モーショングラフによる回避動作の実現
Generating Avoidance Motion Using Motion Graph

正岡 直樹† 尾下 真樹‡

Naoki MASAOKA† Masaki OSHITA‡

† 九州工業大学 大学院 情報工学府 情報科学専攻

† Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology

‡ 九州工業大学 大学院 情報工学研究院 システム創成情報工学研究系

‡ Department of Systems Design and Informatics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

E-mail: † masaoka@cg.ces.kyutech.ac.jp, ‡ oshita@ces.kyutech.ac.jp

1. はじめに

1.1 背景

近年、コンピュータゲームの中には、キャラクタ同士が格闘を行うものが多数製作されている。このようなゲームでは、相手の攻撃を上手く回避するような回避動作を生成する必要がある。現在のコンピュータゲームでは、モーションキャプチャ技術などを用いて作られた多数の動作データを用意しておき、ユーザの操作や状況に合わせて適切な動作データを再生することによりキャラクタの動作を実現している。しかし、攻撃動作に合わせた適切な回避動作を実現するためには、あらかじめ多数の回避動作を用意した上で、攻撃動作に応じて、適切な位置、時間で回避する動作を選択・再生する必要がある。

しかし、このような回避動作を選択するためには、２つの問題点がある。１つは、大量な動作データを扱うことが困難であることである。２つ目は、適切な動作データを探索し取得するには多くの時間がかかり、動的にアニメーションを生成することが困難であることである。そのため、現在の格闘ゲームでは、どのような位置に攻撃しても対応できるように体全体を使って大きく回避する動作のみが用いられており、攻撃に合わせて紙一重で回避するような、現実の格闘動作でみられるような自然な回避動作は実現されていない。

1.2 目的

本研究の目的は、相手の攻撃に合わせた適切な回避動作を動的に生成する手法の開発である。本研究では、大量の動作データを用いて多様な回避動作を実現するため、モーショングラフ[1]を使用する。

モーショングラフとは、Kovar らによって提案された手法であり、動作データを自動的に有向グラフに変換することで、もとの動作データの断片を組み合わせて連続的な動作を生成することを可能とする。モーショングラフは、動作データ中の類似した姿勢に注目し、類似姿勢で動作データを分け、動作の遷移を作成する。モーショングラフはエッジとノードで構成され、エッジは短い動作、ノードは姿勢を表す。一定の遷移ルールに従ってモーショングラフをたどりながら、エッジの動作を再生することで、連続的な動作を生成できる。例えば[1]では、ユーザが入力した軌道に沿って歩く動作を生成するような遷移ルールが提案されている。彼らは、各ノードから次に遷移するエッジを決める際に、生成される動作が最も入力軌道に近くなるエッジを選択する遷移ルールを用いている。

本研究では、モーショングラフを用いて、攻撃に対して、適切な回避動作を生成できるエッジの遷移を実現する手法を開発する。

1.3 アイデア

このような目的を実現するため、本研究では次の３つの提案手法を開発した。

１つ目の手法は、実行可能なモーショングラフの範囲内に、タイミングよく攻撃を回避できる回避動作が見つからない場合でも、動作の再生速度を変化させることで回避動作のタイミングを合わせる、というものである。これは、動作データの中には再生速度を多少変化させても不自然にならない動作があることに注目したものである。

２つ目の手法は、攻撃を紙一重で適切に回避できるかを評価するために、回避を行うキャラクタの身体領域・回避領域を考慮する、というものである。身体の位置を表す身体領域に加えて、攻撃を回避する位置を表す回避領域を判定に用いることで、自然な回避動作を実現する。しかし、このような領域同士の衝突判定には、多くの処理時間がかかるという問題がある。

そこで、３つ目の手法として、グラフィックスプロセッシングユニット（GPU）を用いて高速に衝突を判定する。２つの領域の衝突を判定する際に、領域同士の正確な距離を計算するのではなく、各領域を GPU メモ

リ上の画面に描画することで、領域同士の重なりの判定により衝突判定を行う。また、高速化のための他の工夫として、あらかじめ各ノードに攻撃位置に応じた回避動作のパス候補を計算して、探索時に利用する。

2. 関連研究

近年、モーショングラフを用いて格闘動作を生成する研究が行われている。

Lee ら[2]は、攻撃の目標位置が指定されたときに、攻撃動作を実現する手法を提案している。この手法では、あらかじめモーショングラフの各ノードにグリッド情報を設定し、グリッドの各点に、その位置に対して移動や攻撃を行うための適切な遷移先エッジの情報を、あらかじめ計算して記録しておく。この情報を用いることで、エッジを探索する必要がなく、動作生成を高速に実現する。しかし、この手法では、目標位置は点で表され、領域や時間を考慮していない。この手法で領域や時間を扱おうとすると、インデックスが大量に必要となったり、領域と重なる全てのグリッド点に含まれる情報を統合して繊維先を決定したりする必要がある、という問題が起きる。本研究では、3 次元空間の攻撃領域や時間を考慮して、適切な動作の選択を行う。本手法でも、一部の処理で、グリッドを利用した高速化を行っている。しかし、それだけでは領域や時間に対応できないため、GPU を用いることにより領域同士の交差判定の高速化を行っている。

Shum ら[3]は、攻撃・回避側の各キャラクタのエッジの遷移を並行して探索しながら格闘動作を生成する手法を提案している。しかし、本研究のような、回避領域や領域同士の距離を用いた適切な回避動作の選択や、再生速度の調節などの工夫は行われていないため、自然な回避動作は実現できなかった。本手法では、GPU を用いた衝突判定を利用することで、より適切な回避動作を実現できる。

図 1：衝突判定に使用する領域

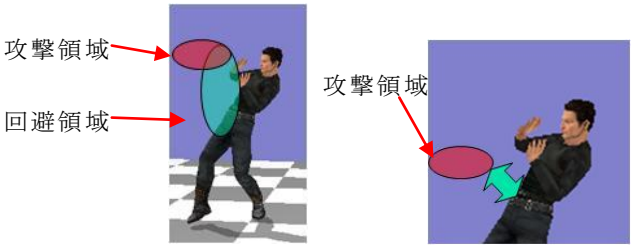

図 2：領域同士の交差判定。攻撃領域と回避領域の交差（左）攻撃領域と身体領域の距離（右）

近年、GPU を使った高速化は、様々な用途で利用されている。GPU を使った粒子法による物理シミュレーションでも、粒子同士の衝突判定が GPU で実現されている。しかし、従来の衝突判定では、図形同士の距離の判定が主に用いられており、3 次元空間での領域同士の衝突判定は実現されていなかった。これは、描画画面は 2 次元であるため、奥行きのある領域の判定を行うのは難しいという問題もある。本研究では、3 次元の領域を高さで分割することにより、3 次元の領域を 2 次元の領域で近似し、画面上の重なりを判定することで、領域同士の衝突判定を実現している。

3. 提案手法

3.1 概要

本提案手法では、任意の攻撃領域（空間・時間）の情報が与えられたときに、回避動作を行うキャラクタの動作を生成する。攻撃領域は、位置・半径・時刻の情報を持つ球の集合によって表わされる。攻撃領域の情報は、アプリケーションに応じて、攻撃を行うキャラクタの攻撃動作の情報から計算することも、今回我々が実験用に作成したプログラムのように、ユーザがクリックした任意の位置に攻撃が行われるように計算することもできる。

回避動作を行うキャラクタの動作は、モーショングラフを用いて生成する。攻撃が与えられていない時には、任意の動作ルールに従ってエッジ遷移を行うことで、キャラクタの動作が生成される。一方、攻撃情報が与えられたときには、現在再生を行っているエッジの次のノード以降のパス（複数のエッジの遷移）の候補の中から、攻撃を回避するために最も適切なパスを選択する。選択されたパスに従って、動作を再生することで、回避動作のアニメーションを生成する。また、このとき、回避動作を実現するためのパスだけでなく、パス内の各エッジの再生速度も同時に決定して、回避と攻撃のタイミングを合わせる。

3.2 パス候補の評価の基準

モーショングラフ中の現在再生中のエッジ以降の一定時間範囲内の全てのパスをパス候補と呼ぶ。このパス候補の中から再生するパスを決めるために、パス候補が攻撃に対して、どの程度自然な回避動作であるか評価し、最も適切であると考えられるパスを一つ決定する必要がある。

適切なパスを選択するために、本研究では、図 1 のように身体領域・回避領域・攻撃領域の 3 種類の領域を考慮する。身体領域とは身体がある領域（図 1 の緑色の領域）である。これは、キャラクタのある時刻における体節や間接の位置によって決められる。攻撃領

域とは攻撃発生から攻撃終了までの攻撃部位がある領域（図 1 の赤色の領域）である。これは、攻撃が発生してから、終了するまでの攻撃が通る領域を表している。回避領域とは、回避する人の回避行動前に身体領域があり、かつ回避中に身体領域がなくなる領域（図 1 の青色の領域）である。つまり、攻撃領域と回避領域が重なるときに、最もうまく回避しているように見える。これは、回避動作の開始から回避動作が終わるまでの時刻の身体領域によって決められる。本手法では、これらの領域を、球の集合によって表現している。攻撃領域・回避領域は動作データ中の攻撃・回避動作の時刻・部位を手動で指定することで、自動的に計算される。身体領域は自動で計算される。

これらの領域を用いて、回避動作を行うパスの評価を行う。パスの評価は 3 つの基準に従って決定する。

まず、ぎりぎりで回避できているかを評価する為に、身体領域と攻撃領域の距離を測る。図 2 のように赤い攻撃領域とキャラクタの身体領域の距離が近いほど、紙一重で回避する事になり、より望ましい回避動作となる。しかし、回避キャラクタの身体領域と攻撃領域の距離が負になると、その回避動作では攻撃が当たってしまうため、負になる回避動作は候補から除外する。

次に、攻撃をうまく回避するパス候補であるかを評価する。上記の距離だけでは、攻撃領域が身体の近くを通るだけで回避しているように見えないことがある。そこで、回避領域と攻撃領域の重なりの大きさに基づき回避動作を評価する。図 2 のように緑色の回避領域と赤い攻撃領域の重なりが大きいほど、回避動作でキャラクタが回避する位置に攻撃が来ることになり、自然な回避動作に見える。

最後に、攻撃にタイミングを合わせるために必要な再生速度の変化の大きさに基づき、再生速度の変化を評価する。動作の再生速度の変化が大きくなるほど、不自然な動作になる可能性が高い。そこで、回避動作のタイミングを合わせる為の再生速度の変化が少ないパス候補を優先する。

3.3 回避動作のパスを決定する処理の流れ

図 3 に示す通り、回避動作のパスを決定する処理では、攻撃領域の情報が与えられると、回避動作を実現するパスの候補の列挙、パス候補の再生速度の決定、パス候補の絞り込み、パス候補の評価と選択、という処理の流れで、実行するパスを決定する。

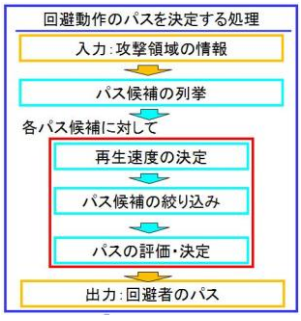

図 3：回避動作のパスを決定する処理の流れ

パス候補の列挙では、現在再生中のエッジ以降のパス（一定時間のエッジ遷移）のうち、回避動作を行うエッジを含むものを、候補として列挙する。これは、現ノードから考えられるパスの中から最も適切な回避動作を選択できるようにするためである。グリッドの情報を用いてパス候補を列挙する。一定時間が長くなるほどより適切な回避動作が含まれる可能性が高い。

パス候補の再生速度の決定では、各パス候補に対して、攻撃と回避の時間から、回避動作と攻撃動作のタイミングを合わせるためには、回避動作を含むエッジまでの各エッジの動作再生速度を何倍にする必要があるかを計算する。

パス候補の数が多い場合、評価項目の身体領域と攻撃領域の距離を全てのパス候補で計算すると、計算時間がかかってしまい、リアルタイムでの処理が困難になる。そこで、パス候補の絞り込みでは、領域同士の距離を計算する代わりに、互いの領域をテクスチャに描画し領域の重なりを計算することで、明らかに回避できないようなパス候補を削除する。

パス候補の評価と選択では、絞りこんだ各パス候補に対して、3.2 節で書かれている 3 つの基準で評価を計算し、総合的に最も評価の高いパスを一つ選択する。

4. 高速化の工夫と事前処理

本節では、本研究で提案する高速化のための手法と、各手法のために行う事前処理について説明する。

4.1 グリッドの作成

パス候補の列挙処理において、パス候補をすべて列挙すると、膨大な数のパスが考えられ、処理に時間がかかってしまい、長時間のパスを考慮することができない。そこで、この処理を高速にするために、各ノードにグリッドを設定する。キャラクタ周囲の空間をグリッド状に分割し、グリッドの各セルには、そのセルに攻撃が来たときに回避することができるパス候補の集合を設定する。よって、2 つ目の評価基準である攻撃領域と回避領域の重なりがあるものが列挙される。

事前処理では、図 4 に示すように各ノードに一つのグリッドを生成する。グリッド作成では、まず、キャラクタの周囲を細かなセルに分ける。本研究では、一辺 10cm の立方体を一つのセルとした。各セルには、ノードから一定時刻範囲内で考えられる回避動作を含むパスにおいて、そのセルの領域に回避動作の回避領域があるパスの集合を設定する。

各セルに設定するパス候補の集合を決定するために、現在のノードから一定時刻内に回避動作を含むエッジが存在する全てのパス候補を挙げる。セルに登録する

パス候補には、セルを攻撃領域とした時の回避領域と攻撃領域の重なりがあるパスとする。回避領域は身体領域が回避動作中になくなる領域のことなので、回避領域を求めるために身体領域を用いる。パス候補の回避動作の回避開始時刻の身体領域と終了時刻の身体領域のセルが重なっていないセルにそのパスを登録する。

4.2 再生速度の可変の判定

一般に、もとの動作の速度が速ければ、再生速度をある程度遅く変更しても、不自然に見えないと考えられる。また、同様に元の動作の速度が遅ければ、再生速度をある程度速く変更しても不自然に見えないと考えられる。また、ほとんど動きのない（動作速度が極端に遅い）動作は、再生速度を速くしても遅くしても、動きがないため不自然に見えないと考えられる。

このような仮定に基づき、あらかじめ各エッジに対して、そのエッジの動作を解析し、動作を速くすることが可能であるか、また遅くすることが可能であるかという情報を設定しておく。本手法では、キャラクタの全ての末端部位（右手、左手、右足、左足）の速度（一定時間の移動量）を解析することで、動作の速度を判定する。もし、末端部位の速度が遅ければ、その動作は速度が速くすることができると判定して、速くすることが可能という情報を設定する。同様に、末端部位の速度が速ければ、その動作は速度が遅くすることができると判定して、遅くすることが可能という情報を設定する。今回は、一定時間内に末端部位が 8cm 進まなければ遅い動作、20cm 進めば速い動作とし、動作内で 5cm 動かなければ静止動作とした。

4.3 領域テクスチャの作成

パス候補の絞り込み処理では、身体領域と攻撃領域の距離を判定する代わりに、2 次元のマップ上で領域同士の重なりを判定することで、攻撃が当たっているパス候補を高速に取り除く。また、評価に使われる回避領域と攻撃領域の衝突判定も、2 次元のマップ上で重なりを判定する。ここでは、簡易的に衝突判定を行い、処理時間を短縮する。この衝突判定では、あらかじめ計算しておいた回避動作の身体領域全体と攻撃動作の攻撃領域全体を、GPU メモリ上の仮想画面に描画して、領域同士の重なりを判定する。この処理は、仮想画面での重なりを判定するだけであるため、領域間の距離を計算する処理に比べて、領域間の距離する処理に比べて、高速に行うことができる。

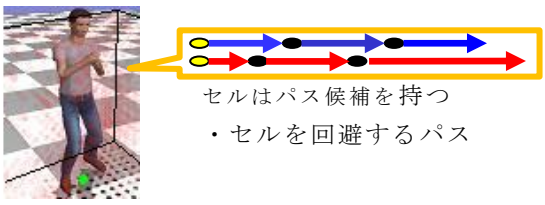

図 4：グリッド情報

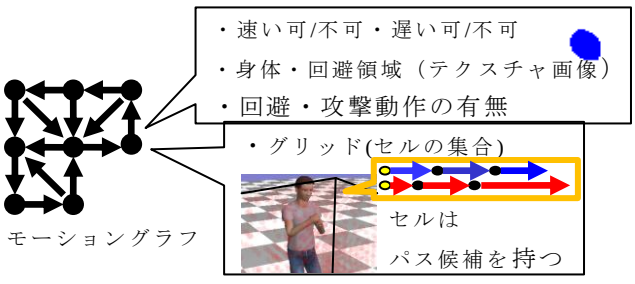

図 5：エッジ・ノードの情報

しかし、仮想画面にテクスチャを描画することで衝突判定をするため、3 次元空間中の領域同士の判定は実現が難しい。よって、キャラクタが含まれる 3 次元空間全体を、一定の高さ範囲で複数の水平な区間に分割し、各区間に身体・回避・攻撃の領域テクスチャを作成する。このとき、区間の数（テクスチャの枚数）を増やすと、正確な衝突判定ができる代わりに、処理速度が遅くなってしまう。本研究の実装では、高さ区間の幅を 30cm とした。キャラクタの回避動作一つに対して作成される身体領域と攻撃領域のテクスチャの枚数は領域の垂直方向の幅によって異なるが、本研究の実装では平均 5 枚程度作成される。

回避動作の身体・回避領域と攻撃動作の攻撃領域は決まっているため、これらを含む全てのエッジに対して、事前処理により、キャラクタを中心とするテクスチャに各領域を描画して、各領域のテクスチャを保存する。衝突判定処理では、回避動作時のキャラクタの位置・向きに応じて、テクスチャの拡大・縮小・移動・回転を行い、仮想画面に描画することで、判定する。

身体テクスチャの生成では、エッジの動作の全時間に渡って、身体領域が常に存在する領域をテクスチャに描く。ステンシルバッファを用いて、回避動作の開始時刻から終了時刻まで、各時刻の身体領域を重ねて描画し、前時刻の領域が重なった領域のみをテクスチャに残す。回避領域は開始時刻に身体領域があって、終了時刻にないような領域を描画する。回避領域もステンシルバッファを用いて、開始時刻の身体領域と全時刻で身体領域が描画される領域の差を描画する。また、同様の処理で、攻撃動作の攻撃領域については、一度でも攻撃が通る領域を描画する。

4.4 モーショングラフの作成

モーショングラフは、基本的に Kovar らの手法[1]を用いて生成する。このときに、もとの動作には、あらかじめ回避動作の時間範囲や、攻撃動作の時間範囲の情報を与えておくことで、作成時にエッジに、そのエッジが攻撃、または回避動作を含んでいるかという情報を設定する。また、付加情報として、エッジやノードに、本節で説明した情報を計算して設定する。具体的には、図 5 に示すように、エッジには身体領域のテクスチャ（回避動作をもつエッジのみ）と速度の可

変情報、ノードには、パス候補列挙のためのグリッドの情報を設定する。

5. 探索処理

本節では、図 3 に示される回避動作のパスを決定する処理を説明する。

5.1 グリッドを用いたパス候補の列挙

パス候補の列挙では、回避動作を含むパス候補を出力する。

本処理では、まず、事前に計算したグリッドを用いて、回避動作の回避領域と攻撃領域が重なる可能性のあるパス候補を出力する。現在再生しているエッジの次のノードのグリッドを参照し、与えられた攻撃領域と重なる全てのグリッド中のセルを列挙する。次に、パス動作の開始時刻から、回避動作の開始時刻までに、回避領域に向かって大きく移動するのは不自然であるため、キャラクタの移動距離が最も短くなるように、回避するキャラクタに最も近いセルから回避パスの集合を取り出す。そして決定されたセルに、パス候補の集合が存在すれば、その候補の集合を次の処理に渡す。

一方、上記の処理で決定されたセルにパス候補が存在しない場合は、回避領域と攻撃領域が重なるような回避動作は存在しないということであるため、現在再生しているエッジの次のノード以降の全てのパス候補を探索して列挙する。具体的には、一定時間の深さでグラフを探索して、回避動作を含まないパスも含めた全てのパス候補の集合として列挙し、次の処理に渡す。

5.2 パスの再生速度の決定

パス候補の再生速度の決定では、5.1 節の処理で出力された全てのパス候補について、回避動作の回避開始時間と攻撃動作の攻撃開始時間のタイミングを一致させるための、各エッジの再生速度を計算する。まず、両者の時間を基に、回避動作の開始までの動作再生速度を速くする必要があるのか、遅くする必要があるのか、速度比率を計算する。再生速度を速くする必要がある場合は、パス候補の中に含まれるエッジの内、速くすることが可能なエッジのみの速度を変更し、その他のエッジの速度は変更しない。同じく、遅くする必要がある場合は、パス候補の中に含まれるエッジの内、遅くすることが可能なエッジのみの速度を変更する。このような条件にもとづき、再生速度の変更が可能なすべてのエッジを列挙する。その後、対象の全エッジの再生速度を同一比率 p で変更することで、攻撃動作と回避動作のタイミングを一致させる。ただし、条件に該当するエッジがパス候補の中に全く存在しない場合には、そのパス候補は使用しない。

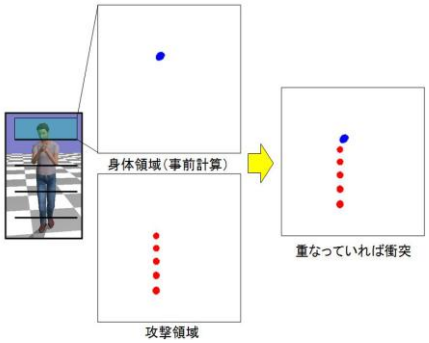

図 6：GPU 処理

5.3 GPU を用いたパス候補の絞り込み

次に、パス候補を絞り込む為に、GPU を用いた交差判定を行う。4.3 節の説明の通り、パス候補の身体領域と、与えられた攻撃領域の交差判定を行い、攻撃を回避できない可能性の高いパス候補を除外する。本処理の判定では、4.3 節の説明のように、２次元空間で判定を行うため、正確な交差判定はできないが、攻撃が当たるパス候補を高速に判定できる。近似領域で判定するため、厳密には回避できるパス候補も、間違って回避できないと判断してしまうことがある。

判定時には、攻撃領域は全てのパス候補で変わらないため、まず、図 6 のように、各高さ区間に GPU 上の仮想画面に、パス開始時のキャラクタの位置・向きを基準とするローカル座標系で、攻撃領域を描画する。そして、用意しておいた回避動作の身体領域のテクスチャを、座標系が一致するように平行移動・回転を行って描画する。そして、OpenGL の機能を用いて、すでに描画されている攻撃領域と重なった身体領域のピクセル数をカウントする。このとき 1 ピクセルでも重なっていればそのパス候補ではキャラクタの身体と攻撃が衝突したと判断し、そのパス候補を削除する。

5.4 パス候補の評価と選択

残ったパス候補を 3.2 節で挙げた 3 つの基準にもとづいて評価し、最も評価の高いパスを決定する。評価関数は次の式(1)で表わされる。

$$V = w_a V_a + w_d V_d + w_t V_t \qquad (1)$$

V はパスの評価値を表しており V_a、V_d、V_t はそれぞれ回避の評価値、距離の評価値、時間変化の評価値を表している。w_a、w_d、w_t は回避の重み、距離の重み、時間変化の重みを表す係数である。全ての項目で評価値が小さいほど評価が高くなる。

回避領域と攻撃領域の重なりの評価値 V_a は、GPU を用いた衝突判定と同様の方法で行う。身体領域テクスチャではなく回避領域テクスチャを用いて、攻撃領域との重なりのピクセル数をカウントする。領域の重なりが多いほど、自然な回避動作であるため V_a の値が小さくなり高評価となるようにした。

身体領域と攻撃領域の距離の評価値 V_d は、全ての身体領域を表す球と攻撃領域を表す球を全ての時刻で比較し、最小となる距離により評価する。距離が短いほど紙一重で回避する動作であるため、高評価となるようにした。

再生速度の変化は5.2節で計算した変数 p に基づき、次の式(2)で評価する。

$$V_t = (p-1)^2 \quad (2)$$

再生速度の比率が 1 に近いほど、高評価になるようにした。また、1 倍から離れれば離れるほど、評価が低くなるようになっている。

6. 実験

6.1 実験概要

提案手法の評価のため、提案手法を実装し、ユーザが任意の位置をマウスで指定すると、指定された位置へ球を飛ばし、その球の軌跡を攻撃領域として回避動作を生成するテストプログラムを開発した。キャラクタ同士の格闘を行うプログラムでは、評価関数や適切な回避を実行できるか確認するために、任意の攻撃領域を指定することが難しいため、今回は任意の攻撃領域を指定できるプログラムを開発した。そのため、パス候補の絞りこみ処理では、攻撃領域のテクスチャを貼る代わりに、攻撃領域をリアルタイムで計算する。今回の実験では、合計約 2 分 40 秒の動作をモーションキャプチャで作成し、モーショングラフを作成した。入力動作データには、8 回の回避動作が含まれている。

提案手法の実行速度を評価するため、パス選択処理にかかる処理時間を計測した。また、本手法の高速化の工夫の効果を確認するため、グリッド使用時の GPU を使用時と未使用時、グリッド未使用時の GPU を使用時と未使用時での処理速度を計測した。それぞれの場合で、ランダムに与えた攻撃動作を回避する動作を 6 回実行し、それぞれの平均の実行速度を計測した。実行環境は、CPU：Pentium4 2.53GHz VGA：GeForce FX5700 256MBMemory：512MB である。

6.2 実験結果

実験結果を表 1 に示す。グリッド中にはパス候補がある場合は平均 5 個程度、ない場合は平均 100 個ほどのパス候補が列挙され、GPU 処理により絞込みを行うことで、それぞれ 3 個と 70 個程度に候補を減らすことができ、全てのパス候補を距離にもとづいて衝突判定する場合と比べて、高速化することができた。表 1 よりグリッドが使われなかった場合では約半分の処理時間で行うことができていることがわかる。しかし、本実験では、約 90%の場合、グリッド中にパス候補が存在した。グリッドが使われた場合では、あまり計算時間の短縮は見られなかった。

処理全体では、グリッド未使用時 250ms、使用時は 20ms 程度かかった。1 フレームを 30ms と考えると、攻撃情報が与えられた直後の 1 フレームで回避動作を決めることは困難であるが、通常、攻撃動作が決まってから、回避動作を実行するまでには時間があるため、パスを決定する処理をマルチスレッドで処理するなど工夫することで、リアルタイムに回避動作を生成する必要のある用途にも利用可能であると考えられる。

なお、出力された回避動作の評価では、回避できないパスを選択することや、回避しているように見えない回避動作を行うことが見られ、衝突判定が正確に行われていないことと、回避領域と攻撃領域の交差判定が正確に機能していない可能性があると考えられる。

GPU での高速化では、予想していたよりもあまり高速化はできなかった。簡易的な衝突判定を行い、パス候補を減らすことが目的であったが、あまりパス候補を減らすことができなかったことが原因であると考えられる。また、グリッド中にパス候補がある場合には、もともとパス候補が少ないため、計算時間を短縮することができなかった。

表 1：パス選択の処理時間(単位:ms)

		処理時間
グリッド中にパス候補なし	GPU 処理あり	225
	GPU 処理なし	496
グリッド中にパス候補あり	GPU 処理あり	14
	GPU 処理なし	18

7. 終わりに

本研究では、適切な回避動作を実現するような手法を提案した。領域を用いた評価やグリッドの情報、GPU 処理を用いることにより、それらを用いない場合と比べて、適切な回避動作の選択と高速化を実現した。今後の課題として、リアルタイムでの動作を実現するためにマルチスレッド化等が挙げられる。本論文では球を回避するという動作を生成したが、二人のキャラクタの格闘動作を再現する動作を作成できるようにする。

また、今回の手法では、攻撃を回避できない場合や、回避しているように見えないような回避動作が生成されてしまう場合があったため、より適切な回避動作を選択できるような評価基準が必要である。また、処理速度を向上するために、他の処理にも GPU を用いた高速化を行うなどの工夫する必要がある。

なお、現在では、再生速度の変更以外には動作に変更を加えていないが、今後は、必要に応じ、モーションワーピングなどを利用し、動作中の姿勢の変化をすることで、適切な回避動作を生成できるようにすることも検討していきたいと考えている。

文　献

[1] Lucas Kovar, Michael Gleicher, Frederic H. Pighin. Motion graphs. SIGGRAPH 2002, pp. 473-482, 2002.

[2] Jehee Lee, Kang Hoon Lee. Precomputing avatar behavior from human motion data. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004, pp. 79-87, 2004.

[3] Hubert P. H. Shum, Taku Komura, Shuntaro Yamazaki. Simulating competitive interactions using singly captured motions. Virtual Reality Software and Technology 2007,pp. 65-72, 2007.