is recognized using decision tree with training data. Pushed pieces is classified using Sequential Minimal Optimization with first position informations of board. As experimental results, board position was recognized with the accuracy of about 97.0%, the cells were classified with the accuracy of 99.7%, and placed pieces were classified with the accuracy of 100%.

1. はじめに

近年，コンピュータや撮影装置の高性能化が進み，高画質な動画像を入手して高速に処理をする事が容易になってきた．そのような背景を元に，動画像情報を入力として対象について自動認識を行う技術について様々な分野において研究がなされている．

将棋や囲碁などにおける棋譜の取得は，局面の動いた・打たれた駒や石について第三者が目視する事によって判断し，その時の状況を手動で入力するという手順で行われているのが現状である．また棋譜の入力がされていない限り，対局の検討を行う際には実際の棋譜に従ってその時の状況について手動で再現を行う手間がある．棋譜を記録する第三者が存在しない場合，局面について再現を行うためには，対局者がその盤面状況を記憶していなければならない．従って，初心者同士の対局などでは，検討を行うのが困難になる場合がある．

また，伝統的な将棋盤と駒で将棋を指す事は娯楽の提供という意味では重要で，将棋ソフトウェアなどでマウスを用いて対戦するのは苦痛を感じる利用者が少なくない．

そのような人的な労力問題を解決し，人間にとって使いやすいものを提供する一つの手法として，現実の対局を，動画像情報を入力として自動的にコンピュータの仮想空間上に取り込むシステムが挙げられる．画像情報を利用し，その認識が高精度に行われる事によって，その時の局面状況を理解し，コンピュータによって様々なアクションを返す事が出来る．

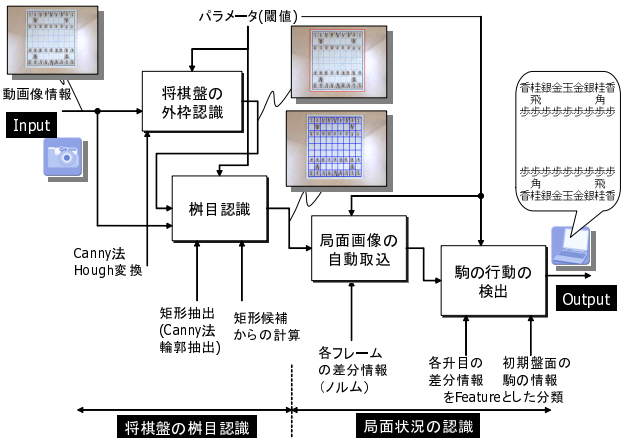

図 1　本手法の概要：Idef0

そこで本研究では画像情報を利用した精度の高い自動認識を目的とし，その認識手法について提案する．

このような自動認識が成されれば，その時の局面状況について実際の局面とコンピュータの仮想空間上の局面のインタラクティブな検討を容易に行うことが出来る．例えば初心者の対局において序盤で定跡を外れた場合，それについて警告を行うという一連の作業をリアルタイムで行うユーザ指導システムも実現可能になる．また，単純にゲーム上の規則を外れた場合

† 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
†† 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, The University of Tokyo

に対しても自動で警告を発する事が出来るようになる．一方，実力者同士の対局でも，その各局面における優劣についてリアルタイムで評価を行う事も可能になる．また，一度自動的に棋譜の入力・保存がされてしまえば，過去の局面に対するアクセスも容易に行うことが出来る．

今回提案する自動認識手法は，入力を画像情報とし，出力をその時点での局面状況とする．その概要を図 1 に示す．画像情報を用い 1) 将棋盤の桝目の認識をし，更にそこから 2) 局面状況の認識を行い，最終的な出力とする．以下にそのステップについて述べる．

1) 将棋盤の桝目の認識　将棋盤の外枠の認識を Canny アルゴリズムや Hough 変換を用い行う．その結果を関心領域とし，将棋盤の桝目の特徴に注目して矩形抽出処理を行う．その処理で得られた矩形を元に，桝目の最外枠を計算し，最終的に桝目の認識を行う．

2) 局面状況の認識　フレーム毎のノルムを元に，局面画像の自動取得を行う．自動取得を行った画像の差分情報を用いて，駒の動きの検出を行い，打った駒に関しては初期盤面からの情報を元にその駒の種類を判定する．

上記の一連の認識を行うことによって，現実の将棋盤の局面を仮想空間上に取り込むことが可能になる．

本報では将棋を対象として，今回用いた認識手法の適用実験を行い，有用性について検証を行った結果について報告する．

2. 関 連 研 究

本章ではゲームの棋譜取得を行っている関連研究で用いている手法について示す．前章でも述べた通り，最終的に局面を取得するために必要なステップは，将棋盤の桝目の認識と局面状況の認識である．この２つの認識が正常になされれば，その時の局面は正しく取り込まれる．

ここでは関連研究として囲碁を対象としたもの[1)2)]，および将棋を対象としたもの[3)] を取り扱う．将棋における認識は囲碁と違い駒の判別・成りの問題がある為に，局面状況の認識に対して，より複雑な処理を要求する．将棋を対象とした画像情報を利用した自動局面認識を行う既存システムとして，山下が開発したソフトウェア「木偶の坊」[4)] がある．このソフトウェアは，コンピュータ上の将棋ソフトウェアを監視し，画像情報から局面状況を認識して，異なった将棋ソフトウェア同士のネットワーク上での対戦を可能にする．

2.1 桝目の認識

ゲーム盤の桝目の認識は，局面状況の認識の精度も左右する重要な事項であり，より正確にその画像にある盤の桝目の位置情報を捕らえる事が出来る手法が必要になる．この桝目の認識という処理は基本的に，一連の認識処理の中で最初の一回行われれば良いので，多少時間がかかっても正確に認識が行われるような手法を採用すべきである．

林山らの手法[1)] では，囲碁盤の入力画像に対して盤面であると予測される画素値を RGB の範囲として持たせておき，その範囲内の色を抽出し，設定された範囲外の画素値は (R,G,B)=(0,0,0) に設定する．この範囲を関心領域 (Region of Interest，以下 ROI) としながら画像に対して Sobel フィルタを用いて，エッジの検出を行う．エッジの検出を行った画像を２値変換し，Hough 変換を用いて直線を検出し，その直線の

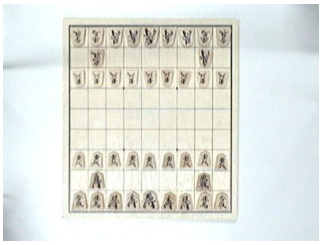

図 2　入力として用いる画像例

情報から囲碁盤の桝目を認識している．

2.2 局面状況の認識

局面状況の認識は，桝目の認識と異なりリアルタイムで処理をする事になるので，技術の実用化の為には処理時間の短縮を成さなければならない．

垣内ら[3)] は n 手目と n+1 手目の桝目における分散値の違いにより，移動駒が重ならない場合・重なる場合の分類をしている．打ち駒の判定には，低周波成分のパワースペクトルをニューラルネットワークにかけて文字分類を行っており，入力層 36，中間層 6，出力層 3 で実験を行った結果，平均 80%の認識率を出している．

駒の判定に対して免疫的アルゴリズム（Immune Algorithm，以下 IA) を用いて囲碁盤の認識を行う実験もなされている[5)]．IA は生物の免疫システムを工学的に模倣した最適化アルゴリズムの一種である．この手法では，初期の個体群をランダムに生成して，各個体でテンプレート画像を ROI にマッピングし，それを ROI の元画像と比較して画素値の差の絶対値を用いて適応度を求めている．全個体の中からランダムに一部の個体群を選択し，その個体群の中で適応度の高い個体を抽出してより高い免疫度を与える．これは他の個体よりも優秀な個体の可能性が高いためである．以上の処理を一世代の中で一定回数繰り返す事で集団内の優秀な個体は高い免疫度を持っていくので，その優秀な個体のみを残して残りを淘汰する．淘汰した個体の代わりに同数の新個体を突然変異・局所探索を用いて生成し，位置座標が近い個体は適応度を比較して，ここでも優秀な個体のみを残していく．一連の処理を１世代として，予め設定しておいた世代数の進化をさせた時点で終了とさせる．

この実験についての評価実験では，10 回行った所，200 世代進化後に 8 個の解のうち平均 7.8 個の解を，500 世代進化後では解を全て発見することに成功している．探索時間は，盤面の８種類の将棋の駒を全て判定するのに 200 世代で 61 秒かかっている．

3. 認 識 手 順

本章では，実際の認識手順について解説する．全体の概要は図 1 に示している．

本手法では，まず将棋盤外枠の認識を行い，処理範囲を絞って桝目の認識を行う事で，最終的に桝目の最外枠の判断を行う．従来手法では，将棋盤の形状の特徴に注目し．画像情報から直線を判断して，その形状から将棋盤の外枠そして桝目の認識まで行っている．しかし，直線抽出手法のみで，確実に将棋盤の外枠の範囲まで取得する事は，ノイズの影響もあり困難であると言える．実際その直線抽出手法のみを用い，直線を判断し，将棋盤の外枠を正確に得られた例は，１００回程実験を試

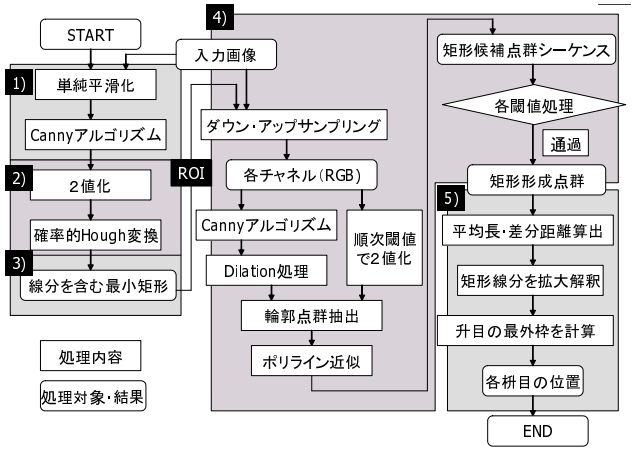

図 3 盤面認識の手順

した所１割にも満たなかった．更にそこからの桝目の判断についても，桝目の最外枠の位置を正確に得られなければならず，将棋盤の外枠と桝目の最外枠の区別が困難である事から，桝目の認識も難しい．また，画素値 (RGB や HSV) を用いて盤面を判断する手法も，盤面の色やその時の明暗値に因っているので，汎用性のあるものとは言えず，パラメータの設定がその時と場合によって大きく異なってしまう．

今回の認識手法では，将棋盤の外枠の認識結果を用いて桝目の認識を行う．これは，その将棋盤の外枠と，桝目の最外枠の要素を正確にどちらか判断するのではなく，そのどちらかの要素を上下左右に持つ ROI を利用し，桝目の認識を行っていく．

3.1 将棋盤の桝目認識手順

まず，カメラを用い，ビデオストリームを読み込む構造体を確保し，カメラからフレームを取り出す．そこで取り出されたフレームを入力画像とする．今回，カメラから得られる入力画像は，図 2 のように将棋盤の真上から得られたものとしている．取り出されたフレームの入力画像に対し，盤面認識処理すなわち将棋盤の外枠，および桝目の認識を以下の手順で行う．

1) 入力画像に対して，エッジの検出を行う

入力画像に対して，前処理としてシンプルスムージング（単純平滑化）を行う．ここでスムージングを用いたのは，後にエッジを多く検出したいので，ノイズの除去をやりすぎないためである．

エッジの検出には Canny アルゴリズム[6] を用いる．Canny（拡張ゼロ交差法）は，Sobel・Laplacian（ゼロ交差法）などのエッジ検出法を拡張した手法であり，局所的な変化に対して頑強であり，エッジの連続性が高い事が知られている．(弱いエッジが強いエッジに接続されている場合に限り，弱いエッジを出力に含める．そのため他の方法に比べてノイズの影響を受けにくく，より正確に弱いエッジを検出する) この事によりノイズなどエッジと間違えられるものを少なくし，真の弱いエッジを検出する事が可能になる．エッジ接続・初期検出のためのパラメータを２つ用意する．エッジの接続では，注目ピクセルの８近傍（4 方向）に対する勾配が求められ，最も強い勾配を持つ方向に（閾値を下回らない限り）エッジが接続される．出力画像はシングルチャネルのグレースケール画像となる．

2) エッジ出力画像に対して直線抽出処理を行う

前処理として，エッジ出力画像に対し閾値処理（２値化処理）を行う．画素の閾値は 128 とし，処理後は 0 と 255 に分類される．直線抽出には確率的 Hough 変換を用いる．Hough 変換[7] はノイズに強い線抽出法であり安定的に複数の直線を検出でき，画像処理の不完全性によって直線が不完全でも有効に働く場合が多いので採用した．確率的 Hough 変換は Hough 変換と異なり，全ての線を返す事は行わずに条件に適合した線分のみを返す．条件の閾値パラメータとして，最小の線の長さと２つの線分間を同一線分として扱う最大許容間隔を設定する．この処理を行う事によって，直線で長い形状の線という特徴を多く持っている将棋盤の線をより検出しやすくする．

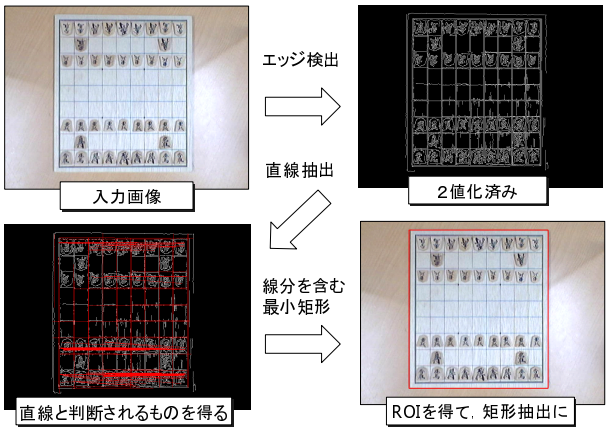

図 4 将棋盤の外枠認識の過程

3) 直線抽出画像より盤面の ROI を設定する

2) で得られた線分を含有する最小矩形を求める．その矩形を ROI に設定する．この ROI は，正確に将棋盤の領域を表している訳ではないが，盤面を含んでいることになる．

4) 得られた入力画像の ROI で矩形抽出処理を行う

ここでは，桝目の候補の検出を行う．矩形（またはそれに近い形）だと判断される輪郭を抽出する処理の手順を以下に示す．

まず入力画像の ROI に対して，ダウンサンプリング，アップサンプリングを順次行い，ノイズの除去を行う．

ここからの処理は入力画像の色空間の画素において，同一チャネルのみの情報を用い処理を行う．すなわち，入力画像の同一チャネルを，シングルチャネルの画像に変換して処理をしていく．得られたグレースケール画像に対し 1) と同じように Canny 法を用い，エッジを検出．エッジ検出画像に対し，dilation 処理を，エッジを強調する為に成し，輪郭の抽出を水平・垂直・斜めの成分を圧縮しながら行う，つまり輪郭端点を検出していく．その抽出輪郭端点を対象として，Douglas-Peucker アルゴリズム[8] を用いポリライン近似を行い，その結果を保存する．その処理の結果，点数が４つであり，面積が閾値以上・閾値以下，更に輪郭が凸であるならばその点群シーケンスは矩形候補となる．矩形候補の角度を内積により計算し，全ての角度が９０度付近（許容する閾値：矩形度閾値を設定する）であるならば，矩形であると判断し結果を保存していく．以上の輪郭抽出から矩形と判断するまでの処理を，エッジ検出の変わりに２値化処理を，閾値を順次変えながら（増加させながら）行う．

矩形と判断された，点群シーケンスはその面積 s と周囲長 l から (1) 式に示す伸長度 w を計算し，伸長度が閾値以上であ

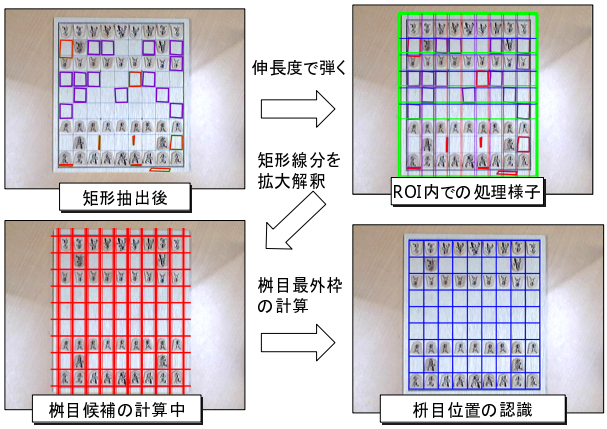

図 5　桝目認識の過程

ると判断された場合には矩形候補から除外する．

$$\boldsymbol{w} = \frac{l^2}{s} \tag{1}$$

5) 得られた矩形から桝目の最外枠の位置を計算

ROI 内にある矩形の平均サイズとその座標（位置）を計算する．矩形の左右上下位置を判断し，(x, y) 座標において，それぞれ隣接する矩形からその差分を計算する．これは，抽出された矩形は桝目の内側を示しているのでこの点群シーケンスをそのまま桝目の候補として使用する事は出来ないからである．よってその誤差分を計算している．矩形の上下左右位置から，その差分を計算に入れ，ROI 内で矩形の縦横それぞれの平均サイズ分ずつ線分を拡大していく．その線数が１０個ならば，それを桝の区切り位置の候補であると判断．つまり最も外側にある直線を桝目の最外枠候補とする．この処理を全ての桝目候補において行い候補を出力し，その候補群から最終的な桝目の最外枠を決定する（現在は単純に極端な値を省いた候補の中での平均を用いている）その桝目の最外枠から，将棋盤の桝目は９×９であるという知識を用い，桝目の位置を計算する．

3.2　局面状況認識手順

n 手目における局面と n+1 手目における局面の画像情報を用いて駒の移動，または打ちの検出を行い，局面状況の認識を行っていく．リアルタイムに局面状況について処理をするための，画像検出手順を以下の節に示す．

3.2.1　局面画像の入手

まず，初期盤面の画像をカメラからストリームを経てフレームを取り出す事で得る．ある一定感覚毎にカメラからフレームを取り出し，画像をキャプチャする，これを次フレーム画像とする．次フレーム画像と，前のフレーム画像とのノルムを計算する．ノルムは各ピクセルの残差平方和の総和の平方根として得る．イメージサイズ $(I \times J)$ とチャネル数に対するノルムを計算し標準化したノルム N を導く．式 (2) にその標準化ノルムの式を示す．

$$N = \frac{\sum_{k=1}^{3} \sqrt{\sum_{i=0}^{I} \sum_{j=0}^{J} (v(x_{1ijk}, y_{1ijk}) - v(x_{2ijk}, y_{2ijk}))^2}}{3 \times I \times J} \tag{2}$$

ここで，前フレーム画像の k チャネルにおける各画素値を $v(x_{1ijk}, y_{1ijk})$，次フレーム画像の k チャネルにおける各画素

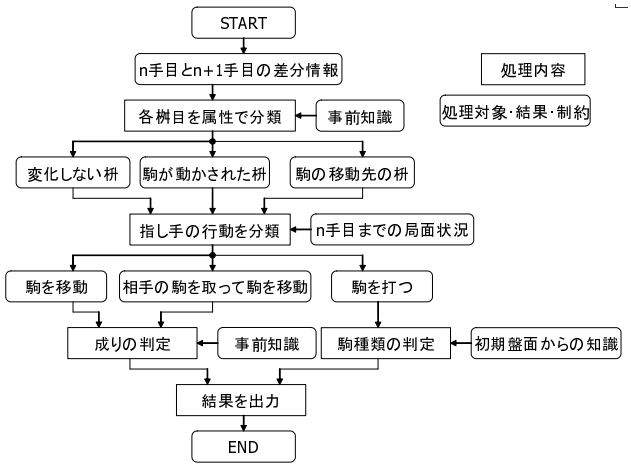

図 6　局面状況認識の手順

値を $v(x_{2ijk}, y_{2ijk})$ としている．また，$k = 1$ で色空間の第一成分，$k = 2$ で第二成分，$k = 3$ で第三成分を示す．以上のノルムが閾値以下なら，局面の状況は変化していないものと判断する．その場合は，ノルムを計算した後に，次フレーム画像を前フレーム画像として保存し，前フレーム画像は破棄する．また，人が駒を打つときには，人の手がキャプチャされた画面に侵入する事になる．よって，その時の次フレームと全フレームのノルムは，局面の状況に変化が無い時に比べ，極端に大きくなる．このノルムの乱れを検出した時，局面状況の変化があったと考える．変化があった場合，次フレームの画像を前フレーム画像へ置き換える操作は行わず，そのまま破棄する．このようにして人間の手が画面に侵入している間は，次フレーム画像は破棄され続ける．人間の手の進入が終わり，次フレームと前フレーム間のノルムがある閾値以下になったら，次フレームと前フレームを n+1 手目・n 手目と見なして局面状況の認識に入る．局面状況の認識を行うのは，無駄な計算を省くため局面状況に変化があった場合のみにしている．ここでもし人の手が一回画像に進入して，何も局面状況の変化が無かった場合は，後の局面状況の認識処理によって何も変化していないと返す．これは後に述べる移動前桝・移動先の桝のどちらもが検出されなかった事を表している．

3.2.2　駒の動きの検出

次に，局面の状況，つまり駒がどのような動きをしたか，を認識する方法について述べる．処理の概要を図 6 に示す．

１手の間における将棋の駒の動きは，1)〜3) に大きく分けられる．1) 何も無い桝に駒が移動する 2) 相手の駒がいる桝に駒が移動する 3) 何も無い桝に駒を打つ 1) は将棋で一番多く見られるパターンであり，単純な駒の移動である．2) は相手の駒を取る事を示す．3) は終盤の局面に多くなってくるが，自分の保持している駒を新たに打つ事を指している．

局面状況の認識は前述の通り n 手目と n+1 手目の画像の差分情報を用いて行う．差分情報で検出される桝は，「駒の移動前の桝」と「駒の移動先の桝」である．この検出に用いる特徴量などは後に述べる．1) 2) の行動ではこの「駒の移動前の桝」と「駒の移動先の桝」はどちらも検出される事になる．3) では，「駒の移動先の桝」だけ検出されることになる．つまり，「駒の移動前の桝」と「駒の移動先の桝」がどちらもが検出されれば，1) 2) のどちらか，「駒の移動先の桝」だけが検出され

れば 3) の行動を駒はしていると判断することができる．1) と 2) の区別であるが，これは n 手目と n+1 手目における認識の時，知識として得られている n 手目までの局面の状況より，どちらに属しているかはあるかは判定出来る．相手の駒を取って駒が動かされる場合と，普通に駒が移動する場合とでは，移動先の桝の特徴量は違う傾向を持つと考えられるが，今回の実験ではこれらは同じラベルとして扱っている．

「駒の移動前の桝」と「駒の移動先の桝」を検出するための特徴量として,「桝の画素値平均差分（各チャネル）」,「桝の画素値標準偏差差分（各チャネル）」,「桝の各チャネル画素値平均差分の絶対値の平均」,「桝の各チャネル画素値標準偏差差分の絶対値の平均」,「枡のノルム」を算出する．

桝全体の画素差は，HSV 色空間の要素で扱う．HSV は色相 (Hue)，彩度 (SaturationChroma)，明度 (BrightnessLightnessValue) の三つの成分からなる色空間である．HSV の特徴として，人間の知覚手法と同じ傾向を持っている事があげられる．RGB 色空間において，(R,G,B) が最小値 0.0，最大値 1.0 の範囲で設定されている時，HSV 色空間でその値に相当する (H,S,V) の値は以下の式で定義される．MAX を (R,G,B) の最大値，MIN を最小値とする．

$$V = MAX \tag{3}$$

$$S = \frac{MAX - MIN}{MAX} \tag{4}$$

$$H = \begin{cases} 60 \times \frac{G-B}{MAX-MIN} & (MAX = R) \\ 60 \times \frac{B-R}{MAX-MIN} + 120 & (MAX = G) \\ 60 \times \frac{R-G}{MAX-MIN} + 240 & (MAX = B) \end{cases} \tag{5}$$

この計算では，H が 1〜360，S と V が 0〜1 の範囲で出るが，内部の処理では各チャネルの値を，H を 0〜180(H/2)，S と V の値を 0〜255(255 × S(V)) として扱う．RGB 色空間の要素も用いて実験を行ったが，ここでの判定には HSV 色空間での特徴量を用いた方が精度の高い結果が出た．

桝目における k チャネル画素値平均差分 $S_m(k)$ は，前フレーム画像・次フレーム画像の k チャネル画素値の和をそれぞれ $m_1(k)$，$m_2(k)$ としたとき以下で表される．ここでは桝目のサイズを $(I' \times J')$ としている．

$$S_m(k) = \frac{m_2(k) - m_1(k)}{I' \times J'} \tag{6}$$

$$m_1(k) = \sum_{i=0}^{I'} \sum_{j=0}^{J'} v(x_{1ijk}, y_{1ijk}) \tag{7}$$

$$m_2(k) = \sum_{i=0}^{I'} \sum_{j=0}^{J'} v(x_{2ijk}, y_{2ijk}) \tag{8}$$

また，k チャネル画素値標準偏差差分 $S_d(k)$ は前フレーム・次フレーム画像の k チャネル画素値標準偏差 $(\times\sqrt{I' \times J'})$ をそれぞれ $d_1(k)$，$d_2(k)$ とすると以下で表される．

$$S_d(k) = \frac{d_2(k) - d_1(k)}{\sqrt{I' \times J'}} \tag{9}$$

$$d_1(k) = \sqrt{\sum_{i=0}^{I'} \sum_{j=0}^{J'} (v(x_{1ijk}, y_{1ijk}) - m_1(k))^2} \tag{10}$$

$$d_2(k) = \sqrt{\sum_{i=0}^{I'} \sum_{j=0}^{J'} (v(x_{2ijk}, y_{2ijk}) - m_2(k))^2} \tag{11}$$

また，桝の各チャネル画素値平均差分の絶対値の平均 T_m と画素値標準偏差差分の絶対値の平均 T_d は式 (12)，式 (13) で表される．

$$T_m = \frac{\sum_{k=1}^{3} |S_m(k)|}{3} \tag{12}$$

$$T_d = \frac{\sum_{k=1}^{3} |S_d(k)|}{3} \tag{13}$$

ノルムについては既に式 (2) に示している，ここでは $I \to I'$，$J \to J'$ とする．

以上の特徴量を用いて,「駒の移動前の桝」と「駒の移動先の桝」の検出をしていく．つまり次フレーム画像と前フレーム画像の差分情報より，各桝目の「駒の移動前の桝」と「駒の移動先の桝」と「変化の無い桝」の分類を行っていく．

直感的に考えると,「変化の無い桝」よりも「駒の移動前の桝」と「駒の移動先の桝」の方が,「ノルム」の値は大きくなるであろうし,「画素平均の差」や「画素偏差の差」も大きな違いが出てくると推測される.「変化の無い桝」は全ての特徴量で大きい変化をしない．また,「駒の移動前の桝」と「駒の移動先の桝」の間でも,「駒の移動前の桝」は，今まで駒が置かれていて桝目の画素偏差は大きかった箇所が，駒が移動して桝目だけになり単純な画素値の値になって画素偏差の値が小さくなると推測される.「駒の移動先の桝」は逆になり，画素偏差の差に大きな違いが出ると考えられる．

ここで，各局面における各桝目に「駒の移動前の桝」,「駒の移動先の桝」,「変化の無い枡」のラベルを関連付け，上で述べた特徴量を用いて，分類をするための学習を行った．今回は，事前実験として訓練データを得ておき，新たな局における駒の動きの検出に用いることにする．学習手法については次章で述べる．

以上の分類を行う事で，現在の局面における駒の挙動が (1),(2),(3) のうちどれに属しているかを判断する事が出来る．(1) だった場合には駒を移動させその場所を記憶する，(2) では取った駒と移動した駒を記憶する．ここで (1)，(2) の場合では,「駒が成る」という問題があるので，その判定を行う必要がある．駒が成る為の条件を満たしているかは，n 手目までの局面の状況を元に判定することが出来る．実際に駒が成ったかどうかの判定は，ルールの縛りだけでは決定する事が出来ない．そこでここにも，先ほど用いた特徴量を用いて駒の判定を行う．各枡目の属性分類と同じように，事前実験として得た訓練データから成り駒とそうでない駒の分類をする学習モデルの構築を行い，それに従って成りの判定をしていく．

また，(3) の場合には，打った駒がどの駒であるかを判定する必要がある．その場合の判定については実験結果の章で解説する．

3.2.3 打った駒の判定

打った駒の判定には，初期盤面から訓練画像を取得しておき，その画像で学習を行い，実際の打った駒の分類を行っていく．訓練画像は初期盤面の自分の駒と相手の駒から王（絶対に打つ事は無い駒）を除外した計３８画像である（歩:18，香車：4，桂馬：4，銀将：4，金将：4，角行：2，飛車：2) 訓練画像は，初期取得桝目のずれに対応するように，桝目の境目の線を含まない為に，上下左右を多少カットしている．今回その長さは縦横それぞれ桝目の縦横サイズの 1/8 と設定している．しかし，実際の実験ではこの訓練画像のみでは，訓練例が少なくあまりいい精度の分類が出来なかった．そこで，駒の位置ずれ

にも柔軟に対応できるように，桝目の取得範囲を，同じ大きさで中心を (x, y) 軸で± 1 ずつ（2 まで）ずらしていき，元画像を含め 25 の画像を取得した．のこれを訓練データに含めて，つまり $38 \times 25 = 950$ のデータを用いて学習を行った．

駒の特徴量として，以下の値を用いた．

「画素値平均 (RGB)」,「画素値標準偏差 (RGB)」

「画像の上半分の画素平均」,「画像の下半分の画素平均」

「離散フーリエ変換により求めた低周波成分」

「PPED アルゴリズムにより算出したエッジ特徴ベクトル」

上の特徴量を用いて，テストデータの分類を行った場合に，最もよい精度が出た．ただエッジ特徴量については後述する特徴選択を行っている．

「画素平均 (RGB)」は前述した式 (7),(8) の画素値の和を画像の画素数（サイズ）で割り，画素値の平均を求めたものである．「画素値標準偏差 (RGB)」も式 (10),(11) を画像の画素数の平方根で割ったものである．今回の分類では HSV 色空間ではなく RGB 色空間を用いている．これは実験結果で RGB 色空間を用いたほうが良い精度が出た事もあるが，駒の分類では事前知識を持つことなく，その局毎に訓練データを初期盤面の駒がある桝目の画像から得ているので，その時の環境の違いを考慮する必要が無くなっている事が理由として挙げられる．

「離散フーリエ変換により求めた低周波成分」は，２次元の離散フーリエ変換 (DFT) を行い，原点（直流成分）周辺の低周波成分である 7×4 成分から同一の情報である 3 成分を省いた 25 成分を用いた．(x, y) 座標での２次元フーリエ変換の定義を式 (14) に示す．ここでは $u = 0...M-1, v = 0...N-1$ である．

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i (\frac{ux}{M} + \frac{vy}{N})} \qquad (14)$$

「PPED アルゴリズムにより算出したエッジ特徴ベクトル」について以下で説明する．PPED（Projected Principal-Edge Distribution）アルゴリズム[9] とは，64×64 のピクセル画像を 64 次元ベクトルで表現するアルゴリズムのことであり，データ量を 1/64 にしながら，人が目視する感覚と近い位置にベクトルをマッピングする事が可能である．このアルゴリズムによって求められる特徴ベクトルは，ものの大体の形を表すとされている．認識を 64×64 ピクセルの ROI とし，認識対象を認識カーネル内に取り込む．入力画像に対して 5×5 のエッジ検出フィルタ演算を行い，水平方向，垂直方向，± 45 °方向のエッジを検出する．エッジが存在する場所のフラグを 1 として得られるものを Feature Map と呼ぶ．次に Feature Map の二次元情報を一次元情報へと圧縮する．水平方向の Feature Map は水平へ，垂直方向の Feature Map は垂直へ，± 45 °方向はその方向へエッジのグラフを加算していく．平均化操作をし，各 Feature Map から 16 個のエレメントを作成して，つなぎ合わせて 64 次元ベクトルとしている．この特徴ベクトルは，ものの形の特徴量としては有効であるが，回転や位置ずれに弱い．そこで今回は，隣接する４つのエレメントを加算して平均化し，次元数を 1/4 に更に圧縮している．

これらの特徴量を先ほど述べた初期盤面の訓練データに対してそれぞれ算出し，その訓練データを用い打った駒を判定するための学習モデルを構築する．学習手法については次章で示す．

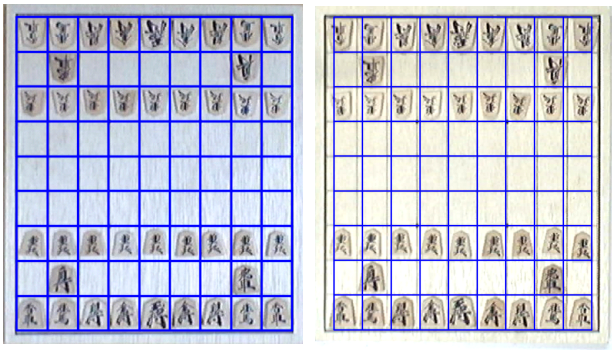

図 7 桝目認識の結果例

4. 実験結果

本章では前章までに説明した認識手法の適用実験を項目ごとに示す．実験に用いた計算機の主要スペックは，CPU：Intel CoreTM Solo U1400 1.20GHz，メモリ：1GB である．実装は C++言語を用い，一部の処理については Intel が公開をしている Open Source Computer Vision Library[10] を利用している．また今回の実験全体では，カメラの解像度は (縦, 横) が (640,480) の低解像度のものを用いた．これは，この程度の解像度の状況でも認識出来るということを示すためでもある．

4.1 将棋盤の桝目の認識

最終的な将棋盤の桝目の検出結果の例を図 7 に示す．処理時間は平均で約 0.92 秒であった．図を見るとわかるようにしっかりと桝目が取れている例もあれば，縦横それぞれに誤差が出てしまっている場合もある．今回の認識実験でその誤差をプロットしたものが図 8 であるこれは 130 回程の実験サンプルに対する，手法適用後の認識した桝目の最外枠における各方向へのずれを示している．ここで誤差は各状況で桝目や将棋盤の大きさや形状が変わるので，その値を標準化するために左右方向なら枡目サイズの横幅，上下方向なら縦幅で割ったものを用いている．すなわち，１なら桝目の大きさ１つ分だけずれているという事になる．各サンプルに対する実験状況は，碁盤の場所・明るさ・碁盤の種類を変化させながら桝目の取得を行ったものである．図 8 を見ると，左右上方向はずれはほぼ誤差内に収まっているのに対して，下方向へのずれが４回程１枡分ずれている事が分かる．これは，初期の ROI 検出で，範囲が狭かったために，桝目の最外枠の位置の判断を誤った為である．この４つの実験サンプルはいずれも，初期のエッジの閾値を低く調整すれば桝目が正常に認識出来た．だが，閾値のパラメータの調整をなさないで完全に桝目の検出が可能になるのが理想であり，それは今後の課題である．

今回の実験では，確率的 Hough 変換における，第一閾値を画像の縦横サイズの平均の 1/8，第二閾値を 1/100 とし，Canny の第一閾値を 80，第二閾値を第一閾値の 3 倍，面積の閾値上限を ROI サイズの 1/50，下限を 5pixel としている．また，矩形許容閾値の矩形度については，伸長度の閾値を設ければどの値に設定しても問題なく処理は働く事が今回分かっている．伸長度の閾値は今回 16.2 と設定した．

4.2 局面状況の認識

局面状況の認識実験について，以下に示す．

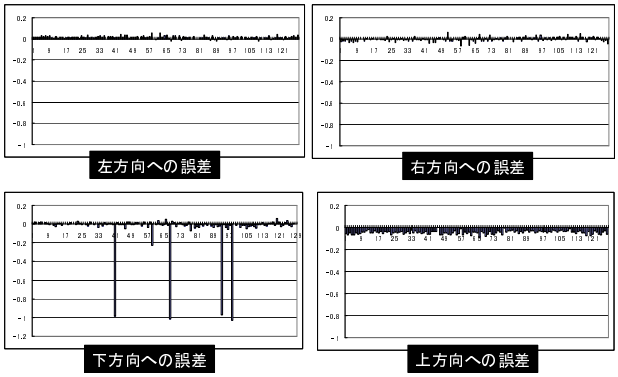

図 8 桝目の最外枠における各方向への誤差（横軸：サンプル数）

表 1 局面画像自動入手，実験結果

	実験局面	正常認識	前と同じ	手が進入	認識されず
数値	1025	997	16	1	11
レート	1.000	0.973	0.016	0.001	0.011

表 2 桝目属性の分類結果

Label	TP Rate	FP Rate	Presition	F 値
変化が無い枡	0.997	0.000	1.000	0.999
駒の移動先の枡	0.989	0.003	0.805	0.888
駒の移動前の桝	0.979	0.000	0.979	0.983

表 3 桝目属性の分類数

認識結果	変化のない枡	駒の移動先の枡	駒の移動前の枡
変化が無い枡	93819	264	0
駒の移動先の枡	0	1175	13
駒の移動前の桝	0	20	937

4.2.1 局面画像の入手

カメラを用いた局面画像の入手を行った実験について，表 1 に示す．ここで各フレームは，0.3 秒毎に取得していった．実験は 10 局，計 1025 局面のものとする．今回の実験での自動認識成功確率は 97 %であった．ここで誤って認識されてしまった事例を表 1 で 3 つに分離した.「前の局面と同じ」というのは，連続して同じ状況の局面が認識された事例である．これは，対局者が一回駒を持ち，また置きなおして考え直したりする場合に起きる．しかし，このように同じ状況の局面が認識されても，局面状況の認識処理により,「駒の移動前の枡目」と「駒の移動先の桝目」がどちらも検出されなければ，動いていないと判断されるので問題は無い.「盤中に手が進入」については，今回の実験では 1 例だけであったが特殊な例として事例を分離した．これは得られた画像の将棋盤の中に手が若干進入している事例であり，この取得画像を用いると局面状況認識の時に差分情報が誤った結果を出す場合がある.「認識されず」は，打った事が検出されず，その局面の画像を取り込むことが出来なかった事例である．これは，手をあまり将棋盤に進入させず端の方だけ動かした場合に，ノルムが閾値を超えず発生する事が多い．以上の 2 つの誤認識の事例は，そのままの状態だと局面状況の認識で誤った結果が出てしまうので，問題である．

4.2.2 駒の動きの検出

各局面における各桝目に「駒の移動前の桝」,「駒の移動先の桝」,「変化の無い枡」のラベルを関連付け，上で述べた特徴量を用いて，分類をするための学習を行う．ここで用いた手法は，枝刈りの決定木 (J48) である．ここでは特徴量が多くないので，単純な構造で高速に分類を行う為に決定木を採用した．また今回実験した所，NaiveBayes や SVM，多層パーセプトロンを利用した他の分類手法より決定木を用いたほうが精度の高い（または同じ程度の）結果が出た．実験では訓練データおよびテストデータそれぞれ 1 局につき 2，3 回，対極の途中に配置してある将棋の駒を全部崩して，また同じ位置に配置を行った．これは，駒がずれても正常に動作するようなロバストなシステムになっている事を確かめる為である．今回，10 局分桝目データを訓練データとしてモデルを構築し，同じく 10 局分・1188 局面の 96228 の桝目テストデータを分類していった所，99.69%の桝目が正しく分類された．その分類結果と分類数を表 2，3 に示す．

ラベル x で正常に x に分類されたテストデータの総数を a とする．この時，a を x ラベルの True Positive と呼ぶ．また，ラベル x のテストデータの総数を b とすると，x ラベルの True Positive Rate(TP Rate) は式 (15) で表される．

$$TruePositiveRate = \frac{a}{b} \tag{15}$$

ラベル x 以外で誤って x に分類されたテストデータの総数を c とする．この時，c を x ラベルの False Positive と呼ぶ．また，ラベル x 以外のテストデータの総数を d とすると，x ラベルの False Positive Rate（FP Rate) は式 (16) で表される．

$$FalsePositiveRate = \frac{c}{d} \tag{16}$$

また，Precision（適合率）と，Recall（再現率）は以下の式 (17)，(18) で表される．

$$Precision = \frac{a}{a+c} \tag{17}$$

$$Recall = TruePositiveRate \tag{18}$$

この時，適合率と再現率の調和平均で表される F 値（F-measure）は式 (19) となる．

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{19}$$

F 値が高いほど，分類結果は良好である事を示している．

分類において，変化が無い枡が駒の移動先の枡と認識される事例が多く生じた．これは，実験で駒の配置をずらした事によって，変化が無い枡の画素情報（ノルムなど）が大きく変化してしまった場所である．駒の移動前の箇所が 2 箇所以上検出された場合には，将棋のルールを用いればその移動候補箇所から真の移動箇所を限定する事は可能である．しかし，ルールを警告するようなユーザ支援システムを目指す為にはそのような将棋の知識を持たずに，画像情報のみで正確に局面状況について認識を行う必要がある．

この結果は，対局において駒全体を 2，3 回崩したとしても画像情報だけでこれだけの認識が出来る事も示している．

また，成った駒の判定は決定木を用いて分類を行った所，今回の実験では R の全体画素標準偏差の差 (閾値：28.45) のみで分類が出来た．これは成り駒の文字が赤いという条件を持っている場合の結果であるので，それ以外の種類の成り駒の判定は今後の課題である．

4.2.3 打った駒の判定

前章で述べた特徴量を用い，打った駒を分類していった結果

表 4 駒の分類結果

Label	TP Rate	FP Rate	Precision	F 値
歩	1.000	0.000	1.000	1.000
香車	1.000	0.000	1.000	1.000
桂馬	1.000	0.000	1.000	1.000
銀将	1.000	0.000	1.000	1.000
金将	1.000	0.000	1.000	1.000
角行	1.000	0.000	1.000	1.000
飛車	1.000	0.000	1.000	1.000

表 5 実験した中での駒の分類数

認識結果	歩	香車	桂馬	銀将	金将	角行	飛車
歩	343	0	0	0	0	0	0
香車	0	23	0	0	0	0	0
桂馬	0	0	56	0	0	0	0
銀将	0	0	0	228	0	0	0
金将	0	0	0	0	195	0	0
角行	0	0	0	0	0	96	0
飛車	0	0	0	0	0	0	66

について示す．

ここで，駒を打つという行動は１局の中で中々起こらないので，駒を打った時の駒がある桝目のデータを全て取得していってもテストデータを多く集めるのは困難であった．そこで今回は，成り駒と王以外の駒を動かした，打ったときの対象の駒がある桝目のデータを全てテストデータとして扱っている．また駒の分類結果は，ここでは１０局対戦を行ったものの，それぞれのラベルの結果に対する平均値を表している．学習・分類の手法としては逐次最小最適化（Sequential Minimal Optimization，以下 SMO）を用いた．SMO は Support Vector Machine（SVM）の高速な実装手法として Platt[11] が提案した．これはデータ数 N に対し，従来 $O(N^2)$ のメモリ容量が必要とされていたものが $O(N)$ で十分なようになっており，従来行われていた数値計算による解法とは違い，解析的に学習の解を入手できる手法であり，学習結果に含まれる誤差が非常に少なくなるとされている．

また，特徴量におけるエッジの特徴ベクトルは，この中で情報利得が高いものを選択して用いた．これは，PPED アルゴリズムによって求められるエッジの特徴ベクトルは回転や位置ずれに弱いのでその時の初期盤面の訓練データの状況によって特徴量の変化が大きい．従って，この中で特徴選択を行った方が良いと判断したためである．実際この特徴量全部を用いたり，全部を省いたりするより，この特徴量の中で特徴選択を行った方が精度の高い結果が出た．ただこの選択によって出た結果の実験数がまだ多いとは言えないので，十分な実験をもって検証を行う必要がある．

また画素の特徴量においては，全体・上下の画素値平均や画素値標準偏差以外の，左右や 3×3，2×4，4×4，5×5 などの範囲を ROI として特徴量を求めたものも使って実験を行ったが，これらの特徴量はいずれもあると精度を落とす他，特に変化を生じないものであったので，今回の実験では除外している．

表 4 の結果を見るとわかるように，少なくとも今回の実験では，分類の成功率は 100 %となっている．これは打った時だけではなく，王・成り駒を除く他全ての駒の移動において駒の種類が何であるか判定できている事を示している．SMO モデルの構築には平均 3.2 秒ほどかかっている．パーセプトロンを用いた場合も同じ精度が出るが，モデルの構築に 150.8 秒程かかる．駒の分類数（１０局合計，テストデータ数：1007）を示したものが表 5 である．表 4 に従って全ての駒が正常に分類されているのがわかる．ここで香車のテストデータ数が極端に少ないのは，将棋において香車が動かされたり打たれたりする機会が他の駒に対して多くないためである．

また，今回の局面状況認識の一連の処理時間は，平均で約 0.15 秒ほどであった．

5. おわりに

本報では，将棋を対象として認識手法の適用実験を行い，精度の検証を行った．今回の実験結果の各認識率は将棋のルールを「ほぼ」使用せずに導いたものであるので，その条件での認識率が更に向上すれば，ユーザ支援などの機能も期待出来ると考えられる．将棋盤の桝目認識のパラメータを自動調整して盤面や背景・環境によらず完全な認識が出来るようになる事は今後の課題であり，盤面のずれにも対応出来るようにしたい．今回認識に問題が若干あった，動いた駒の判断は，将棋のルールの縛り（駒の動き・二歩など）を用いればかなりその動きが絞れるものがあり，実際にその縛りを実装に入れて実験を行った所，その正解率は大幅に良くなった．しかし，そのような縛りを持たずに画像情報のみで判断が出来るようになる事が，これからインタラクティブなシステムを作って行く上では必要になってくると考えられる．

参 考 文 献

1) 林山剛久，柳井啓司，野下浩平 「 囲碁対局テレビ番組からの棋譜自動生成システム，コンピュータビジョンとイメージメディア研究報告，Vol.2002, No.034, 2002.
2) Shiba K., Furuya T., Nishi S.and Mori K. *Automatic Go-record System using Image Processing*, IEEJ Transactions on electronics information and systems, Vol.126，Part8, pp. 980–98, 2006.
3) 垣内大樹，福山忠男，堂前篤恵，衣景増，岡崎耕三 「 画像処理による将棋棋譜の自動記録」，マルチメディア・仮想環境基礎研究会，2001.
4) 山下宏 「YSS と彩のページ」，http://www32.ocn.ne.jp/˜yss/index_j.html
5) 森山賀文，小野智司，中山茂「免疫的アルゴリズムによる画像認識の研究」，PC Conference，2003.
6) Canny J.，*A Computational Approach to Edge Detection*，IEEE Trans. Pattern Anal. Machine Intell，Vol. 8, No. 6, pp. 679-698, Nov. 1986.
7) D. H. Ballard， *Generalizing the Hough transform to detect arbitrary shapes*，Pattern Recognit., vol.13, no.2, pp.111-1222，1981.
8) David H. Douglas，Thomas K. Peucker， *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature*, Canadian Cartographer, 10, No. 2，1973.
9) Yagi M.，Adachi M.，Shibata T. Pric.27^{th} European Signal Processing Conference，Tampere Finland，2000.
10) Intel，Open Source Computer Vision Library，http://www.intel.com/technology/computing/opencv/
11) John C. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*，Microsoft Research，2000.