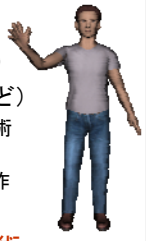
コンピュータグラフィックス特論 II

第14回　キャラクタアニメーション（5）

九州工業大学　尾下 真樹

キャラクタ・アニメーション

- CGにより表現された人体モデル（キャラクタ）のアニメーションを実現するための技術
- キャラクタ・アニメーションの用途
  - オフライン・アニメーション（映画など）
  - オンライン・アニメーション（ゲームなど）
    - どちらの用途でも使われる基本的な技術は同じ（データ量や詳細度が異なる）
    - 後者の用途では、インタラクティブな動作を実現するための工夫が必要になる
- 人体モデル・動作データの処理技術



全体の内容

- 人体モデル（骨格・姿勢・動作）の表現
- 人体モデル・動作データの作成方法
- サンプルプログラム
- 順運動学
- 逆運動学
- 姿勢補間
- 動作接続・遷移・補間
- 動作変形・生成・制御

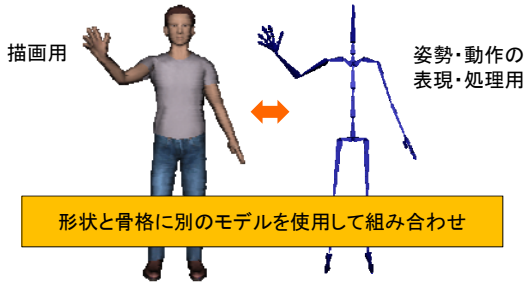
今回の内容

- 動作変形
- 動力学を考慮した動作生成
- 動作状態機械
- 自律動作制御

前回までの復習

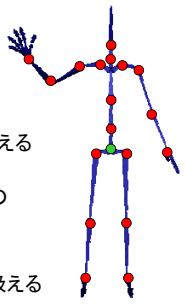
人体モデルの表現

形状モデル（ポリゴンモデル）　骨格モデル（多関節体）



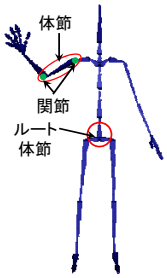
骨格モデルの表現

- 多関節体モデルによる表現
  - 複数の体節（部位）が関節で接続されたモデル
  - 体節
    - 多関節体の各部位、剛体として扱える
    - 複数の関節が接続されており、体節の長さや体節内での各関節の接続位置は固定
  - 関節
    - 2つの体節の間を接続、点として扱える
    - 関節の回転により姿勢が変化する



骨格・姿勢の表現方法

- 骨格情報と姿勢情報を分ける
- 骨格情報の中で、関節・体節を分ける
  - 体節
    - 複数の関節と接続
    - 各関節の接続位置
      - 体節のローカル座標系
  - 関節
    - 2つの体節の間を接続
      - ルート側・末端側の体節



骨格モデルの表現方法

- 骨格情報の中で、体節と関節を分ける

```
// 人体モデルの体節を表す構造体
struct  Segment
{
    // 接続関節
    vector< Joint * >      joints;
    // 各関節の接続位置（体節のローカル座標系）
    vector< Point3f >      joint_positions;
};

// 人体モデルの関節を表す構造体
struct  Joint
{
    // 接続体節
    Segment *              segments[ 2 ];
};
```

```
// 人体モデルの骨格を表す構造体
struct  Skeleton
{
    // 体節・関節の配列
    vector< Segment * >  segments;
    vector< Joint * >       joints;
};
```

姿勢の表現方法

- 骨格情報と姿勢情報を分ける

```
// 人体モデルの姿勢を表す構造体
struct  Posture
{
    Skeleton *  body;
    Point3f     root_pos;        // ルートの位置
    Matrix3f    root_ori;        // ルートの向き（回転行列表現）
    Matrix3f *  joint_rotations; // 各関節の回転（回転行列表現）
                                 // [リンク番号] リンク数分の配列
};
```

骨格モデルの表現方法のまとめ

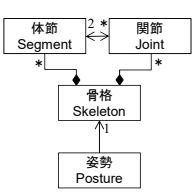
- 骨格情報と姿勢情報を分ける
- 骨格情報の中で、体節と関節を分ける

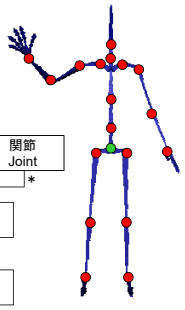
```
// 多関節体の体節を表す構造体
struct  Segment

// 多関節体の関節を表す構造体
struct  Joint

// 多関節体の骨格を表す構造体
struct  Skeleton

// 多関節体の姿勢を表す構造体
struct  Posture
```





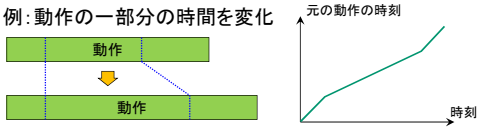
動作変形

動作データの変形

- 既存の動作データを編集したり、動作データをもとに新しい動作を生成するための技術
  - 動作補間、動作遷移・接続（前回の授業で説明）
  - タイム・ワーピング
  - モーション・ワーピング
  - モーション・リターゲッティング
  - （動力学を考慮した動作生成）
- 一般的なアニメーション編集ソフトウェアでは、これらの機能を使うことで動作データの変形を行える
  - オンラインアニメーションでも、動作データを動的に変形するために、これらの技術が用いられることがある

タイム・ワーピング

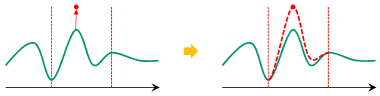
- 動作データの再生時間を変化させることで、動作の速度を変更する
  - タイムワープ関数により時間変化を指定
    - 例：動作の一部分の時間を変化



  - 再生時間を大幅に変化させたり、再生速度が急激に変化したりすると不自然な動作になるので、タイムワープ関数を滑らかにするなど工夫が必要

モーション・ワーピング

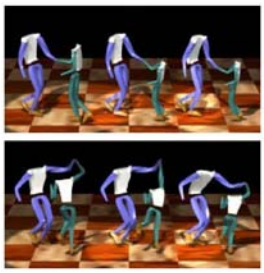
- 動作データ中のキーフレームの姿勢変化に合わせて前後の動作を滑らかに変化させる
  - キーフレームの時刻、変形後の姿勢、変形を適用する区間の開始・終了時刻を指定
  - 区間中の各時刻の姿勢に、姿勢ブレンドを適用
    - 時刻に応じてブレンド比率を変化（ブレンド比率関数）



  - 姿勢変化が大きいと、不自然になる可能性がある
    - 変形区間やブレンド比率関数の設定が重要

モーション・リターゲッティング

- 骨格の異なるキャラクタ間で動作データを変換
  - 例：モーションキャプチャデータを、元の俳優の骨格から、実際のキャラクタの骨格用に修正
  - 単純に関節角度をコピーするだけでは、足の位置がずれたりするため修正が必要



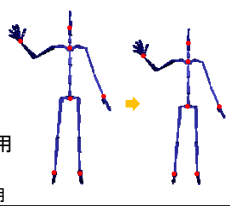
[Gleicher 98]

モーション・リターゲッティング

- モーション・リターゲッティングの計算方法
- 動作最適化問題として解くのが一般的
  - 目標関数 f(動作) を定義
    - 変形後の各姿勢が元の姿勢になるべく近くなる
    - 変形後の各姿勢が元の姿勢の制約条件を満たす（足が地面に接しているときに固定したり、キャラクタ同士が接触しているときに接触を保つなど）
    - 動作中の前後の姿勢が連続的に変化するようにする
  - f(動作) がなるべく小さくなるように、動作全体を最適化（変形）

モーション・リターゲッティング

- 簡易的なモーション・リターゲッティング
  - 部位の位置にもとづくリターゲッティング
  - 変換後の姿勢の各部位の位置（＋向き）を計算し、逆運動学計算により姿勢を生成
    - 例：身長の比率にもとづき部位の位置をスケーリング
    - 接触時は、接触位置から部位の位置を決定
    - 前後のフレームで姿勢の連続性を保つようにIKを適用
      - 解析的IKを適用
      - 前フレームの姿勢にIKを適用



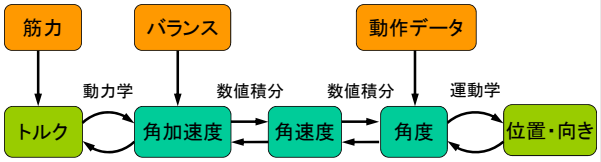
動力学を考慮した動作生成

動力学を考慮したアニメーション

- 運動学（キネマティックス）
  - 多関節体の機構を扱うための手法（関節回転と各部位の位置・向きの関係を扱う）
  - 現在は、こちらが主に使われている
  - 力学的に正しい動きが生成されることは保証されない
- 動力学（ダイナミックス）
  - 運動学に加えて、力学的な要素を考慮する手法

動力学を考慮したアニメーション

- 主に2種類の方法がある
  - 動力学シミュレーションによるアプローチ
  - 動作最適化によるアプローチ
- 多関節体の動力学

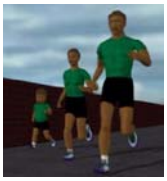


動力学シミュレーション

- 動力学シミュレーション
  - 本物のロボットと同様に、制御理論に基づいて関節のトルクを制御するコントローラを構築
  - 動力学シミュレーションを行って動きを生成
    - 人間らしい動きを生成することは困難
    - 結果をうまく制御することが難しい

$$\tau_i = K\left(\theta_{i,goal} - \theta_{i,curr}\right) - K\left(\dot{\theta}_{i,goal} - \dot{\theta}_{i,curr}\right)$$

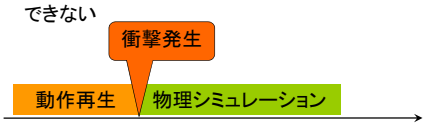
PD制御
目標角度－現在角度に適当な係数をかけることで各関節のトルクを決定



[Hodgins 97]

動力学シミュレーションの応用例

- ラグドール・シミュレーション
  - 一部のコンピュータゲームで使われている
  - 通常は、動作データを再生
  - 衝突時に、物理シミュレーションに切り替え
    - 受動的な動作のみで、能動的な動作は実現できない
    - 倒れずに途中で回復して元に戻るような動作は実現できない



動作最適化

- 動作最適化
  - 関節トルクやバランスの不自然さを評価する関数 f（動作）を設計
  - f（動作）がなるべく小さくなるように動作データを少しずつ修正する
    - 計算に非常に時間がかかる
    - 人間のような多関節体には、自由度が高すぎてそのまま適用するのは困難

[Popovic 98]

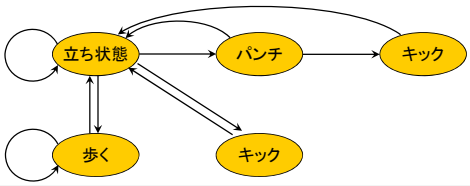
動作状態機械

動作データの再生

- オンラインでの動作再生
  - 動作を連続的に行う必要がある
  - ユーザの操作や周囲の状況に応じて適切な動作を行う必要がある
- ゲームなどでの一般的な方法
  - あらかじめ基本的な動作データを多数用意
  - 適切な動作データを順番に再生することで、連続的な動作再生を実現
  - 動作データの間がなめらかにつながるように作成しておく必要がある

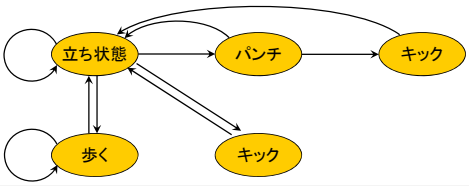
動作状態機械

- あらかじめ各動作データ間がつながるように考慮して動作データとそのつながりを作成
  - 前動作の終了姿勢と次動作の開始姿勢が同じになるように動作を作成する
  - 各動作間の遷移のタイミングや方法を設定



動作状態機械

- 状態機械（State Machine）、動作ツリー、動作グラフなどの呼び方がある
  - 「モーショングラフ」は、別の技術（後述）を指す
  - データ構造としては、ノードが短い動作、エッジが前後の動作の接続を表す有向グラフとなる



動作状態機械の利用

- 動作状態機械の作成
  - 各動作データや、動作間の接続の設定を、専用の編集用ソフトウェア上でデザイナーが作成
    - 前後の動作がうまくつながるように動作データを作成
    - 動作接続・遷移のパラメタを調整
- 動作状態機械の利用
  - 動作状態機械に従って、連続的に動作を再生
  - 次にどの動作状態に遷移するかを、利用者の操作や状況に応じて決定
  - プログラム上での処理が必要

動作状態機械による方法の問題点

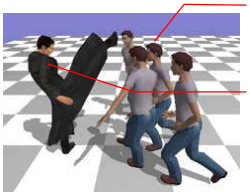
- 動作状態機械の作成には多くの手間がかかる
- 一定動作の繰り返ししかできない
  - ワンパターンな動きになる
  - 特に衝突や外力などの力学的な影響に応じた自然な動作が困難

- 自律動作の実現は困難
  - 敵キャラクタや群集などを自律的に動かすためには、適切な動作ルールが必要になる

自律動作制御

自律動作制御

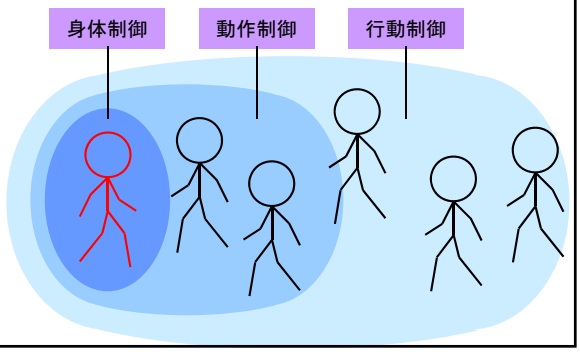
- 自律動作制御
  - 与えられた指示や周囲の状況に応じて、適切な動作を生成
  - 例:コンピュータゲームにおける自律動作制御



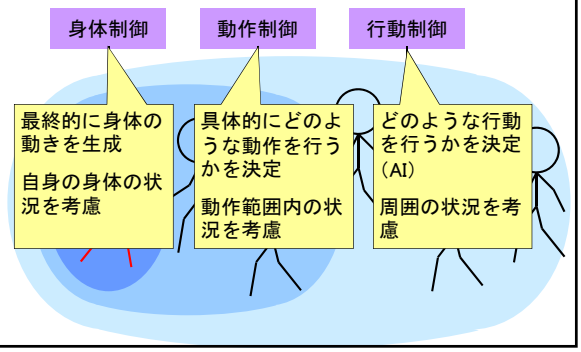
Non-Player Character
完全に自律的に動作を行う必要がある

Player Character
半自律的に動作を行う必要がある
(ex. プレイヤーの抽象的な操作に応じて具体的な動作を実現)

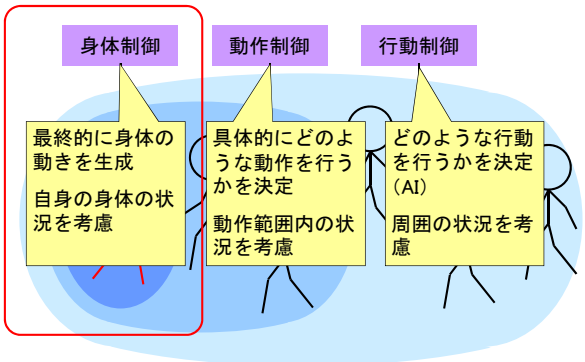
自律動作制御のレベル



自律動作制御のレベル



身体制御の技術



身体制御の課題

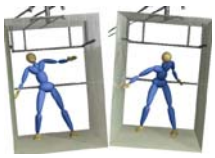
- 衝突や外力などの力学的な影響に応じた身体制御(動作生成)
  - 動力学シミュレーションを使う方法では、受動的な動作は生成できるが、能動的な動作は生成できない

身体制御の技術

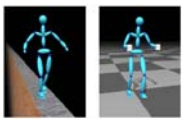
- モデルベース手法 vs データベース手法
  - モデルベース手法
    - 何らかのモデル(プログラム)に従って動きを生成
    - 個人ごとの動きの変化の実現などは困難
  - データベース手法
    - 大量のデータを用意しておくことで、問題を解決
    - 準備が必要、適切なデータの検索・変形は困難
  - ハイブリッド手法
    - 両方をうまく組み合わせる手法
    - どのように組み合わせるかは、難しい

関連研究(モデルベース手法)

- 関節角度制御 [Jain 2009]
  - 力学的要素を考慮して、最適な角度(+速度、加速度)を決定
  - 最適化問題に帰着

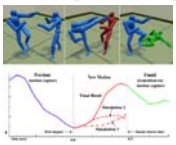


- 歩行制御モデル [Wang 2009]
  - 動力学シミュレーションによる歩行動作
  - 安定性を上げるための工夫



関連研究(ハイブリッド手法)

- シミュレーションと動作データの利用 [Zordan 05]
  - 衝撃後の動作を物理シミュレーションで計算(ODE)
  - 後にうまくつながるような転倒動作をデータベースから検索



- 動作データの選択と変形 [Arikan 05]
  - 衝撃・姿勢に応じたリアクション動作データを検索
  - 各部位に一定の速度を加えてIKにより姿勢を変形

関連技術(ハイブリッド手法?)

- Endorphin, Euphoria (Natural Motion 社)
  - タイムライン上でビヘイビアを指定すると自動的に動作生成

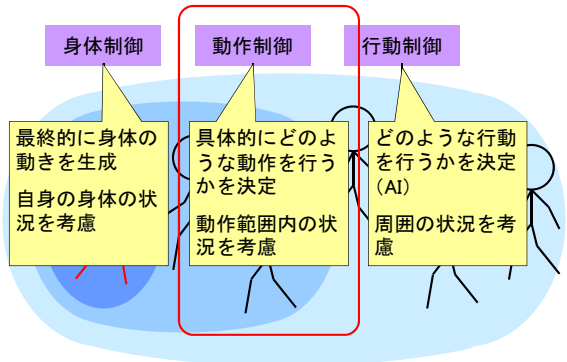
Natural Motion Ltd

    - さまざまなビヘイビアの制御モデルが用意されており、ビヘイビアの種類やタイミングを指定することで、自動的に動作が生成される(Endorphin)
    - ゲーム用のミドルウェアも存在(Euphoria)
    - 詳細な内部技術は公開されていない
    - モデルベースなので、動作を細かく変更したり、キャラクタの個性を出したりすことは難しい?

ロボット工学との関連

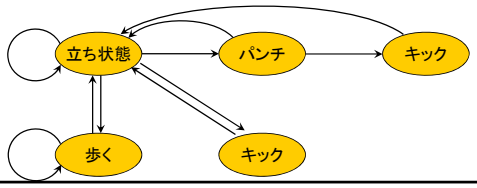
- ロボット制御とはやや目的や手段が異なる
  - ロボット制御
    - 力学的に正確な制御が必要
    - 現在ではごく単純な動作しか実現できていない
    - 人間らしい動きの実現はあまり考えられていない
    - 人間とはモータの機構も異なる
  - アニメーション
    - 力学的には厳密でなくても人間らしい動きを実現
  - 動力学などの基礎的な理論はかなり共通
- 将来的にはヒューマノイドロボットの技術がゲームにも応用可能になる?

動作制御



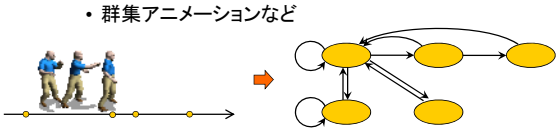
動作制御の課題

- 状況に応じた適切な動作制御（動作プランニング）
  - 動作状態機械を使う方法では、限られた種類の動作しか行うことができない
  - 作成に手間がかかる



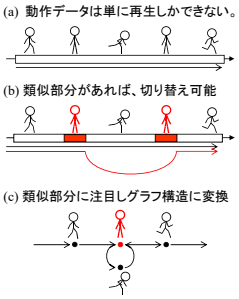
動作状態機械の自動作成

- 長い動作データを解析して、動作状態機械を自動的に作成する技術もある
  → モーショングラフ
  - 自動的に生成されたグラフ構造は整理されていないため、ゲームなどへの利用は難しい？
  - キャラクタをランダムに動かすような用途に有効
    - 群集アニメーションなど



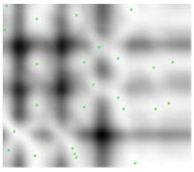
モーショングラフ

- モーショングラフ [Kovar 02]
  - モーションキャプチャデータは、有向グラフ構造に変換できる
  - ノードを順に遷移することで、連続的な動作を生成できる
    - 遷移のルールが重要



モーショングラフの作成方法

- モーション中の各フレーム同士の姿勢間の距離を計算（マッチウェブ）
  - 姿勢同士の距離の評価方法も重要
    - 例：主要部位同士の距離の和、ワンスキンモデルの頂点同士の距離の和、など

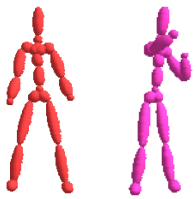


Kovar 02

- マッチウェブ中の極小点をモーショングラフのノードとし、ノード間の動作をエッジとする

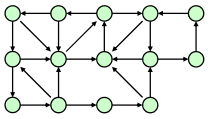
姿勢間の距離の計算方法

- 任意の2つの姿勢間の距離（どれだけ似ているか）の評価方法
- 関節角度の差の総和
  - 関節によって姿勢全体の見た目への影響は異なる
- 位置の差の総和
  - ワンスキンモデルの全頂点 or 主要部位同士の距離の和など
  - 最初に姿勢全体の位置・向きを合わせる必要がある
    - 点集合の平均・分散から計算 [Kovar 02]



モーショングラフによる動作生成

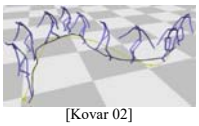
- データ構造
  - ノード・・・姿勢
  - エッジ・・・動作
    - 一般的な動作ツリーとは逆



- 動作生成
  - ノードを辿りながらエッジの動作を再生していくことで、動作を生成
    - 実際には、ノードの前後で、動作ブレンディングや、足を地面に固定するための IK が必要
  - ノード遷移のためのルールが重要

関連研究

- モーショングラフにおけるノード選択条件
  - 歩行パスによる条件[Kovar 02]
  - 手先位置の条件[Lee 05]
  - 動作の種類の条件[Arikan 04]
  - 音楽に合わせた動作


[Kovar 02]

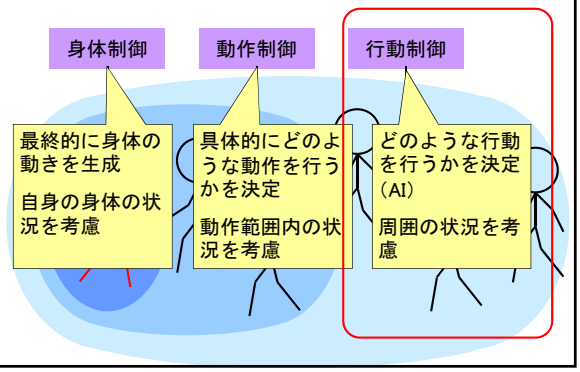

[Lee 04]


[Arikan 04]

モーショングラフの利点

- モーションデータから自動的に構築できる
- 連続的な動作が無限に生成できる
  - 毎回異なる動作が生成される
  - 基本的には同じ動作の繰り返しだが、ノード数が十分にあれば、十分に自然に見える
- デメリット
  - 毎回異なる動作が生成される
    - ゲームによっては必ずしも良いとは限らない？
  - 適切な遷移ルールが必要（計算速度の考慮）

行動制御



行動制御の課題

- 動作ツリーの作成には多くの手間がかかる
- 一定動作の繰り返ししかできない
  - ワンパターンな動きになる
  - 特に衝突や外力などの力学的な影響に応じた自然な動作が困難
- 自律動作の実現は困難
  - 敵キャラクタや群集などを自律的に動かすためには、適切な動作ルールが必要になる

群集モデル

- 多数の人間の動きをシミュレートする技術
  - 映画やコンピュータゲームでの群衆シーンで利用
    - 個々のキャラクタの動きを作成するのは不可能
  - 一定のルールに従って各キャラクタを制御
    - 単純なルールでも、キャラクタ同士が相互作用し合って、ある程度リアルな動きが生成される


Lord of the Rings, 2002


Chronicles of NARNIA, 2005


NINETY-NINE NIGHTS, 2005

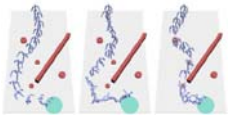
群集モデルの実現方法

- プログラムによる動作ルールの記述
  - if（条件）then（動作）の形式の動作ルールを多数、ゲームなどのプログラム内に直接記述
- GUI環境による高度な動作ルールの記述
  - 群衆アニメーションソフトMassive が有名
  - 多数の映画で使用されている
  - 状態マシン＋ファジィルール による条件を、GUI画面上で記述
  - かなりの知識や労力が必要


Massive

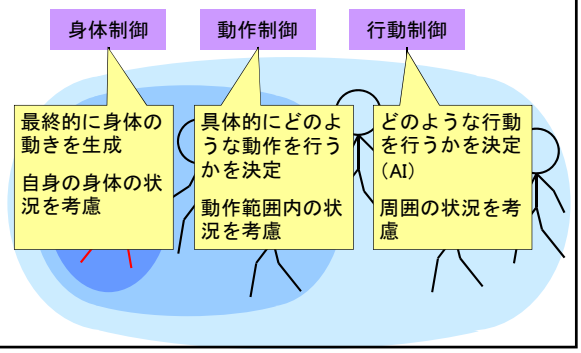
関連研究

- 行動ルールの自動学習
  - 逆強化学習 [Lee 2010]
    - 動作計画のための評価基準を自動学習



- 高度なルールベースの手法
  - 視界にもとづく動作決定 [Ondrej 2010]

自律動作制御のまとめ



まとめ

- 動作変形
- 動力学を考慮した動作生成
- 動作状態機械
- 自律動作制御

全体のまとめ

- 人体モデル（骨格・姿勢・動作）の表現
- 人体モデル・動作データの作成方法
- サンプルプログラム
- 順運動学
- 逆運動学
- 姿勢補間
- 動作接続・遷移・補間
- 動作変形・生成・制御

次回以降の予定

- 次回 授業最終回
- 期末試験は無し
- 第5回 レポート課題 提出締切