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

2021年度

今回の内容

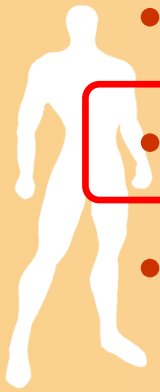
- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ

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、人体形状変形モデル
- 姿勢補間、キーフレーム動作再生、動作補間
- 動作接続・遷移、動作変形
- 逆運動学、モーションキャプチャ
- 動作生成・制御



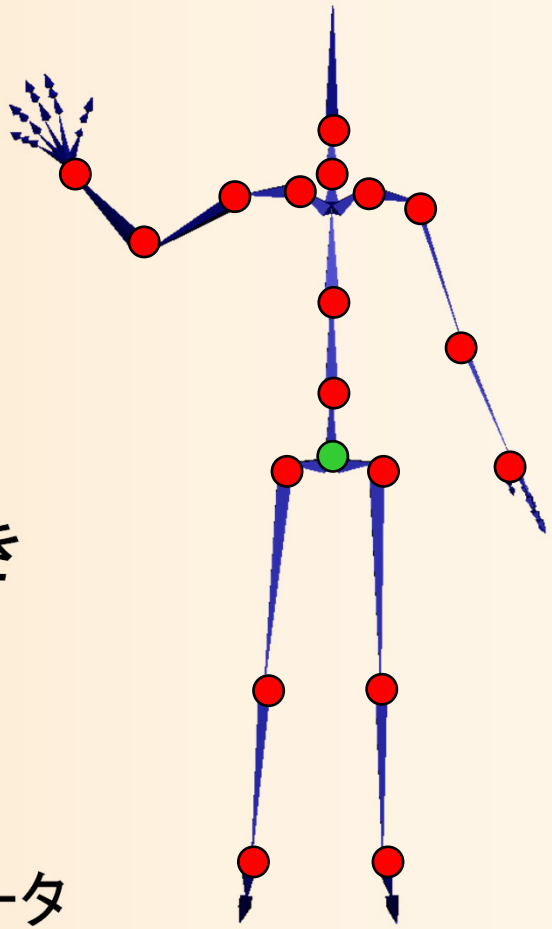
今回の内容

- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ

前回までの復習

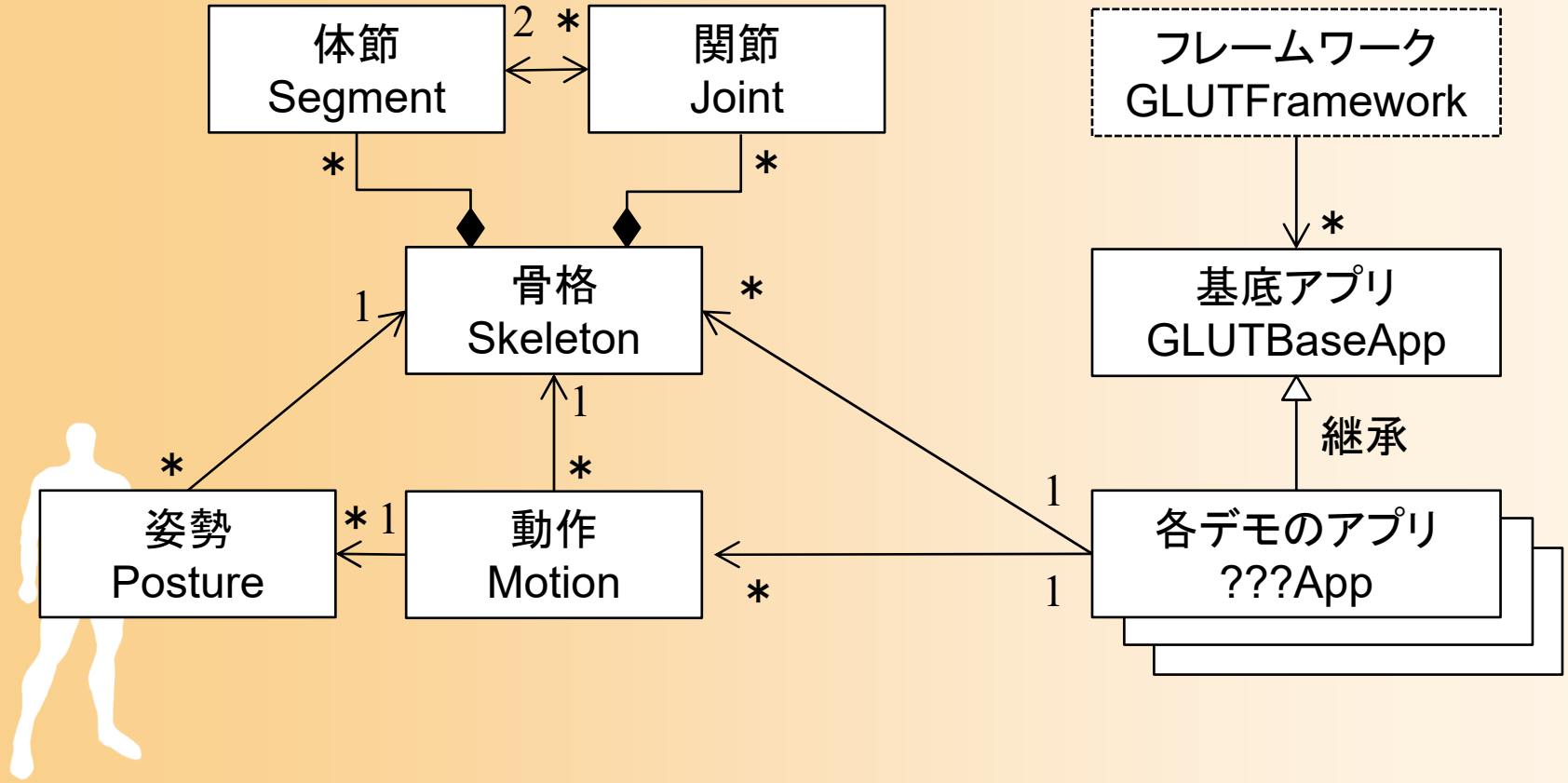
骨格・姿勢・動作の表現

- 人体の骨格の表現
  - 多関節体モデルによる表現
  - 複数の体節と関節
    - 関節は2つの体節の間を接続
- 姿勢の表現
  - 全関節の回転 ＋ 腰の位置・向き
- 動作の表現
  - 姿勢の時間変化
    - 一定間隔 or キーフレーム動作データ



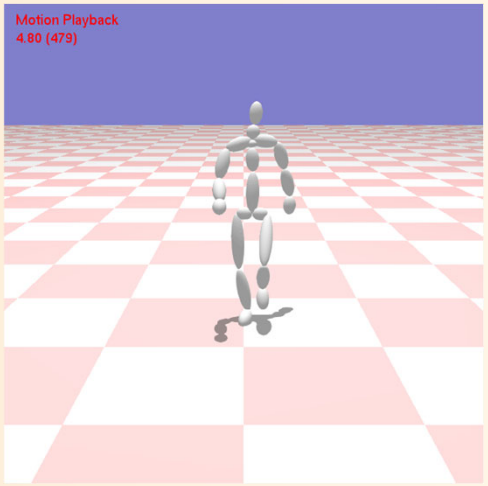
クラス図

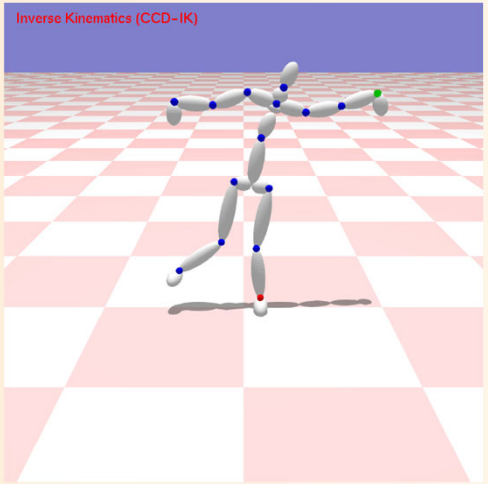
- クラス・構造体間の関係



デモプログラム

- 複数のアプリケーションを含む
  - マウスの中ボタン or m キーで切り替え
- 動作再生
- キーフレーム動作再生
- 順運動学計算
- 姿勢補間
- 動作補間（2つの動作の補間）
- 動作接続・遷移
- 動作変形
- 逆運動学計算（CCD-IK）





サンプルプログラム

- デモプログラムの一部
  - 骨格・姿勢・動作のデータ構造定義（SimpleHumn.h/cpp）
  - BVH動作クラス（BVH.h/cpp）
  - アプリケーションの基底クラス（GLUTBaseApp）
    - 各イベント処理のためのメソッドの定義を含む
    - 本クラスを派生させて各アプリケーションクラスを定義
  - コールバック関数（SimpleHumanGLUT.h/cpp）
    - GLUTBaseAppの定義・実装、全アプリケーションを管理・切替
    - アプリケーションのイベント処理を呼び出すGLUTコールバック関数
  - メイン処理（SimpleHumanMain.cpp）
  - 各アプリケーションの定義・実装（???App.h/.cpp）
    - 主要な処理を各自で実装（レポート課題）

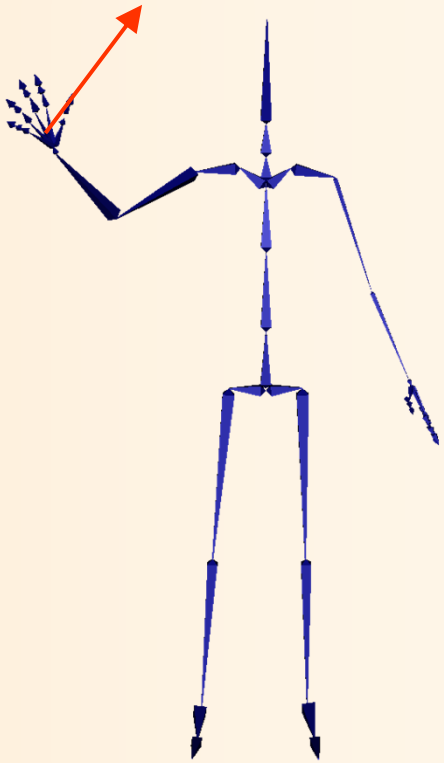
全体の内容

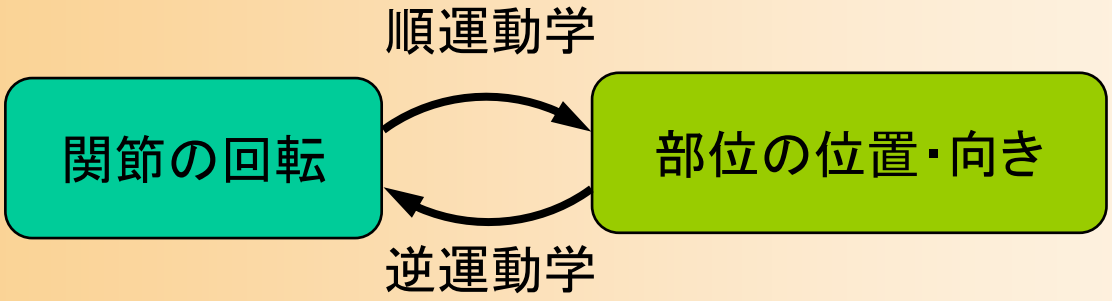
- 人体モデル（骨格・姿勢・動作）の表現
- 人体モデル・動作データの作成方法
- サンプルプログラム
- 順運動学、人体形状変形モデル
- 姿勢補間、キーフレーム動作再生、動作補間
- 動作接続・遷移、動作変形
- 逆運動学、モーションキャプチャ
- 動作生成・制御

逆運動学

運動学(復習)

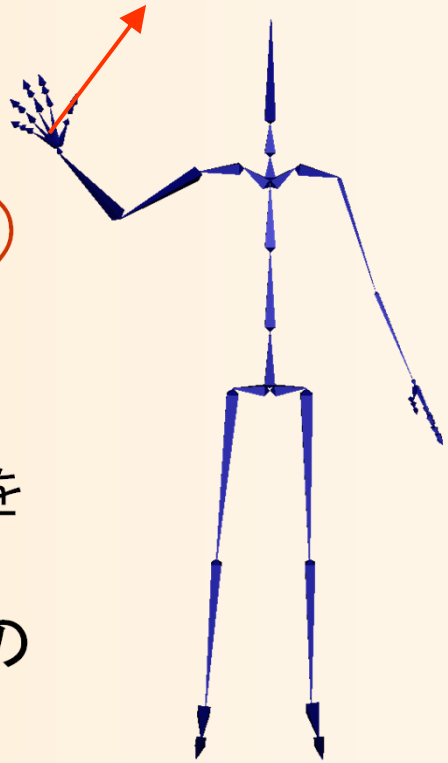
- 運動学(キネマティックス)
  - 多関節体の姿勢表現の基礎となる考え方
  - 人間の姿勢は、全関節の回転により表現できる
  - 関節の回転と各部位の位置・向きの間の関係を計算するための手法





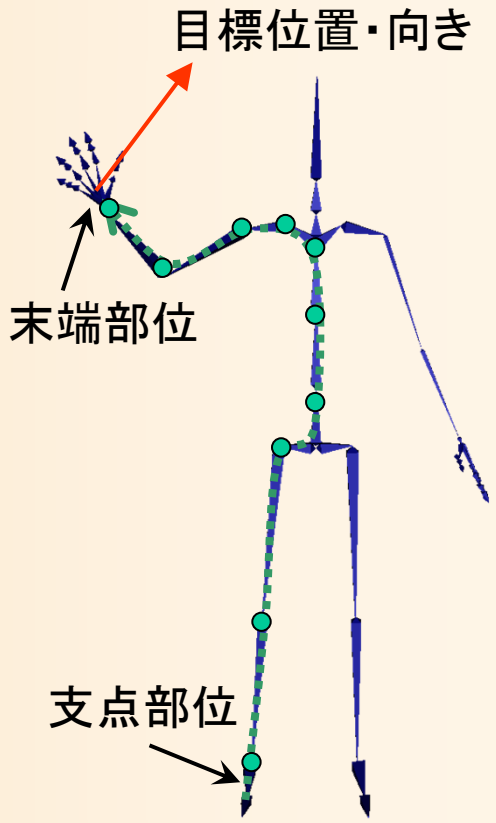
順運動学と逆運動学（復習）

- 順運動学（フォワード・キネマティックス）
  - 多関節体の関節回転から、各部位の位置・向きを計算
  - 回転・移動の変換行列の積により計算
- 逆運動学（インバース・キネマティックス）
  - 指定部位の目標の位置・向きから、多関節体の関節回転の変化を計算
    - 手先などの移動・回転量が与えられた時、それを実現するための関節回転の変化を計算する
  - 姿勢を指定する時、関節回転よりも、手先の位置・向きなどを使った方がやりやすい
  - ロボットアームの軌道計画等にも用いられる



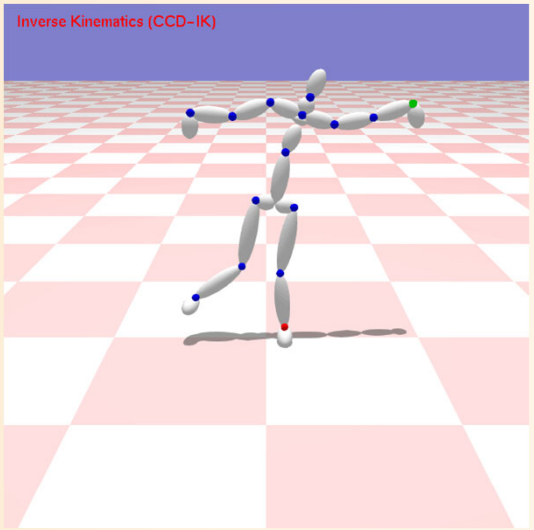
逆運動学

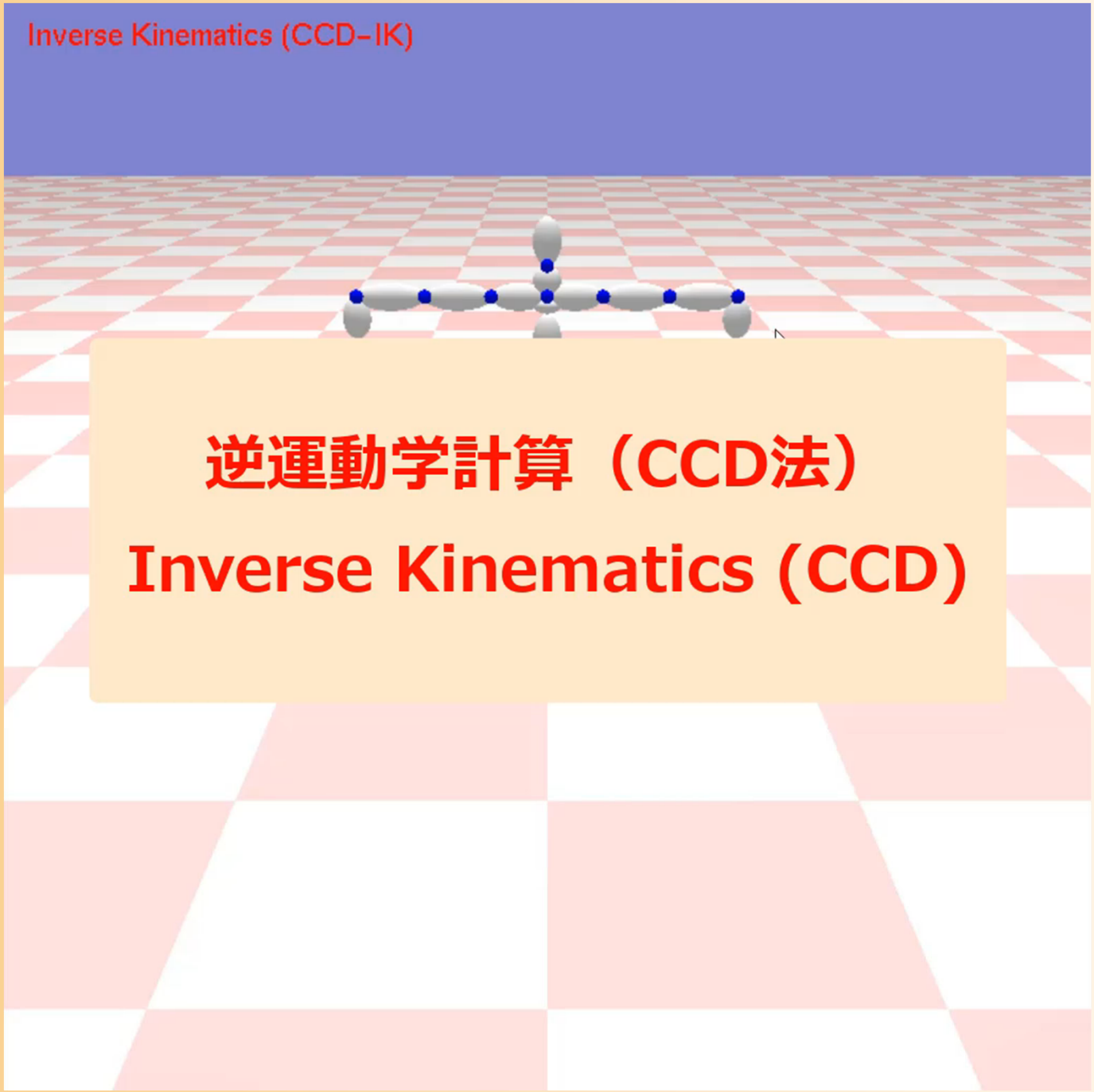
- 逆運動学（インバース・キネマティックス）
  - 入力
    - 現在の姿勢（腰の位置・向き、全関節の回転）
    - 任意の末端部位（エンドエフェクタ）とその目標位置・向き
      - 複数の部位となったり、位置・向きの一部のみを指定する場合がある
    - 支点部位
  - 出力
    - 入力を満たすような姿勢変化を計算



デモプログラム

- 逆運動学計算（CCD法）アプリケーション
  - CCD法による逆運動学計算
  - マウス操作による関節の選択・移動
    - 以前の授業のピッキング技術（スクリーン座標で判定）により末端・支点関節を選択
    - マウスドラッグに応じて、視線に垂直な超平面上で、末端関節の目標位置を移動
      - 本プログラムでは、目標向きは与えず、目標位置のみを与える



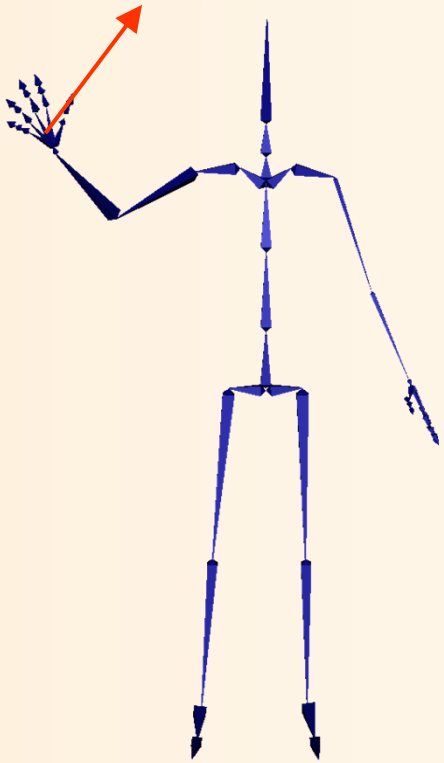

Inverse Kinematics (CCD-IK)
逆運動学計算（CCD法）
Inverse Kinematics (CCD)

逆運動学計算における問題

- 逆運動学での制約条件に関する問題
  - アンダー・コンストレインツ（少な過ぎる制約）
    - ある部位の位置・向きを指定しただけでは、姿勢が一意に決まらない
      - 例：片手の位置が指定されても、その手の位置を実現するような全身の関節回転の組は無数に存在する
    - 何らかの評価基準を入れて適当な解を決定する
      - 例：入力姿勢と出力姿勢の全関節での回転の差がなるべく小さくなる解を選択
  - オーバー・コンストレインツ（多過ぎる制約）
    - 逆に制約が多すぎて、与えられた制約の全てを満足する姿勢がない

逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法

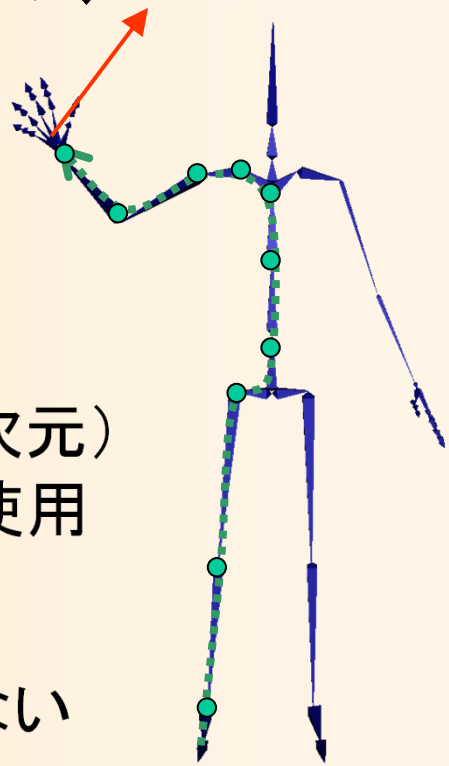


ヤコビ行列による解法(1)

- ヤコビ行列の擬似逆行列による解法
  - 各関節の角度を微小変化させたときの、末端部位の微小変化(J)を計算

    $\mathbf{J}\Delta\boldsymbol{\theta} = \Delta\mathbf{p}$

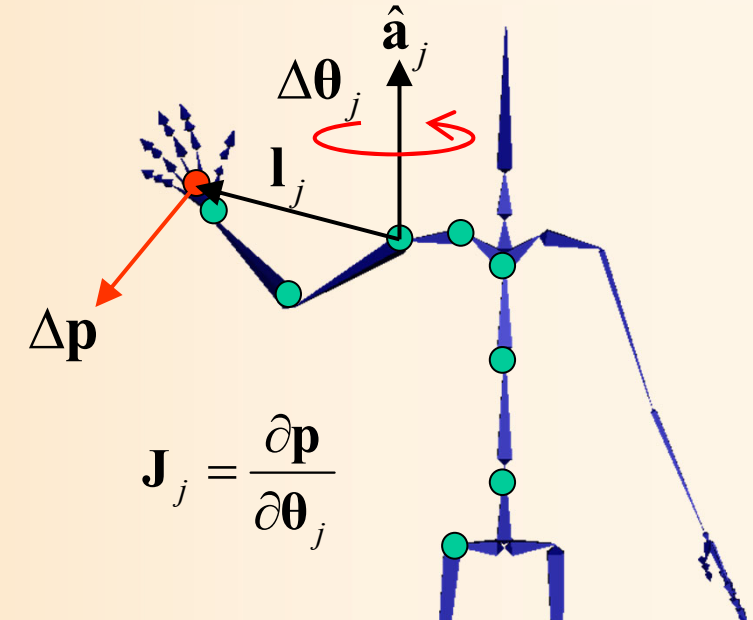
    - $\Delta\boldsymbol{\theta}$ 支点から末端までの全関節の角度変化(n次元)(オイラー角表現が基本)
    - $\Delta\mathbf{p}$ 末端部位の位置・向き変化(3or6 次元)向きにはオイラー角や部位の軸などを使用
  - $\Delta\mathbf{p}$ を満たすような $\Delta\boldsymbol{\theta}$ を計算
    - 通常 n>3or6 なので解は一意に決まらない



ヤコビ行列による解法(2)

- ヤコビ行列の計算
  - 各関節の角度を微小変化させたときの、末端部位の微小変化(J)を計算

$$\begin{pmatrix} & \\ \mathbf{J}_j & \end{pmatrix} \begin{pmatrix} \mathbf{J} \\ \\ \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{\theta} \\ \\ \end{pmatrix} = \begin{pmatrix} \Delta\mathbf{p} \\ \\ \end{pmatrix}$$



関節から末端へのベクトル

$$\mathbf{J}_j = \hat{\mathbf{a}}_j \times \mathbf{l}_j$$

$$\mathbf{J}_j = \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}_j}$$

j番目の関節の回転に応じた末端の変化

回転軸の向きを表す単位ベクトル

※ 全てワールド座標系で計算

ヤコビ行列による解法(3)

- 擬似逆行列による計算
  - **J** は正則ではなので逆行列は計算できない
  - 擬似逆行列を使用($|\Delta\theta|$が最小となる解を計算)

$$\mathbf{J}\Delta\boldsymbol{\theta} = \Delta\mathbf{p}$$

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{+}\Delta\mathbf{p} \qquad \mathbf{J}^{+} = \mathbf{J}^{T}\left(\mathbf{J}\mathbf{J}^{T}\right)^{-1}$$

    - 最小二乗法の一種
    - 逆行列が計算できれば、擬似逆行列も計算可能
  - **J** には、関係のない関節は含まないようにする必要がある(逆行列が計算できなくなる)

ヤコビ行列による解法(4)

- 零空間写像による弱い制約の追加

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{+}\Delta\mathbf{p} + \left(\mathbf{I} - \mathbf{J}^{+}\mathbf{J}\right)\Delta\boldsymbol{\theta}'$$

  - 第2項は末端部位の位置・向きには影響しない
    - 影響しない範囲で、任意の修正($\Delta\boldsymbol{\theta}'$)を適用できる
    - 関節回転可能範囲の制約、重心の位置の制約、等
- 重み付き擬似逆行列による回転割合の制御

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{+}\Delta\mathbf{p} \qquad \mathbf{J}^{+} = \mathbf{W}^{-1}\mathbf{J}^{T}\left(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^{T}\right)^{-1}$$

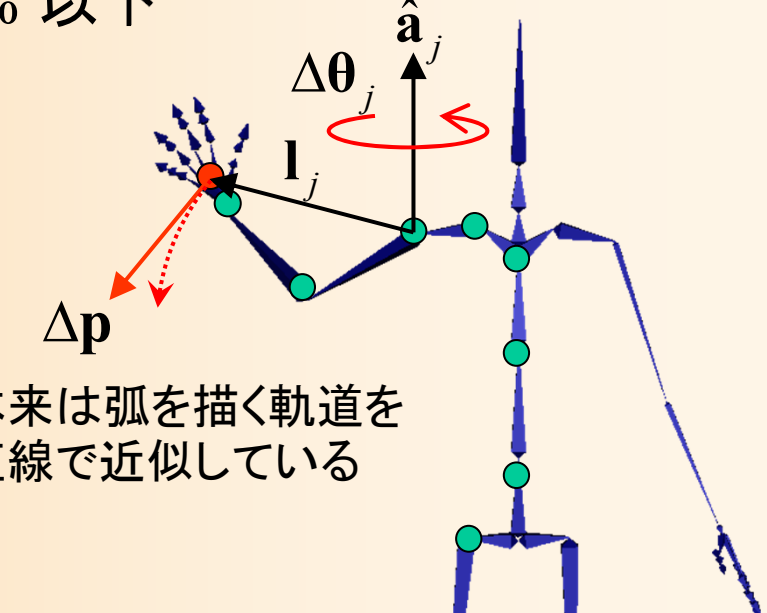
  - $\mathbf{W}$は各関節の重みを表す n×n 対角行列

ヤコビ行列による解法(5)

- ヤコビ行列の疑似逆行列による解法の注意

$$\Delta\boldsymbol{\theta} = \mathbf{J}^{+}\Delta\mathbf{p} + \left(\mathbf{I} - \mathbf{J}^{+}\mathbf{J}\right)\Delta\boldsymbol{\theta}'$$

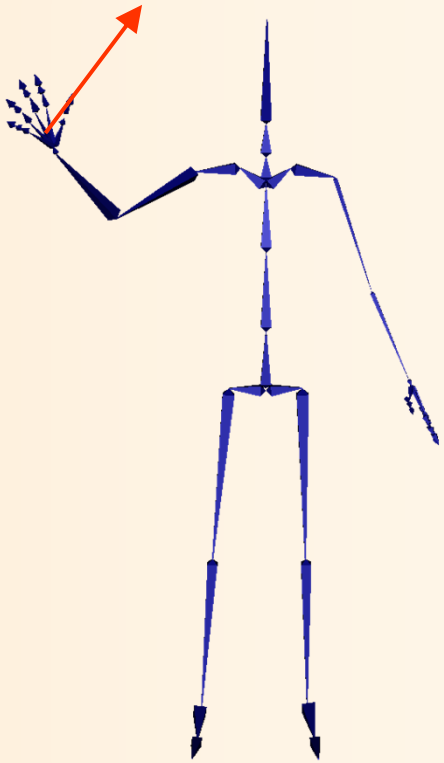
  - $\Delta\mathbf{p}$ の長さを十分小さく取る必要がある
    - リンク全体の長さの 2～10 % 以下
  - 繰り返し計算によって少しずつ姿勢を修正
  - 姿勢が変わる度に $\mathbf{J}$ を計算し直す必要がある



本来は弧を描く軌道を
直線で近似している

逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法



CCD法

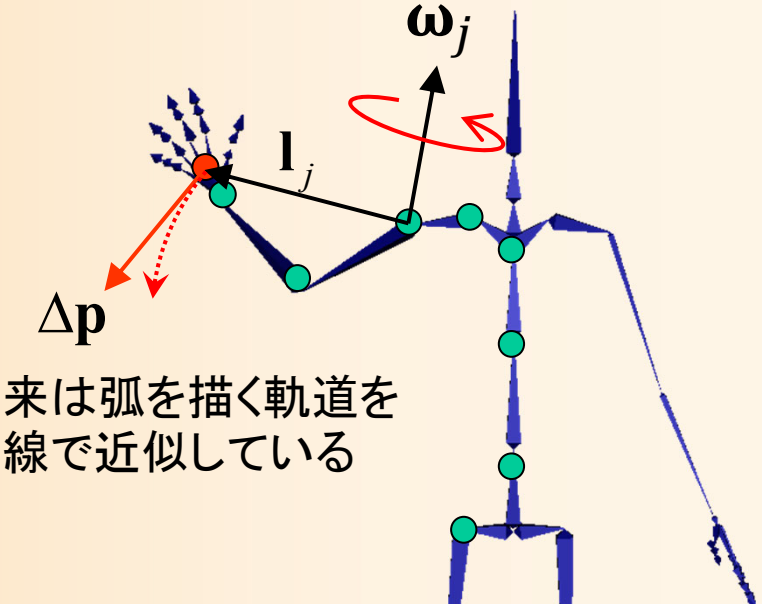
- Cyclic Coordinate Descent（CCD）法
  - ヤコビ行列の擬似逆行列を使う方法と同様の繰り返し計算で姿勢変化を計算
    - ヤコビ行列の擬似逆行列を使う方法より簡単な方法
  - 各関節の回転を独立に計算
    - 末端関節の位置が目標位置に来るように、各関節の回転軸・回転角度を計算

CCD法の概要

- Cyclic Coordinate Descent（CCD）法
  - 各関節の回転を独立に計算
    - 末端関節の位置が目標位置に来るように、各関節の回転軸・回転角度を計算

$$\boldsymbol{\omega}_j = \mathbf{l}_j \times \Delta\mathbf{p}$$

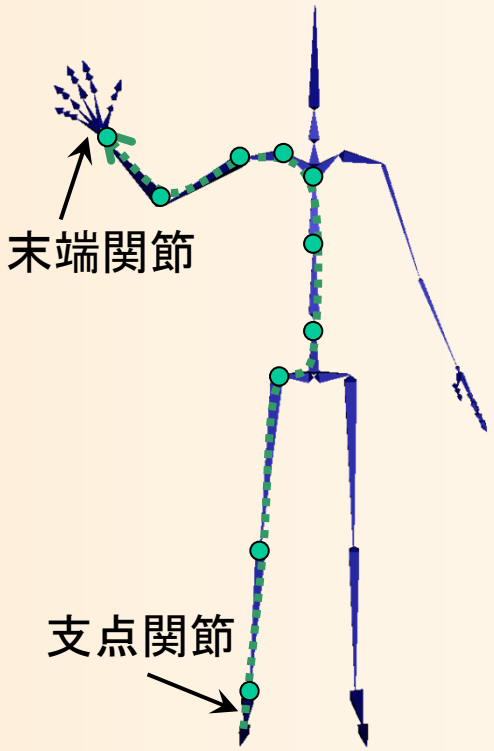
      - 回転軸に回転角度をかけたベクトルの形式で求まる
      - 必要に応じて、回転行列などの別の表現に変換する
    - 必要に応じて、各関節の回転量に重みを適用することも可能



本来は弧を描く軌道を直線で近似している

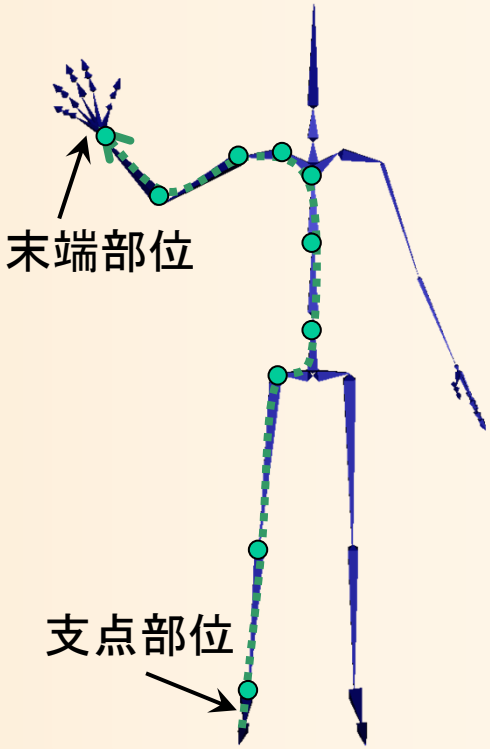
CCD法の計算手順

1. 指定された支点関節・末端関節にもとづき、支点関節から末端関節までの経路を決定
2. 繰り返し計算により、少しずつ姿勢を変化
   - 末端関節から支点関節に向かって順番に処理を適用
     - 末端関節の目標位置を満たすように各関節の回転を計算



CCD法の計算方法(1)

- 末端→支点関節の経路(関節順番)の探索
  - 各体節のどちら側(何番目の関節側)に末端・支点関節があるかが分かれば、経路を決定可能
    - 事前にインデックスを作成しておく
    - あるいは、ルートがどちらにあるかは分かるので、末端・支点関節からそれぞれルートに向かって辿り、合流した体節でパスを結合する
      - 後の説明では、こちらの方法を使用
    - 実装が難しければ、最初は常にルート体節が支点と仮定して作成すると、簡単になる

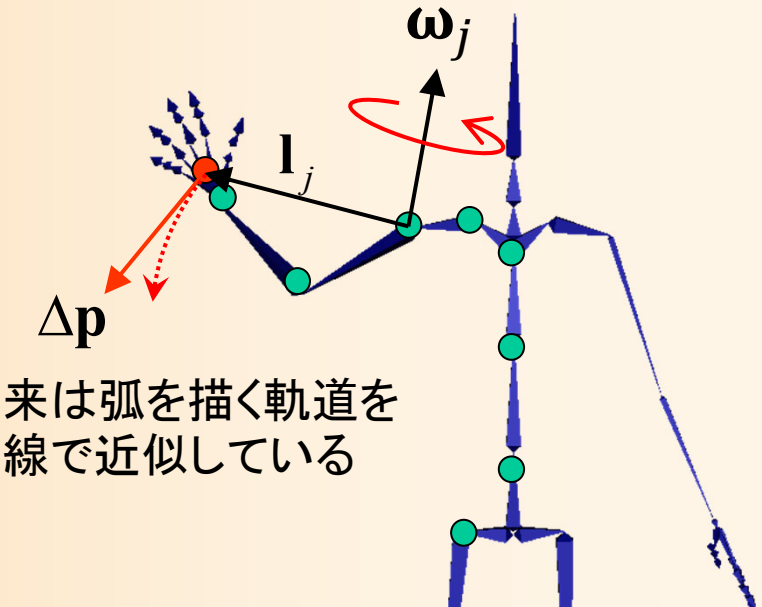


CCD法の計算方法(2)

- 繰り返し計算により、少しずつ姿勢を変化
  - 末端関節から支点関節に向かって順番に処理を適用
    - 各関節のヤコビ行列にもとづき、
      各関節の回転を独立に計算
      - 全関節の回転の和の最小化はできない
    - 目標を満たすための各関節
      の回転軸・回転角度を計算

$$\boldsymbol{\omega}_j = \mathbf{l}_j \times \Delta\mathbf{p}$$

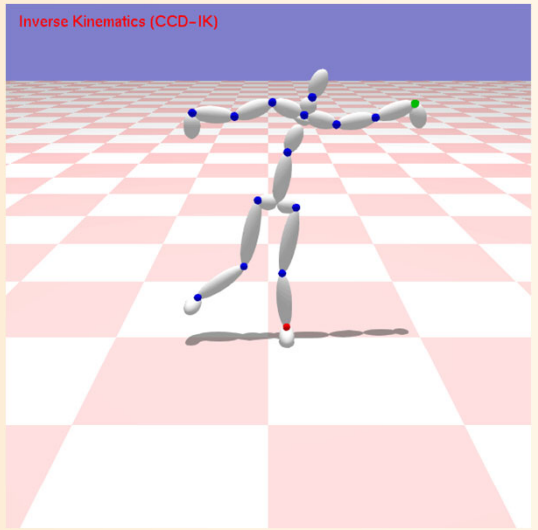
  - 末端関節との間に腰
    (ルート)がある場合は、
    ルートの移動・回転が必要
  - 各関節の回転量に重みを
    かけて調整することも可能



本来は弧を描く軌道を
直線で近似している

プログラミング演習

- 逆運動学計算（CCD法）アプリケーション
  - マウス操作による関節の選択・移動（実装済み）
    - 以前の授業のピッキング技術（スクリーン座標で判定）により末端・支点関節を選択
    - マウスドラッグに応じて、視線に垂直な超平面上で、末端関節の目標位置を移動
      - 本プログラムでは、目標向きは与えず、目標位置のみを与える
  - CCD法による逆運動学計算（各自実装）



逆運動学計算アプリケーション

- InverseKinematicsCCDApp（一部未実装）
  - CCD法による逆運動学計算のアプリケーション
  - マウス操作による関節の選択・移動（実装済み）
    - 以前の授業のピッキング技術（スクリーン座標で判定）により末端・支点関節を選択
    - マウスドラッグに応じて、視線に垂直な超平面上で、末端関節の目標位置を移動
  - CCD法による逆運動学計算（各自実装）
    - ルート体節を支点とする → 任意の関節を支点とする
    - FindJointPath関数の一部を作成
    - ApplyInverseKinematicsCCD関数の一部を作成

CCD法のプログラミング(1)

- CCD法による逆運動学計算
  - 入力:現在姿勢、支点関節番号、末端関節番号、末端関節の目標位置
  - 出力:逆運動学計算後の姿勢
    - 目標の条件として、1つの末端関節とその位置を指定
      - ルート体節を支点とする場合は、支点関節番号に −1 を指定
    - 現在姿勢の入力と計算後の姿勢の出力に、同じ変数を使用


```
// 逆運動学計算(CCD法)
void  ApplyInverseKinematicsCCD( Posture & posture,
  int base_joint_no, int ee_joint_no, Point3f ee_joint_position );
```

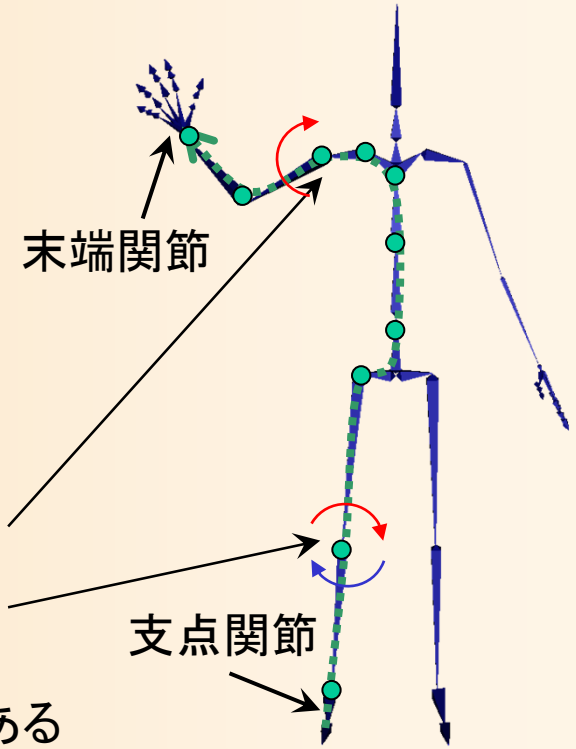
CCD法のプログラミング(2)

- 支点関節から末端関節への経路の探索
  - 入力:骨格情報、支点関節番号、末端関節番号
  - 出力:経路に含まれる関節番号＋方向のリスト
    - 出力には可変長配列を使用
    - 経路に含まれる関節番号＋方向(1 or －1)のリスト
      - ルートよりも支点側(1)にあるか末端側(－1)にあるかで、関節周りの回転の方向が逆転するため、方向も合わせて出力


```
// 末端関節から支点関節へのパスを探索
void  FindJointPath( const Skeleton * body,
    int base_joint_no, int ee_joint_no,
    vector< int > & joint_path, vector< int > & joint_path_signs );
```

CCD法のプログラミング(3)

- 支点関節から末端関節への経路の探索
  - 入力:骨格情報、支点関節番号、末端関節番号
  - 出力:経路に含まれる関節番号＋方向のリスト
    - 関節番号＋方向(1 or －1)
      - ルートよりも支点側(1)にあるか
        末端側(－1)にあるかで、
        関節周りの回転の方向が逆転する
        ため、方向も合わせて出力



末端側:関節回転により末端側が回転

支点側:関節回転により支点側が回転
関節の回転を表すときには、
回転方向を反対にする必要がある

CCD法のプログラミング（4）

```
// 末端関節から支点関節への経路（パス）を探索
void  FindJointPath( const Skeleton * body,
    int base_joint_no, int ee_joint_no,
    vector< int > & joint_path, vector< int > & joint_path_signs )
{
    // 末端関節から探索を開始

    // 末端関節からルートに向かうパスを探索
    while（・・・）
        // ルート体節か支点関節に到達したら終了
        // 現在の関節をパスの配列に追加、関節の符号は全て 1 とする

    // 支点関節に到達したら終了（支点がルート体節 or 現在の関節）

    // 支点関節からルートに向かうパスを探索（上と同様の処理）

    // 2つのパスを統合
}
```

CCD法のプログラミング(4)

```
// 末端関節から支点関節への経路(パス)を探索
void  FindJointPath( const Skeleton * body,
    int base_joint_no, int ee_joint_no,
    vector< int > & joint_path, vector< int > & joint_path_signs )
{
    // 末端関節から探索を開始

    // 末端関節からルートに向かうパスを探索
    while(・・・)
        // ルート体節か支点関節に到達したら終了
        // 現在の関節をパスの配列に追加、関節の符号は全て 1 とする

    // 支点関節
    // 支点関節
    // 2つのパ
}
```

最初はルート体節が支点であると仮定して、
ここまでの処理を作成すると良い
うまくいったら、任意の支点関節が指定されて
も対応できるように、残りの処理も作成する

CCD法のプログラミング（5）

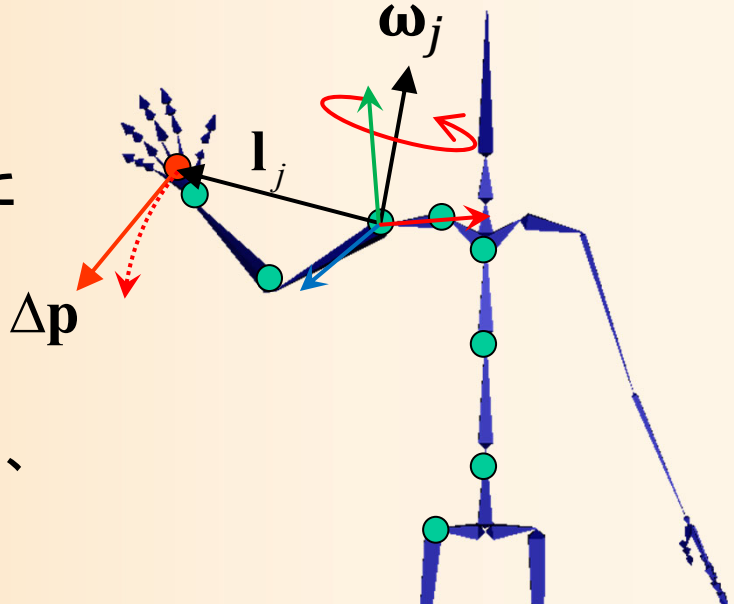
- 繰り返し計算により、少しずつ姿勢を変化
  - 末端関節から支点関節に向かって順番に処理を適用
    - 目標を満たすための各関節の回転軸・回転角度を計算

$$\boldsymbol{\omega}_j = \mathbf{l}_j \times \Delta\mathbf{p}$$

      - 関節のローカル座標系で計算（ルート側の体節の向き）
    - 現在の関節の回転に、求めた回転を適用して、姿勢を更新

$$\mathbf{R}_j' = \mathbf{R}(\boldsymbol{\omega}_j)\,\mathbf{R}_j$$

      - ルートよりも支点側の関節では、順番は逆になる



CCD法のプログラミング（6）

```
// 逆運動学計算（CCD法）
void  ApplyInverseKinematicsCCD( Posture & posture,
  int base_joint_no, int ee_joint_no, Point3f ee_joint_position )
{
    // 末端関節から支点関節へのパスを探索
    FindJointPath( ・・・);

    // CCD法の繰り返し計算（収束するか、一定回数繰り返したら終了）
    for ( int i = 0; i < max_iteration; i++ )
    {
        // 末端関節から支点関節に向かって繰り返し
        for ( int j = 0; j < joint_path.size(); j++ )
        {
            // 末端関節を目標位置に近づける、現在の関節の回転を計算
            // 回転を適用する際には、回転の方向を考慮

            // 順運動学計算を再計算
```

CCD法のプログラミング（6）

```
// 逆運動学計
void ApplyInv
  int base_join
{
    // 末端関節
    FindJointPa

    // CCD法の
    for ( int i =
    {
        // 末端関
        for ( int j = 0; j <                    .size(); j++ )
        {
            // 末端関節を目標位置に近づける、現在の関節の回転を計算
            // 回転を適用する際には、回転の方向を考慮

            // 順運動学計算を再計算
```

前スライドの説明通りの処理を行う

1. 順運動学計算結果より、末端体節の位置、現在の関節の位置・向き（いずれもワールド座標系）を取得
2. ベクトル $\mathbf{l}_j$、$\Delta\mathbf{p}$ を計算（関節のローカル座標系）
3. 関節の回転（ベクトル表現）$\boldsymbol{\omega}_j$ を計算
4. 関節の回転をベクトル表現から行列表現に変換
   vecmath の AxisAngle4f 型を経由して変換すると、簡単に変換できる
5. 関節の現在の回転 $\mathbf{R}_j$ に、回転を適用し、新しい回転 $\mathbf{R}_j'$ を求める

CCD法のプログラミング（6）

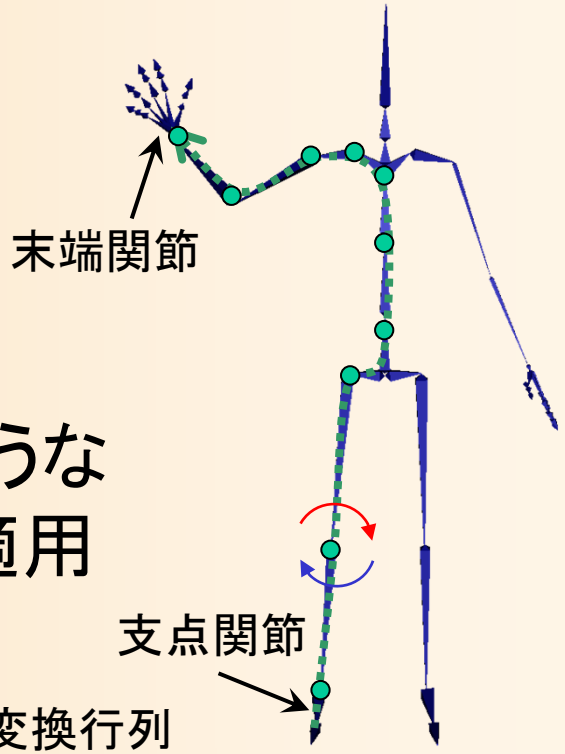
```
// 逆運動学計算（CCD法）
void  ApplyInverseKinematicsCCD( Posture & posture,
  int base_joint_no, int ee_joint_no, Point3f ee_joint_position )
{
    // 末端関節から支点関
    FindJointPath( ・・・);

    // CCD法の繰り返し計
    for ( int i=0; i<max_it
    {
        // 末端関節から支点関                かって繰り返し
        for ( int j=0; j<joint                n.size(); j++ )
        {
            // 末端関節と目標位置に近づける、現在の関節の回転を計算
            // 回転を適用する

            // 順運動学計算
```

最初はルート体節が支点であると仮定して、ここまでの処理を作成すると良い
うまくいったら、任意の支点関節が指定されても対応できるように、残りの処理も作成する

末端関節と現在の関節の間にルート体節がある場合は、ルート体節に移動・回転を適用

CCD法のプログラミング（7）

- ルートよりも支点側にある関節の回転
  - 前のスライドと同じ方法で関節回転を計算
    - 右図の赤矢印の回転が求まる
  - 逆の回転を適用
    - 右図の青矢印の回転を適用
  - そのままでは、ルートが固定されて、支点関節が移動する
  - 支点関節の位置・向きを保つような回転・移動を求めて、ルートに適用



$$\mathbf{M} = \mathbf{M}_{base}\ \mathbf{M}_{base}{}'^{-1}$$

関節回転の適用前・後の支点関節の変換行列

CCD法のプログラミング（8）

```
// 逆運動学計算（CCD法）
void  ApplyInverseKinematicsCCD( Posture & posture,
  int base_joint_no, int ee_joint_no, Point3f ee_joint_position )
{
    ...
    // CCD法の繰り返し計算（収束するか、一定回数繰り返したら終了）
    for ( int i=0; i<max_iteration; i++ )
    {
        // 末端関節から支点関節に向かって繰り返し
        for ( int j=0; j<joint_path.size(); j++ )
        {
            ...
        }

        // 収束判定
    }
```

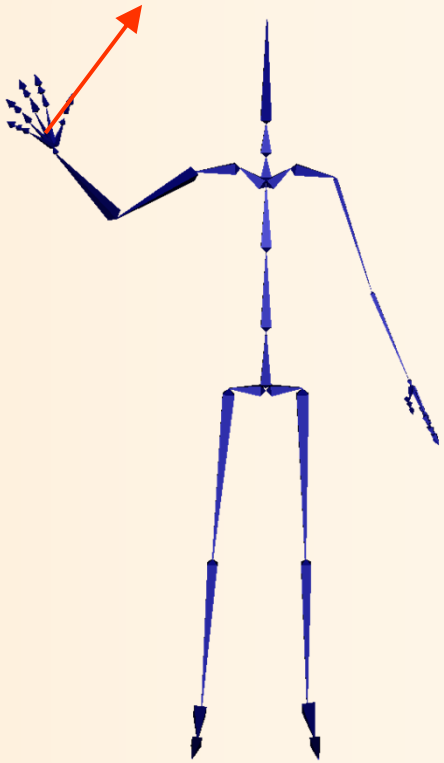

末端関節の目標位置と現在位置の距離が
閾値以下であれば、繰り返しを終了

今回の内容

- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ

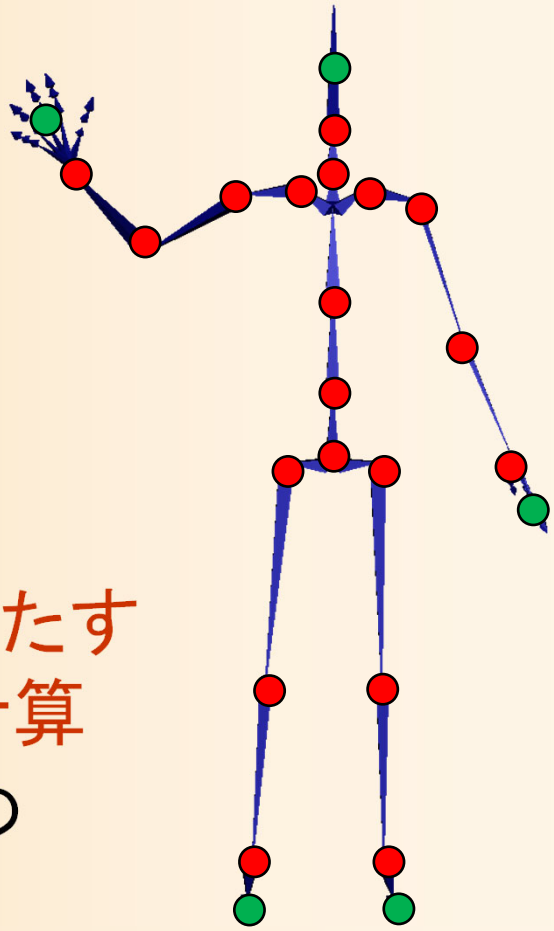
逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法



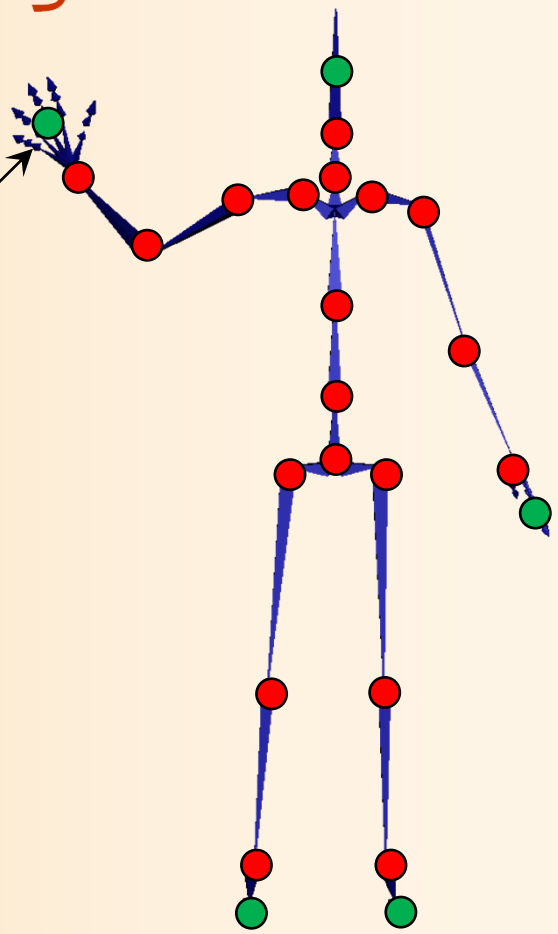
粒子法

- 関節点を粒子とみなして、粒子の位置を計算
  - 例：粒子の物理シミュレーションの手法により、末端関節の目標位置や粒子間（関節間）の距離の条件を満たすような粒子の位置を計算
- 関節点（粒子）の位置の制約を満たすように、姿勢（全関節の回転）を計算
  - 制約の自由度が低いため、軸周りの回転までは決まらない



粒子法での姿勢の計算(1)

- 関節点(＋末端体節点)の位置から姿勢を計算
  - 末端関節については、次の関節が存在しないため、関節の回転を求めるために末端体節の位置も必要



- 計算手順
  1. ルート体節の位置・向きを求める
  2. ルート体節から末端体節に向かって、順番に各関節の回転を求める

粒子法での姿勢の計算(2)

- 関節点(＋末端体節点)の位置から姿勢を計算する関数を定義
  - 全関節と全体節(末端体節の位置のみを使用)の位置の配列を入力

```
// 関節点(＋末端体節点)からの姿勢の計算
void  ComputePostureFromJointPoints(
    const Point3f * joint_positions, const Point3f * segment_positions,
    Posture & pose );
{
    // 腰の位置・向きを計算
    // 末端関節に向かって繰り返し(再帰呼び出し)
        // 次の関節点 or 末端関節点から、i番目の関節の回転を計算
        // 次の関節点が1つと3つで場合分けをして計算
}
```

粒子法での姿勢の計算(3)

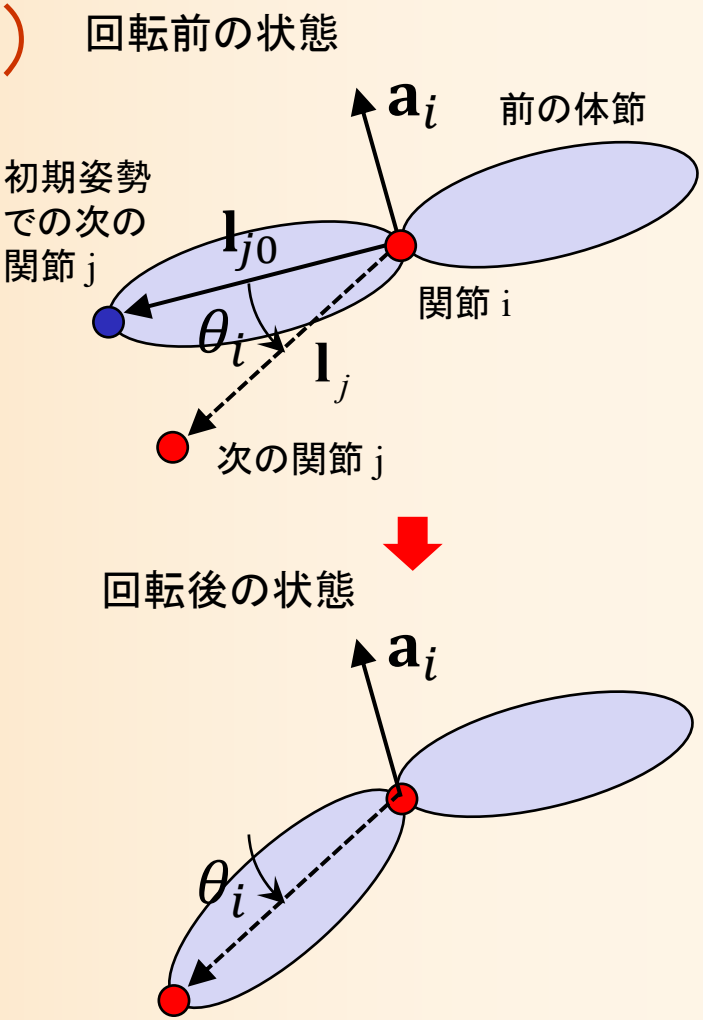
- 各関節での回転の計算(1)
  - 次の関節が1つの場合
    - 次の関節点の位置を満たす、初期姿勢からの関節回転を計算
      - 関節の回転(軸・角度)の計算

$$\mathbf{a}_i = \frac{\mathbf{l}_{j0} \times \mathbf{l}_j}{|\mathbf{l}_{j0} \times \mathbf{l}_j|} \qquad \theta_i = \cos^{-1}\left(\frac{\mathbf{l}_{j0} \cdot \mathbf{l}_j}{|\mathbf{l}_{j0}||\mathbf{l}_j|}\right)$$

      - 関節の回転行列の計算
        (回転ベクトルから変換)

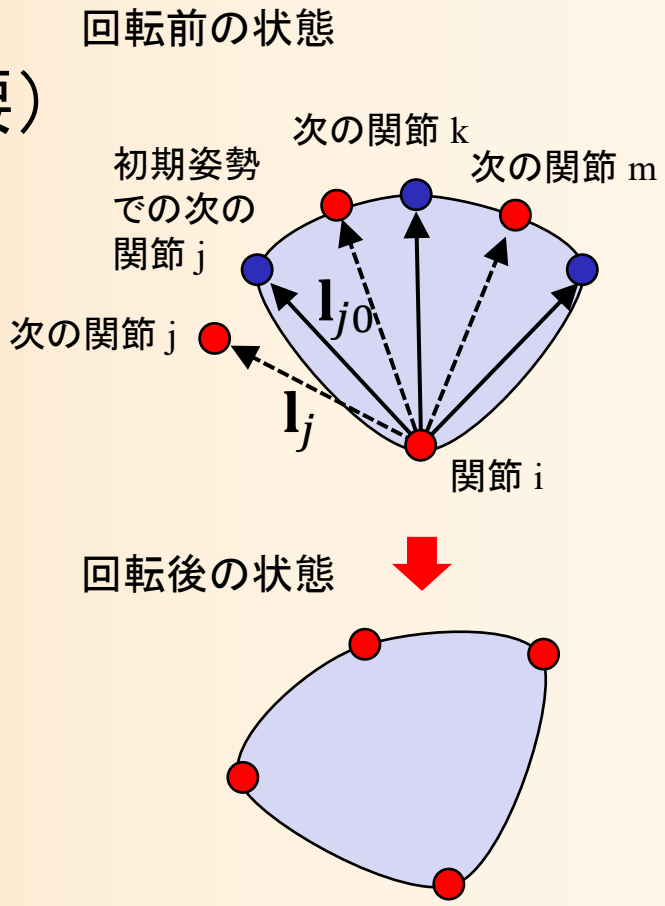
$$\mathbf{R}_i = \mathbf{R}(\mathbf{a}_i, \theta_i)$$

      - 2自由度の回転を計算



粒子法での姿勢の計算(4)

- 各関節での回転の計算(2)
  - 次の関節が3つの場合(胸・腰)
    - 3つの関節点の条件を満たす回転行列を計算



$$\mathbf{l}_j = \mathbf{R}_i \ \mathbf{l}_{j0}$$

$$\mathbf{l}_k = \mathbf{R}_i \ \mathbf{l}_{k0}$$

$$\mathbf{l}_m = \mathbf{R}_i \ \mathbf{l}_{m0}$$

$$\underset{3\times3\text{行列}}{\left(\mathbf{l}_j \ \mathbf{l}_k \ \mathbf{l}_m\right)} = \mathbf{R}_i \underset{3\times3\text{行列}}{\left(\mathbf{l}_{j0} \ \mathbf{l}_{k0} \ \mathbf{l}_{m0}\right)}$$

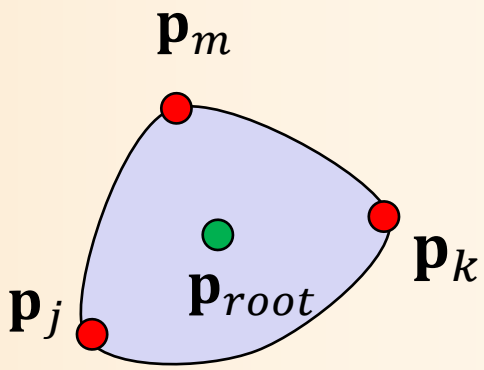
$$\mathbf{R}_i = \left(\mathbf{l}_j \ \mathbf{l}_k \ \mathbf{l}_m\right)\left(\mathbf{l}_{j0} \ \mathbf{l}_{k0} \ \mathbf{l}_{m0}\right)^{-1}$$

※ 正規化が必要

粒子法での姿勢の計算(5)

- 各関節での回転の計算(3)
  - 次の関節が3つの場合(腰)
    - 腰の場合は、3つの関節点の位置から、腰の位置を計算
      - 3つの位置を平均

$$\mathbf{p}_{root} = \frac{1}{3}(\mathbf{p}_j + \mathbf{p}_k + \mathbf{p}_m)$$

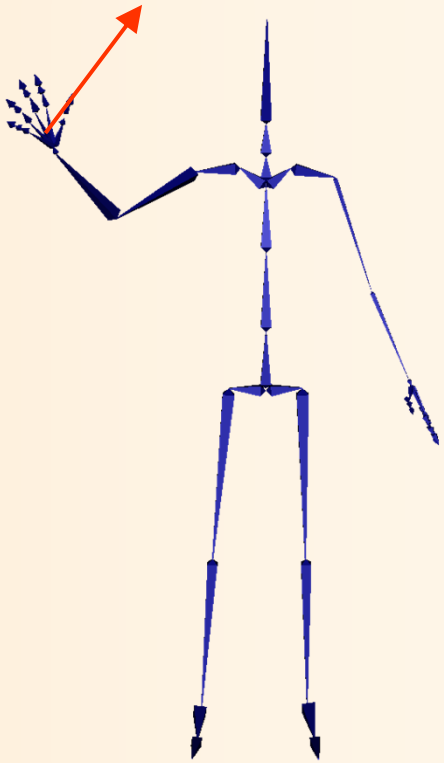


    - その後、前スライドの計算方法で、腰の向きを計算

- 以上の方法で、関節点(+末端体節点)の位置から、一意に姿勢を計算

逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法

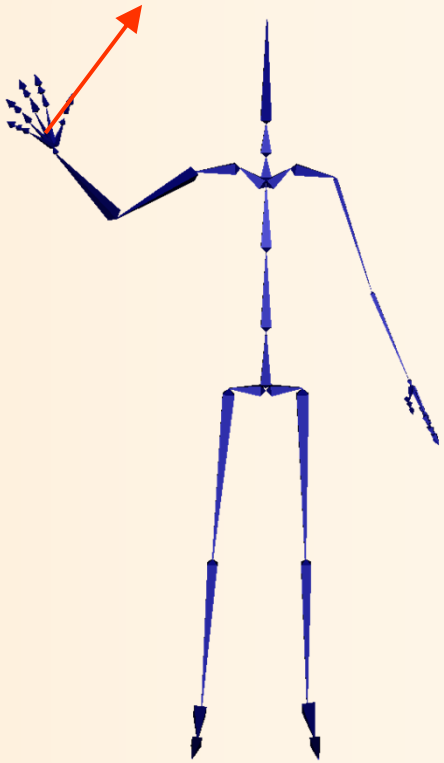


非線形最適化問題による解法

- 最適化問題と考えて、指定部位の位置・向きの条件を満たすような姿勢(関節角度の組み合わせ)を探索
- ヤコビ行列の擬似逆行列による解法(線形問題)を、より一般化した解法
- 目標関数の定義や最適化問題の解法にさまざまな方法がある
  - 手法によっては計算時間がかかる

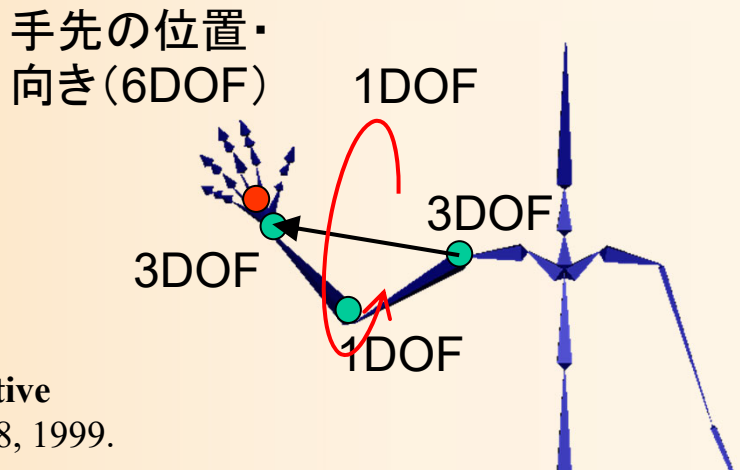
逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法



解析的解法(1)

- 人間の骨格に基づく効率的な計算方法
- 手足の姿勢の解析的な逆運動学計算
  - 肩 or 股関節を支点として、手 or 足の位置・向きを満たす、腕 or 脚の姿勢を計算
  - 手先・足先の位置・向きは6自由度
  - 腕・脚は全体で7自由度
    - 残りの自由度は1(ひじ・ひざの旋回角度)のみ
      - 何らかの方法でこれを決定すれば解は一意に決まる

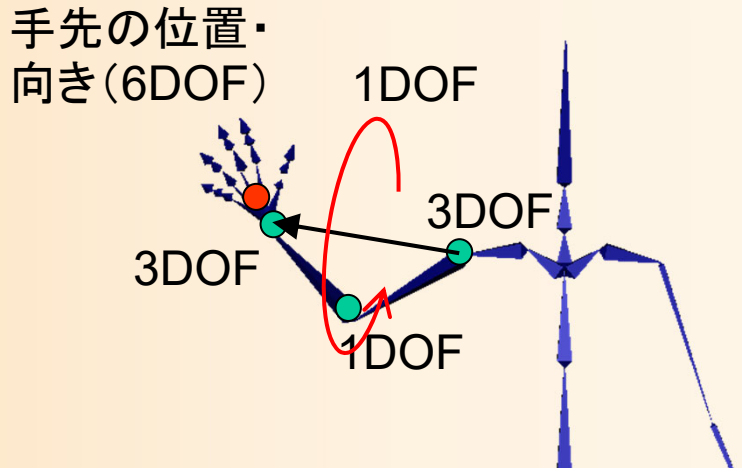


Jehee Lee, Sung Young Shin, **A Hierarchical Approach to Interactive Motion Editing for Human-like Figures**, SIGGRAPH 99, pp.39-48, 1999.

解析的解法(2)

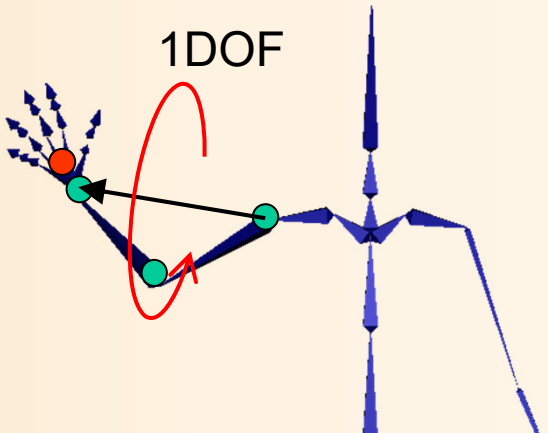
- 入力:手先の目標位置・向き、肩(胴体)の位置・向き、腕の2本の体節の長さ

1. 目標位置と肩の距離から、ひじの屈伸回転を計算(1DOF)
2. 手先が目標位置に来るように、肩の回転を計算(2DOF)
3. 何らかの方法で、ひじの旋回回転を決定(1DOF)
4. 手先の目標向きに応じて、手首の回転を計算(3DOF)



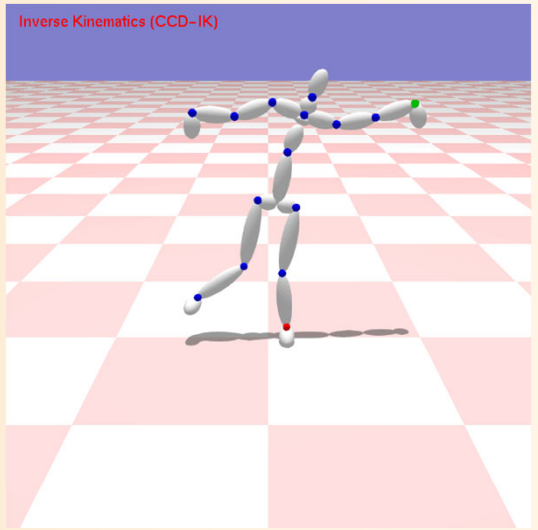
解析的解法(3)

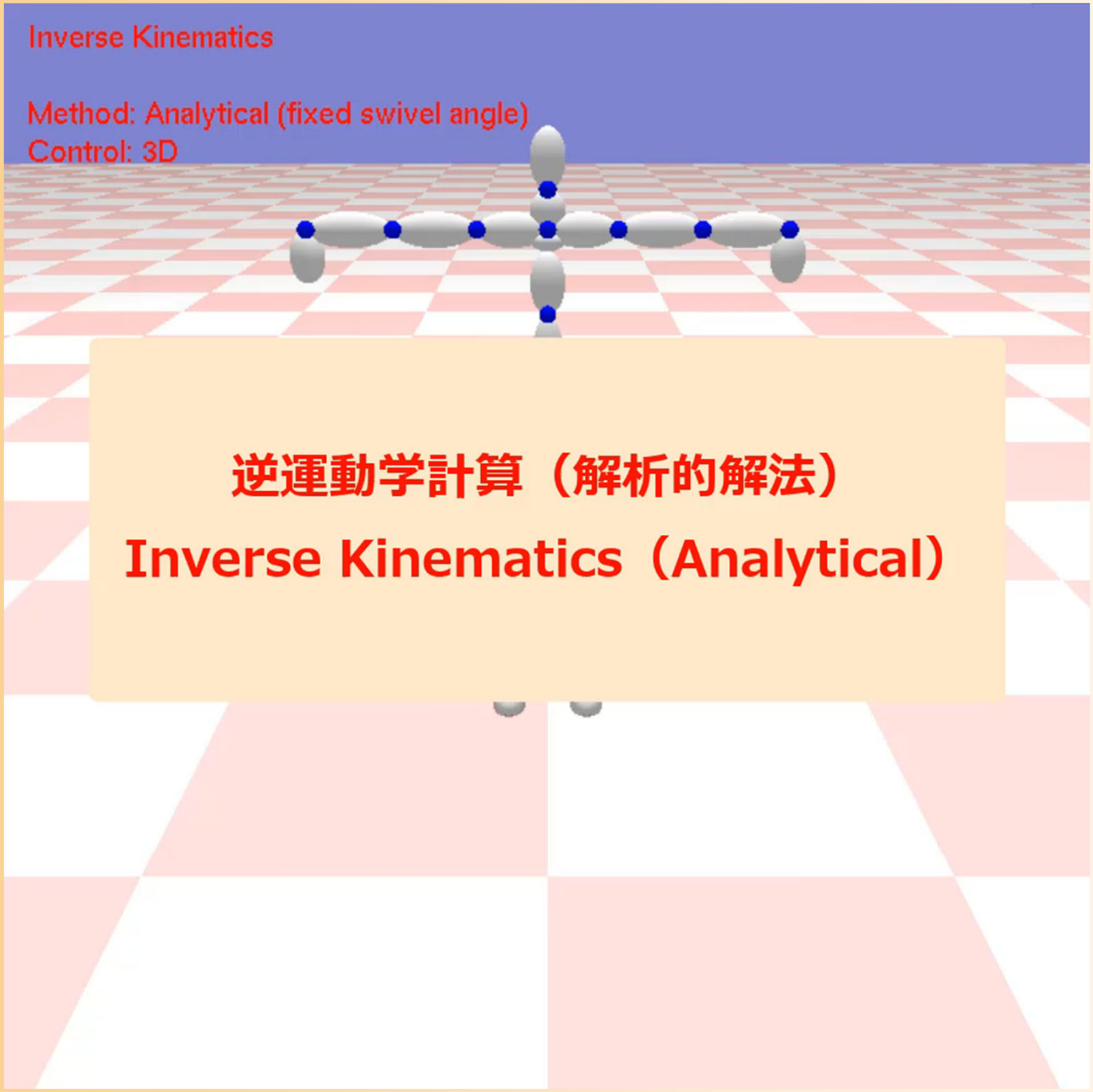
- ひじ(ひざ)の旋回角度の計算方法の例
  - 現在の姿勢の旋回角度を保持
    - 姿勢を少しだけ変更するような場合であれば、有効
  - 標準的な旋回角度を適用
    - どのような目標位置・向きのときに、どのような旋回角度になるかのデータを用意しておき、そのデータにもとづいて決定



デモプログラム

- 逆運動学計算(解析的解法)アプリケーション
  - 解析的解法による逆運動学計算
  - マウス操作による関節の選択・移動・回転
    - 以前の授業のキーフレームアニメーションのサンプルプログラムで利用したもの同様のインターフェースを使って、末端部位の位置・向きを操作
    - 末端部位(手先・足先)しか操作できない



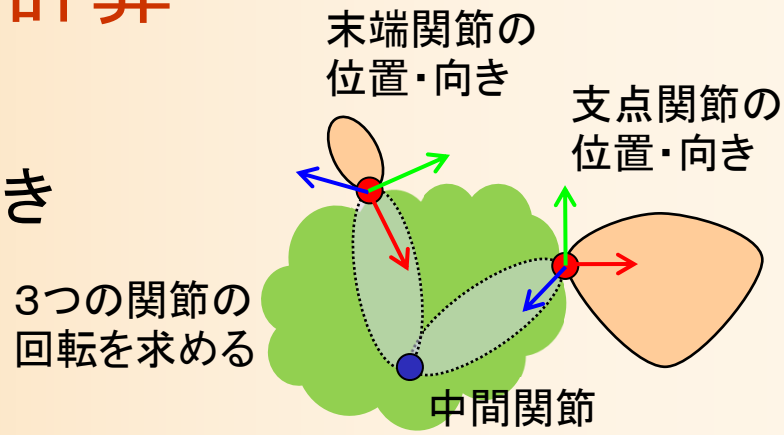


逆運動学計算（解析的解法）

Inverse Kinematics（Analytical）

解析的解法の計算方法(1)

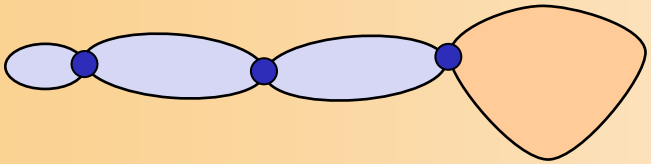
- 一つの手 or 足の姿勢を計算
  - 入力
    - 支点・末端関節の位置・向き
    - 2本の体節の長さ
  - 出力
    - 支点・中間・末端関節の回転



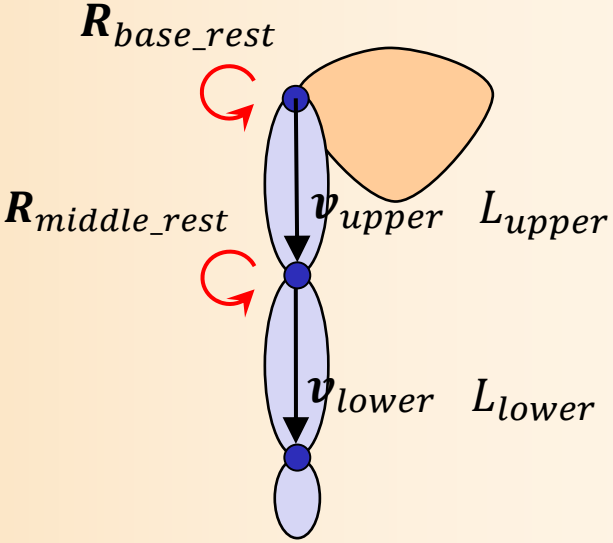

```
// 逆運動学計算(解析的手法)
// 入出力姿勢、末端関節番号(手先・足先に限定)、末端関節の目標位置・向きを入力
// 支点関節の位置・向きは入力姿勢から求まる、体節の長さは骨格から求まる
void  ApplyInverseKinematicsAnalytical( Posture & posture,
    int ee_joint_no, Point3f ee_joint_position, Matrix3f ee_joint_orientation )
{
    // 支点・中間・末端関節の回転を求めて入力姿勢に適用
}
```

解析的解法の計算方法(2)

- 手 or 足を真下に伸ばした状態を基準状態とする
  - 真下に伸ばした姿勢になるように関節の回転を適用（後の処理で使用）
  - 上部・下部の体節のベクトル・長さも後の処理で使用



初期姿勢



手を真下に伸ばした姿勢

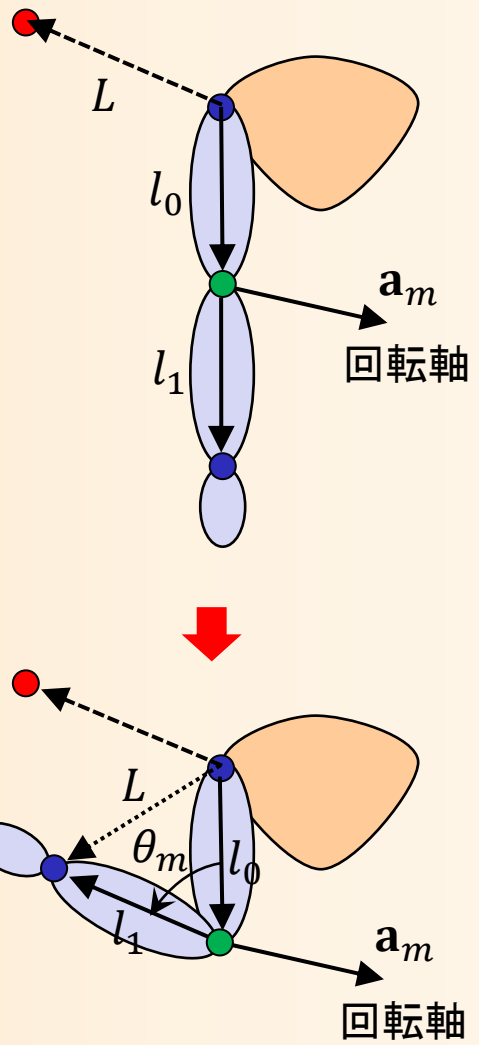
解析的解法の計算方法(3)

1. 中間関節の屈伸回転を計算
   - １自由度の回転角度を計算
     - 支点・末端関節間の目標距離(L)と手の長さ(l_0,l_1)から計算

$$\theta_m = cos^{-1}\left(\frac{l_0{}^2 + l_1{}^2 - L^2}{2\, l_0\, l_1}\right)$$

     - L が範囲外の場合の対応も必要
   - 固定の回転軸周りに回転

$$\mathbf{R}_m{}' = \mathbf{R}_m \ \mathbf{R}(\mathbf{a}_m, \theta_m)$$

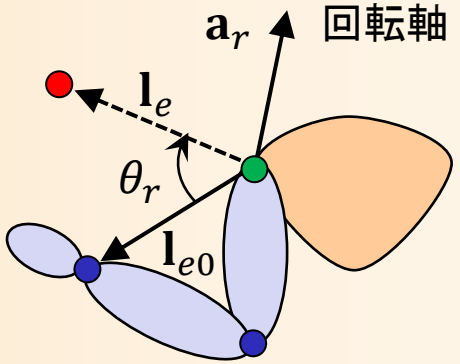


解析的解法の計算方法(4)

2. 支点関節の回転を計算

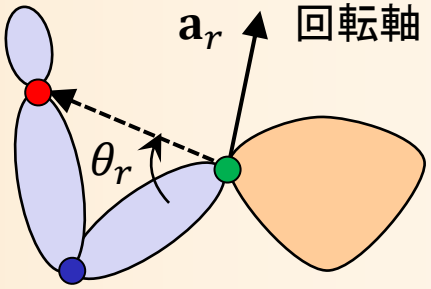
- 2自由度の回転を計算
- 次の関節点の位置を満たす、初期姿勢からの関節回転を計算
  - 関節の回転(軸・角度)の計算

$$\mathbf{a}_r = \frac{\mathbf{l}_{e0} \times \mathbf{l}_e}{|\mathbf{l}_{e0} \times \mathbf{l}_e|} \quad \theta_r = \cos^{-1}\left(\frac{\mathbf{l}_{e0} \cdot \mathbf{l}_e}{|\mathbf{l}_{e0}||\mathbf{l}_e|}\right)$$



  - 関節の回転行列の計算
    - 初期状態の回転 $\mathbf{R}_r$ に、求めた回転軸・角度による回転行列をかける

$$\mathbf{R}_r{}' = \mathbf{R}_r \ \mathbf{R}(\mathbf{a}_r, \theta_r)$$

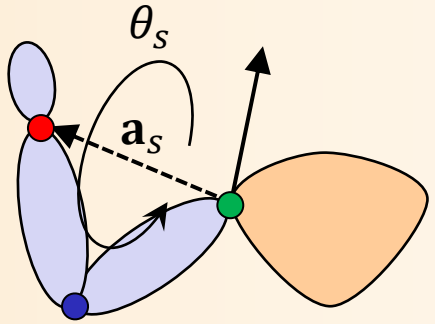


解析的解法の計算方法(5)

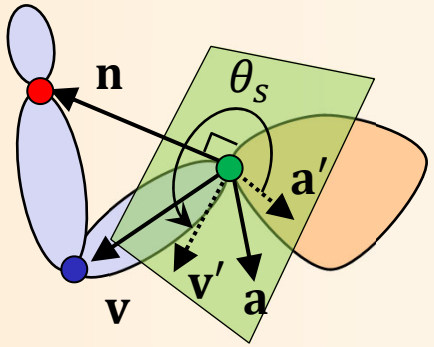
3. 支点関節の回転を計算

- 1自由度の旋回角度 θ_s を計算して、支点関節の回転に適用

$$\mathbf{R}_r{}' = \mathbf{R}_r \ \mathbf{R}(\mathbf{a}_s, \theta_s)$$



- 前のスライドの説明通り、いくつかの方法が考えられる
  - 回転を適用しない
    - 不自然な姿勢になる可能性はあるが、目標位置・向きの条件は満たす
  - 現在の姿勢の旋回角度を保持
  - 標準的な旋回角度を適用
    - 例：足は正面、手は真後ろ
  - 目標位置に応じた旋回角度を適用



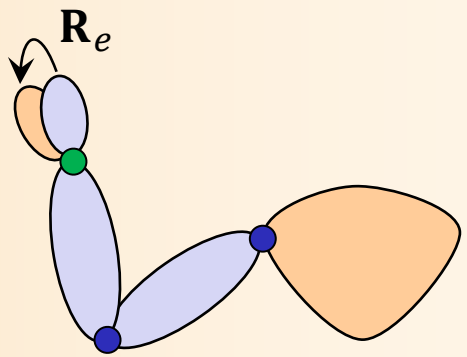
旋回角度の計算方法の例

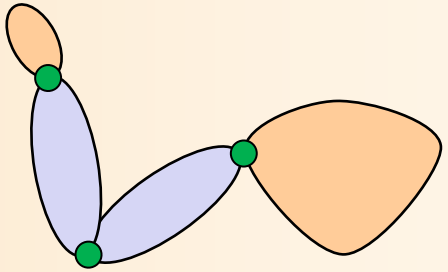
解析的解法の計算方法(6)

4. 末端関節の回転を計算
   - 3自由度の回転を計算
     - 末端関節の目標向き $\mathbf{O}_e$ と
       支点・中間点関節の回転 $\mathbf{R}_r, \mathbf{R}_m$
       から 、末端関節の回転 $\mathbf{R}_e$ を計算

$$\mathbf{O}_e = \mathbf{O}_r \, \mathbf{R}_r \, \mathbf{R}_m \, \mathbf{R}_e$$

$$\mathbf{R}_e = (\mathbf{O}_r \, \mathbf{R}_r \, \mathbf{R}_m)^{-1} \, \mathbf{O}_e$$





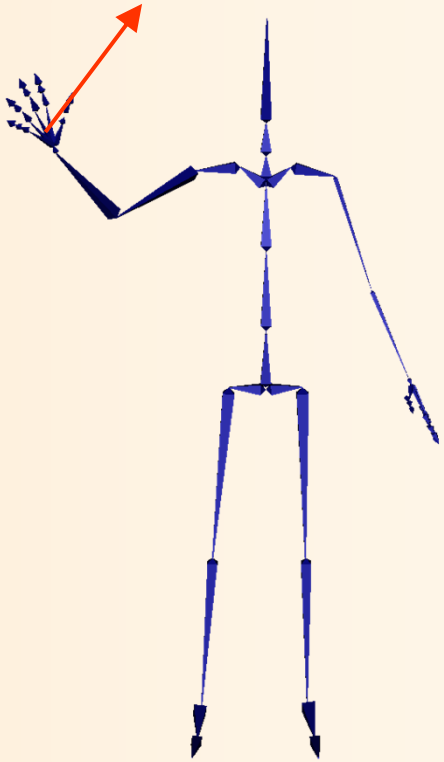
- 以上の手順により、
  支点・中間・末端関節の
  回転が求まる

解析的解法の利用

- 解析的解法は広く用いられている
  - 繰り返し計算が不要であるため、一定の速度で高速に計算できる
    - オンラインアプリケーションでの利用には速度が重要
  - 前の姿勢に依存せず、目標から一意に姿勢が決まる
  - 人体モデルに特化した手法
    - 手足以外の関節の回転についても、何らかのルールに従って変化させることで、全身の姿勢変形に利用
    - アニメーション制作ソフトウェアにも組み込まれている
      - Autodesk Human IK
  - 基本的には大きな姿勢変形には向かない
    - 手足の位置を微調整するような姿勢変形に有効

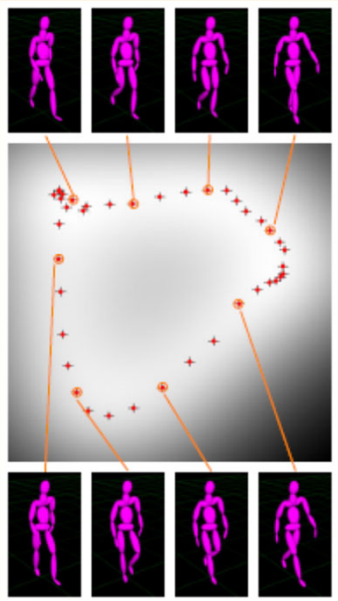
逆運動学(IK)の計算方法

- いくつかの計算方法がある
  - 数値的解法(Numerical)
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent(CCD)法
    - 粒子法(Particle-IK)
    - 非線形最適化問題による解法
  - 解析的解法(Analytical)
  - サンプルデータを用いる解法



サンプルデータを用いる解法

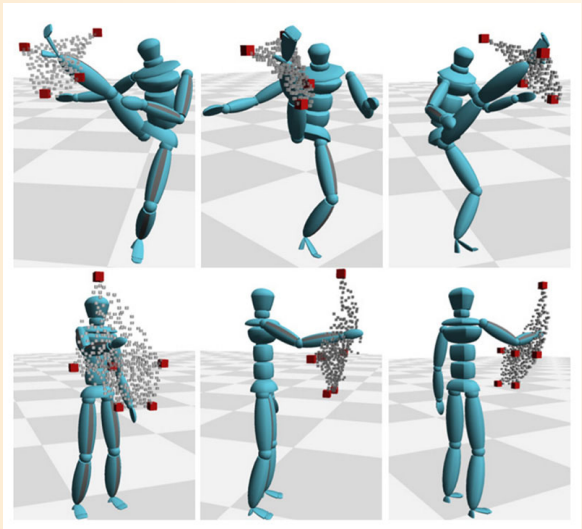
- Style-based IK [Grochow et al. 2004]
  - あらかじめ多数の姿勢データを学習しておき、制約条件が与えられると、制約条件を満たしつつ、学習データに近いような、適切な姿勢を生成
    - 潜在空間（低次元空間）に学習姿勢を写像
    - 潜在空間の中で姿勢を探索・合成
  - サンプルデータを用いることで、逆運動学計算の冗長性の問題を解決
  - 一つの学習モデルで複数の動作は扱えない
    - 同じ制約条件でも動作によって適切な姿勢は異なる



Keith Grochow et al., **Style-based inverse kinematics**, ACM Transactions on Graphics, 23, 3, 2004

サンプルデータを用いる解法

- 姿勢補間を用いる方法もある
  - 少ないパラメタ（決められた動作や末端部位）に対して、十分多数のサンプル姿勢が用意できる場合
  - 目標位置・向きを指定できる部位が、あらかじめパラメタ化されたものに限られる
  - 複数の姿勢の補間が必要



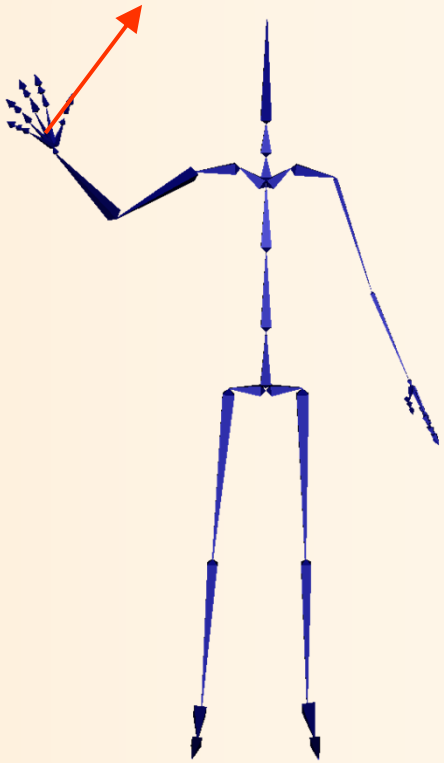
Lucas Kovar and Michael Gleicher, **Automated Extraction and Parameterization of Motions in Large Data Sets**, ACM Transactions on Graphics (SIGGRAPH 2004), Vol.23, No.3, pp.559-568, 2004.

逆運動学計算の利用時の注意

- いずれの方法を用いても、入力姿勢から大きく離れた姿勢を生成することは難しい
  - 広い範囲の自然な姿勢の生成は難しい
- 姿勢や動作によっては、適切な末端関節の目標位置を与えることが難しい
  - 動作生成のための軌道を与えることは難しい
- 微小な姿勢の変化であれば対応可能
  - 手先や足先の位置を周囲の環境に応じて少し修正する程度の姿勢変形

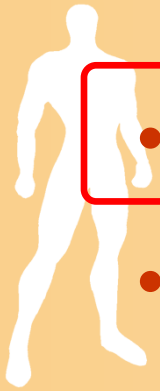
逆運動学の計算方法のまとめ

- いくつかの計算方法がある
  - 数値的解法（Numerical）
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 粒子法（Particle-IK）
    - 非線形最適化問題による解法
  - 解析的解法（Analytical）
  - サンプルデータを用いる解法



今回の内容

- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ



レポート課題（2）

レポート課題

- キャラクタ・アニメーション(2)
  - サンプルプログラムの未実装部分(前半)は作成済み
  1. 順運動学計算
  2. 姿勢補間
  3. キーフレーム動作再生
  4. 動作補間
  - サンプルプログラムの未実装部分(後半)を作成
  5. 動作接続・遷移
  6. 動作変形
  7. 逆運動学計算(CCD法)

レポート課題間の関連

- 前の課題で作成した処理を、次の課題でも使用

1. 順運動学計算
2. 姿勢補間
3. キーフレーム動作再生
4. 動作補間
5. 動作接続・遷移
    1. 動作接続のみ
    2. 動作接続・遷移
6. 動作変形
7. 逆運動学計算（CCD法）
    1. ルート体節を支点とする場合
    2. 任意の関節を支点とする場合

レポート課題

- キャラクタ・アニメーション基礎技術
  - サンプルプログラムをもとに作成したプログラム（???App.cpp）を提出
    - 他の変更なしのソースファイルやデータは、提出する必要はない
  - 全ての課題が終わらない場合は、できた部分だけでも提出する（ある程度できていれば合格点とする）
    - 作成しやすい課題から、任意の順番で作成して良い
    - できなかった課題の項目は、レポートのテンプレートから削除すること
  - Moodleの本講義のコースから提出
  - 締切：Moodleの提出ページを参照

レポート課題 提出方法

Moodleから、以下のファイルを提出

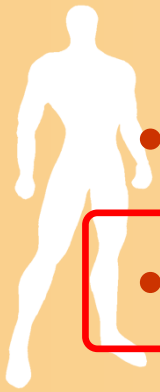
- 作成したプログラム（テキスト形式）
  - MotionTransitionApp.cpp
  - MotionDeformationApp.cpp
  - InverseKinematicsCCDApp.cpp
- 変更箇所のみを抜き出したレポート（PDF）
  - Moodle に公開している LaTeX のテンプレートをもとに、作成する

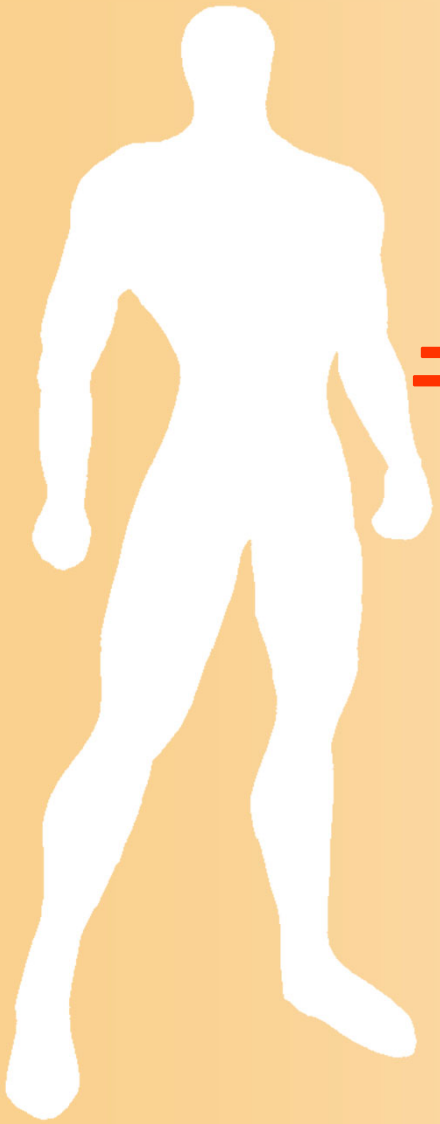
レポート課題 演習問題

- レポート課題の提出に加えて、レポート課題の理解度を確認するための Moodle 演習問題にも解答する
  - 解答締切は、レポート提出と同じ
    - 締切までは解答を変更可、締切後に正解が表示
  - レポート課題のヒントにもなっているので、レポート課題で分からない箇所がえれば、演習問題の説明・選択肢を参考にして考えても良い
  - 本演習問題の点数は、演習課題の成績の一部として考慮する

今回の内容

- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ





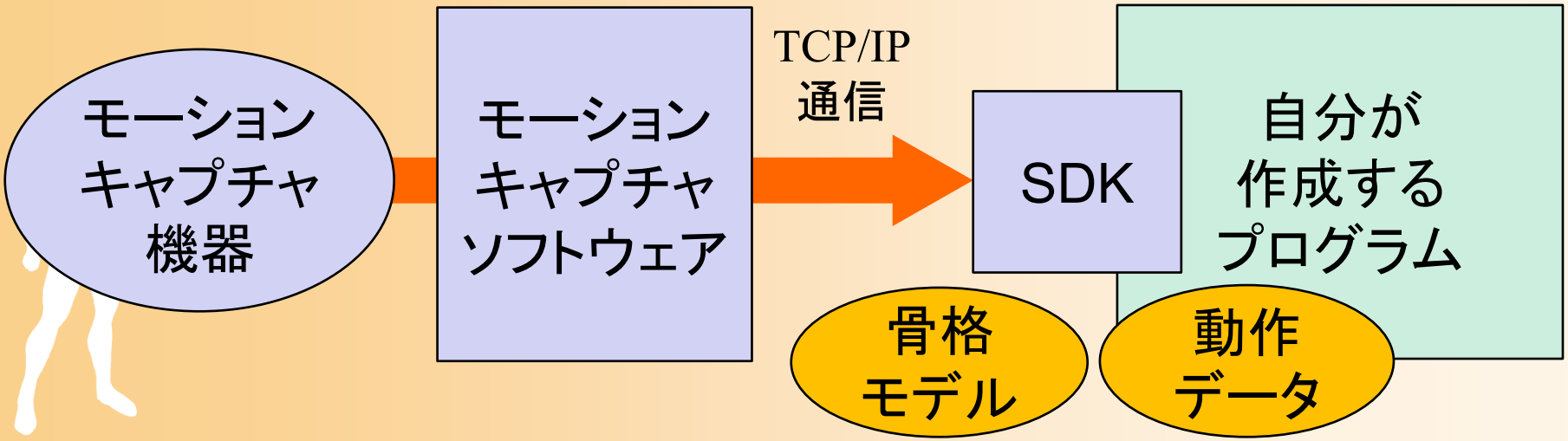
モーションキャプチャ

モーションキャプチャの用途

- 動作データの作成(オフライン用途)
  - 自分のプログラムでモーションキャプチャ処理を行う必要はない
  - モーションキャプチャ用のソフトウェアを利用可能
- 利用者の姿勢・動作の取得(オンライン用途)
  - 自分のプログラムでモーションキャプチャ処理を行う必要がある
  - モーションキャプチャ付属のライブラリを利用可能
    - 大部分の処理はライブラリで行われる
    - 姿勢データの取得や変換のプログラミングが必要

モーションキャプチャ機器からのデータ取得方法

- 多くのモーションキャプチャシステムでは、付属のSDK(ライブラリ)を利用することで、モーションキャプチャソフトウェアから、骨格モデルや動作データをリアルタイムに取得可能



モーションキャプチャの実行手順

1. 骨格モデルの生成
   - 通常、体節・関節の数や種類は、モーションキャプチャ機器によって決まっている
   - 各体節の長さの情報を、モーションキャプチャ機器からの情報にもとづいて決定する
2. 姿勢データの取得
   - 毎フレームに、モーションキャプチャ機器からの情報を取得
   - 取得情報を姿勢データに変換（逆運動学計算）
   - 全フレームの姿勢をまとめて動作データとする

モーションキャプチャ機器の比較（復習）

方式 （製品例）	光学式 （Optitrack）	慣性式 （Perception Neuron）	RGB-Dカメラ方式 （Kinect）
精度	高	中	低
スーツ着用	要	要	不要
専用スペース	必要	不要	不要
動作（姿勢）取得	○	○	△
骨格モデル推定	○	×	△
指の姿勢取得	△※1	○	△※2
顔の表情取得	△※1	×	△※2
最大秒間フレーム数	120	60	30
価格	数百～数千万円	数十～一千万円	数万円

※1　全身の姿勢とは別にキャプチャを行う必要がある
※2　限定された状態の識別しかできない

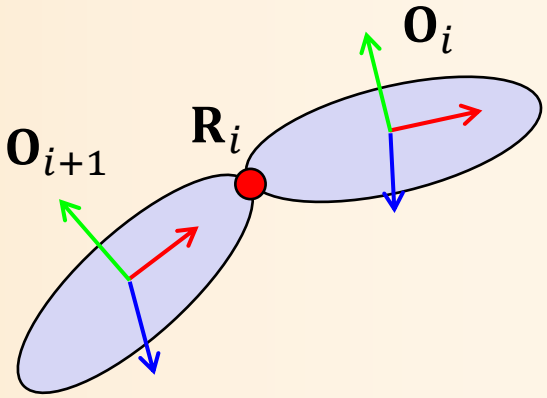
モーションキャプチャ機器からの取得データ

- センサの種類により、取得可能なデータは異なる
  - 光学式
    - 各マーカの位置 → 各体節の位置・向き
  - 磁気式
    - 各センサの位置・向き → 各体節の位置・向き
  - 慣性式
    - 各センサの並進・回転加速度・傾き → 各体節の位置・向き
  - RGBDカメラ（マーカレス式）
    - RGBD画像 → 各関節の位置
- ライブラリによっては、姿勢データ（全関節の回転＋腰の位置・向き）への変換が行われ、取得可能

モーションキャプチャの姿勢計算

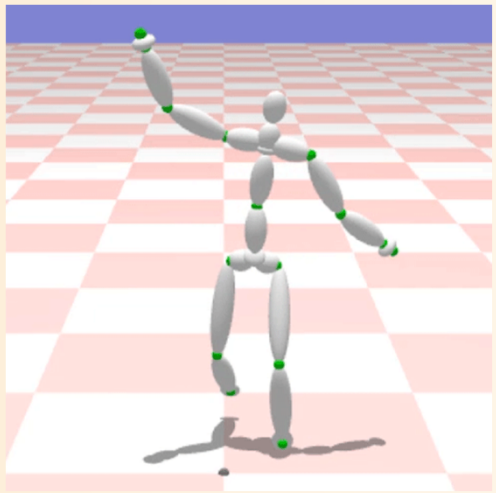
- ライブラリからの取得データから、逆運動学計算により、姿勢を求める
  - 各関節の回転＋腰の位置・向き
    - そのまま姿勢データとして使用可能
  - 各関節の位置
    - 粒子法の逆運動学計算と同様の方法で姿勢を計算
  - 各体節の位置・向き
    - 前後の体節の向きから関節の回転を計算できる

$$\mathbf{O}_{i+1} = \mathbf{O}_i \, \mathbf{R}_j \qquad \mathbf{R}_j = {\mathbf{O}_i}^{\mathrm{t}} \, \mathbf{O}_{i+1}$$

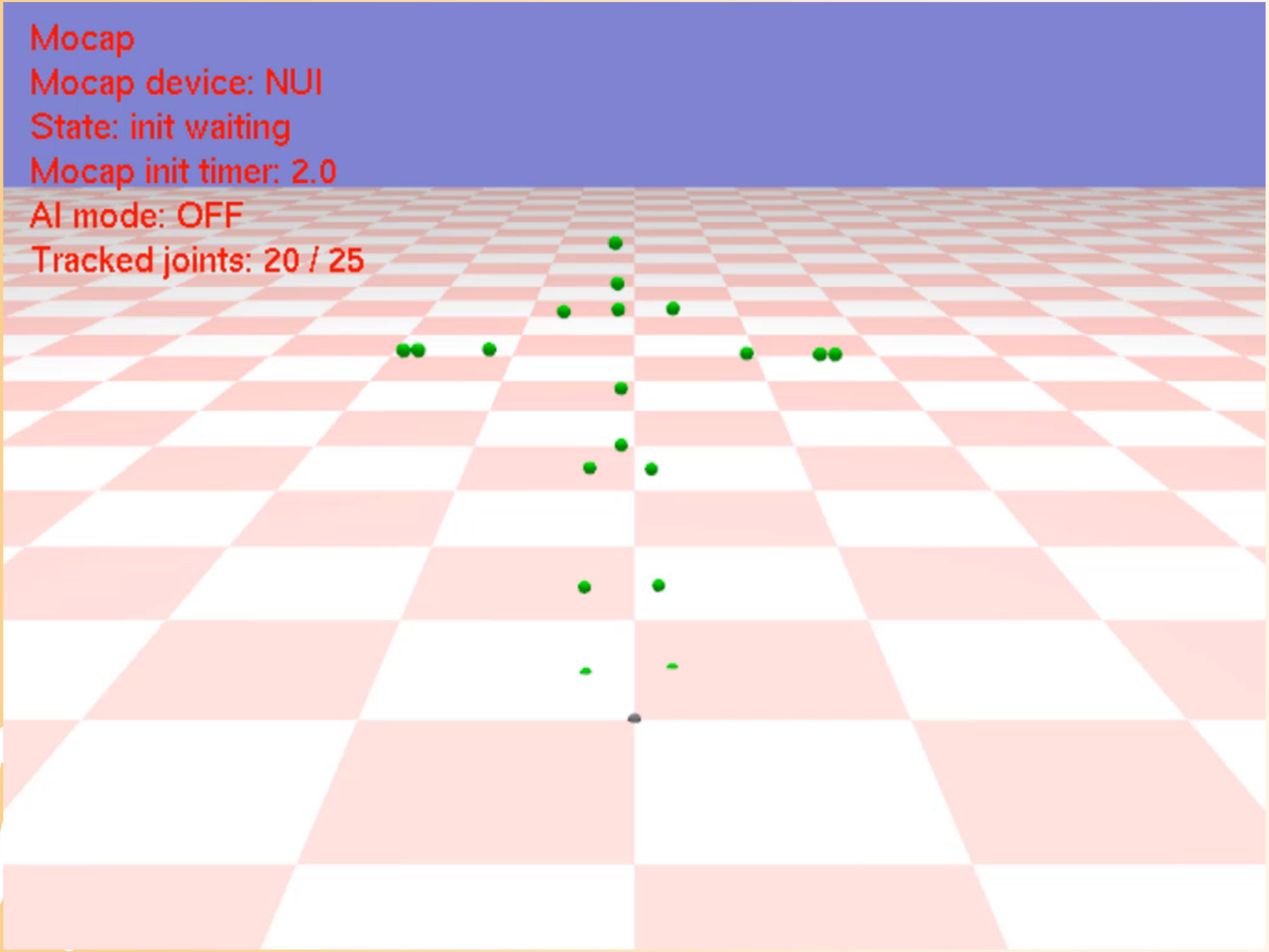


デモプログラム

- モーションキャプチャ
  - RGBDカメラ（マーカレス式）モーションキャプチャ
    - 実行には対応機器・ソフトウェアが必要
  - RGBDカメラから取得した画像から、全関節の位置を解析（外部ライブラリによる処理）
  - 全関節点の位置から姿勢を計算



Intel RealSense Camera + Nuitrack SDK


Mocap
Mocap device: NUI
State: init waiting
Mocap init timer: 2.0
AI mode: OFF
Tracked joints: 20 / 25

モーションキャプチャ利用例(1)

- Optitrack(光学式)
  - 計測データは、各カメラに写る各マーカの2次元位置
  - モーションキャプチャソフトウェアやライブラリで、全体節の位置・向きを計算
  - ライブラリを利用して、全体節の位置・向きを取得可能
    - 全体節の位置・向きから、姿勢を計算

Optitrack © NaturalPoint

モーションキャプチャ利用例(2)

- Perception Neuron(慣性式)
  - 計測データは、各センサの並進・回転加速度や方位(加速度計・ジャイロ・地磁気計)
  - モーションキャプチャソフトウェアやライブラリで、全体節の位置・向きを計算
    - さらに、姿勢(全関節の回転＋腰の位置・向き)まで計算
  - ライブラリを利用して、骨格データと姿勢データを取得可能

Perception Neuron © Noitom

モーションキャプチャ利用例(3)

- Kinect、RealSense、Nuitrack(RGBDカメラ)
  - 計測データは、RGBD画像
  - モーションキャプチャソフトウェアやライブラリで、全関節の位置を計算
    - 一部の関節の位置が求まらない場合もある
  - ライブラリを利用して、全関節の位置を取得可能
    - 全関節の位置から、姿勢を計算
    - 関節の位置が不明な場合の対応

Intel RealSense Camera + Nuitrack SDK

モーションキャプチャ利用時の問題

- 処理速度は、通常は、問題にはならない
  - 環境によっては遅延が問題になることはある
- 計測ノイズの問題への対応
  - センサーによっては、計測ノイズが多く、不自然な姿勢・動作になる場合がある
    - オフライン用途であれば、モーションキャプチャソフトウェアを使った修正が可能
      - 修正作業には、ある程度の手間がかかる
    - オンライン用途であれば、プログラムでの対応が必要
      - 処理速度や将来の姿勢が未知といった問題がある

まとめ

- 逆運動学
  - 数値的解法
    - ヤコビ行列の擬似逆行列による解法
    - Cyclic Coordinate Descent（CCD）法
    - 他の数値的解法
  - 解析的解法
  - サンプルデータを用いる解法
- レポート課題（2）
- モーションキャプチャ

次回予告

- 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、人体形状変形モデル
- 姿勢補間、キーフレーム動作再生、動作補間
- 動作接続・遷移、動作変形
- 逆運動学、モーションキャプチャ
- 動作生成・制御