Forward Kinematics
&
Inverse Kinematics

2024 Computer Animation and Special Effects

Outline

- Overview
- Objective
- Report
- Scoring
- Submission

Overview

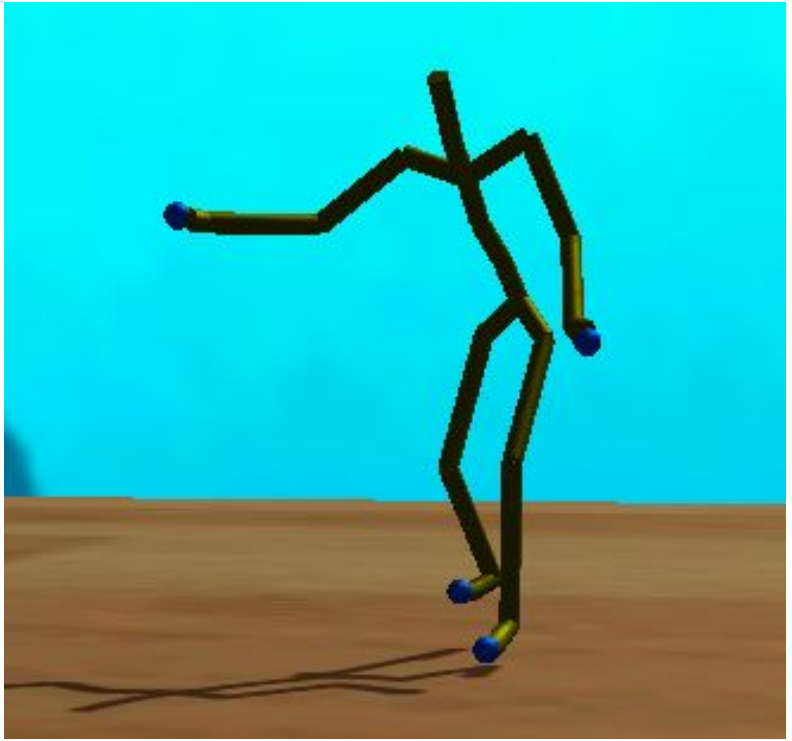
- Forward kinematics

  More amc files you can find in http://mocap.cs.cmu.edu/

Demo link: https://youtu.be/CUQRCwjztrQ

Overview (cont.)

- Inverse kinematics



Demo Link : https://youtu.be/CUQRCwjztrQ?si=h1e-s2eqCPDsbYUQ&t=24

Demo

Demo link: https://youtu.be/CUQRCwjztrQ

Objective

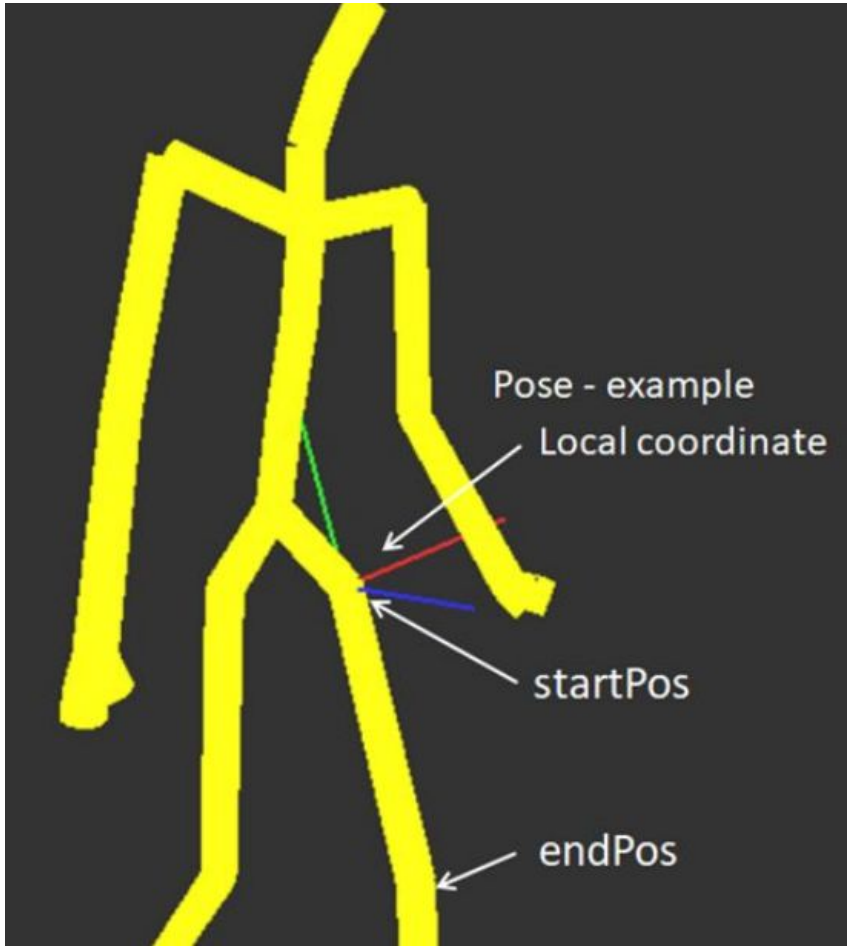
- In FK part, only one function you need to implement
  - in kinematics.cpp
    - void forwardSolver(...)

Objective (cont.)

- void forwardSolver(...)
  - Convert motion data from joint space to the Cartesian space
    - set each bone's global start and end position and rotation
  - Hint
    - review "kinematics.pptx" from p.1 - p.19
    - review "acclaim_FK_IKnote.pdf" from p.1 - p.4
    - read local coordinate data from posture first
    - you can probably use DFS or BFS to traverse all bones
    - you can check
      - struct Posture in posture.h
      - struct Bone in bone.h

Objective (cont.)

- Pose example
- Each bone has
  - local coordinate
  - start position
  - end position



Objective

- Because IK also use forwardSolver function, we **strongly recommend you to complete FK part first**.

- In IK part, there are two function you need to implement in this homework
  - in kinematics.cpp
    - Eigen::VectorXd pseudoInverseLinearSolver(...)
    - bool inverseJacobianIKSolver(...)

Objective (cont.)

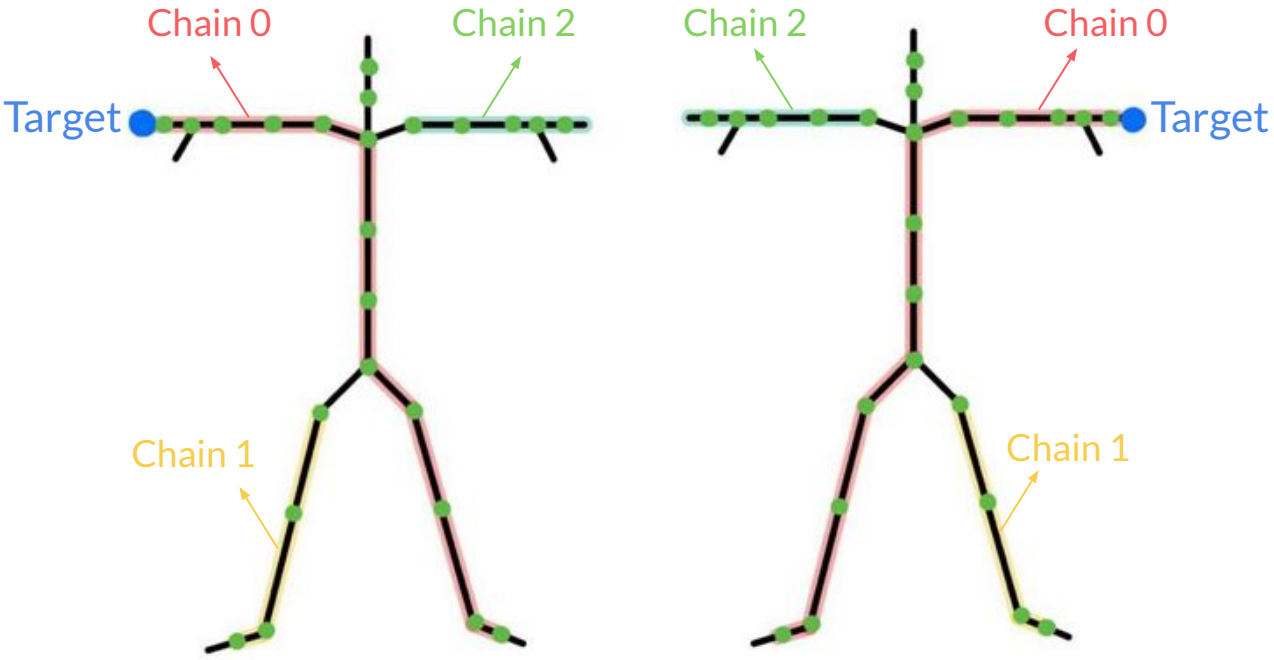
- VectorXd pseudoInverseLinearSolver(...)
  - Goal
    - Find solution of linear least squares system, which will be needed for inverse kinematics
    - i.e. find x which min(| jacobian * x - target |)
  - Hint
    - You might use some pseudo-inverse methods such as SVD
    - There are some built-in functions in Eigen that you can use
    - Eigen::Matrixs4Xf means a matrix with 4 rows and unknown columns
      - Eigen::Matrix4Xf m(4, 10); // A matrix with 3 rows and 10 columns
    - Eigen::VectorXf means a vector with unknown size
      - Eigen::VectorXf v(10); // A vector with 10 elements

Objective (cont.)

- bool inverseJacobianIKSolver(...)
  - Goal
    - Implement inverse kinematics
    - We use inverse-Jacobain method in this homework
  - Hint
    - Review "kinematics.pptx" from p.20 - p.50
    - Review "acclaim_FK_IKnote.pdf" Inverse Kinematics part
    - Traverse each chain by jointChains[chainIdx], boneChain[chainIdx]
      - Make *jointChains[chainIdx][0] touch the ball (target_pos[chainIdx])
      - Joint i and i+1 are the endpoints of bone i
    - You can check struct Bone in bone.h

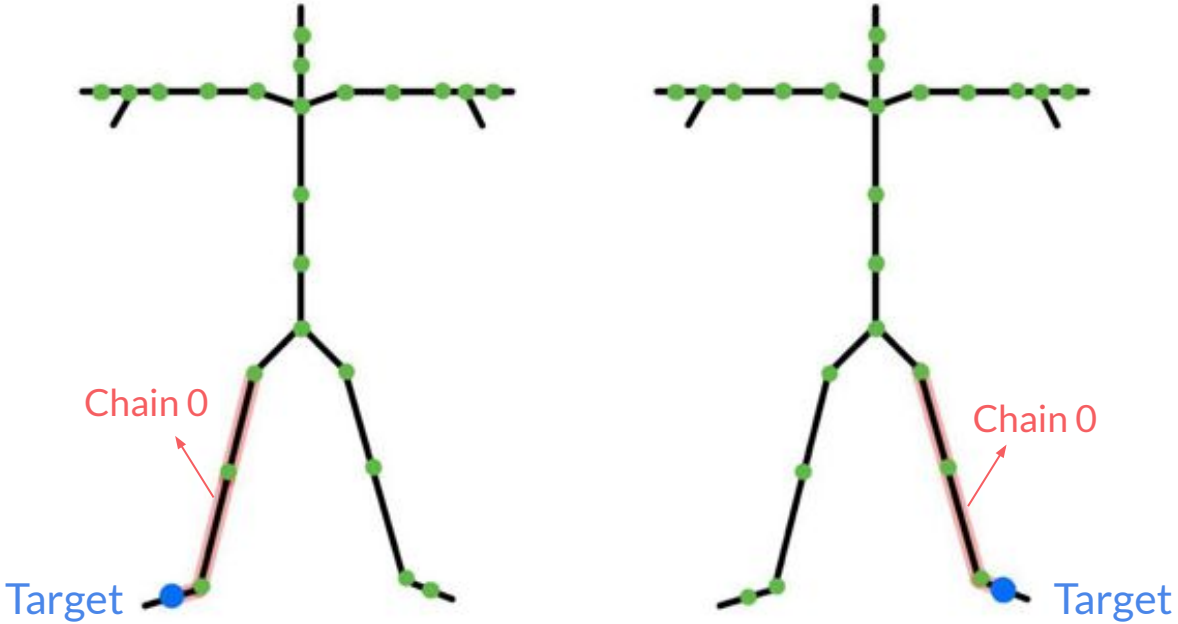
Objective (cont.)

- To achieve root movement, we employed the following methods.



Objective (cont.)

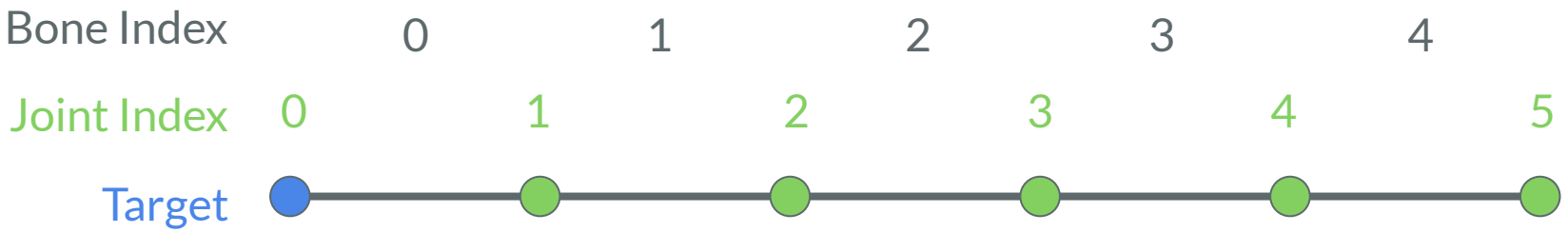
- To achieve root movement, we employed the following methods.



Objective (cont.)

- We will pass the corresponding chain as parameters into the inverseJacobianIKSolver() based on the currently dragged target.

```
std::vector<std::vector<Eigen::Vector4d*>> &jointChains
std::vector<std::vector<acclaim::Bone*>> &boneChains
```



Report

- Suggested outline
  - Introduction/Motivation
  - Fundamentals
    - Describe local and global coordinates in your words
  - Implementation
  - Result and Discussion
    - IK part : How different step and epsilon affect the result
  - Bonus (Optional)
  - Conclusion
  - Demo link (Google Drive, Youtube)
    - Show your FK and IK result
    - Up to 1 min

Scoring

- Forward kinematics (40%)
- Inverse kinematics (40%)
- Report (20%)
- Bonus (up to 20%)
  - Any creativity

Submission

- Please upload kinematics.cpp and report_< your student ID>.pdf respectively
  - If you make any modifications to other codes, you will need to upload them and discuss them in the report.
- Late policies
  - Penalty of 10 points on each day after deadline
- Cheating policies
  - 0 points for any cheating on assignments
- Deadline
  - Sunday, 2024/05/05, 23:59