TECHNISCHE UNIVERSITÄT
CHEMNITZ

BACHELOR THESIS

Implementation of a modular pipeline to evaluate different rigging and retargeting techniques for virtual humans using CrossForge

Faculty of Computer Science
Professorship of Computer Graphics and Visualization

Author:
Mick KÖRNER

Examiner:
Prof. Dr. Guido BRUNNETT
Supervisor:
Dr.-Ing. Thomas KRONFELD

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

December 10, 2024

Declaration of Authorship

I, Mick KÖRNER, declare that this thesis titled, "Implementation of a modular pipeline to evaluate different rigging and retargeting techniques for virtual humans using CrossForge" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY OF TECHNOLOGY CHEMNITZ

Abstract

Professorship of Computer Graphics and Visualization

Bachelor of Science

Implementation of a modular pipeline to evaluate different rigging and retargeting techniques for virtual humans using CrossForge

by Mick KÖRNER

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

Contents

List of Figures

List of Tables

Notes

Chapter 1

Introduction

1.1 Motivation

TODOm 1.1.1: Motiovation or Objectives and Scpoe?

Virtual Humans have been a major Part of Computer Graphics because of its wide range applications, spanning multiple research domains.

Creating a realistic Virtual Human is still a challenge today. Digital Reconstruction techniques like Strucute-from-Motion can create a very Detailed Surface replication of a Person. However, this Mesh is static. If it is desired to animate this Scan with Motion Capture Data, the Mesh does not contain any Information on how to apply these.

While Motion-Capture techniques like Shape-from-Silhouette exist, which are creating an Animation by storing a 4D Mesh. The use Cases for these Results are limited because the Motion and Virtual Character are coupled.

Simplifying the Virtual Human problem to decouple Motion- and Surface Data has naturally developed to be the standard today, not only for Realistic Virtual Humans, but also heavily stylized ones in Movies and Games.

TODOm 1.1.2: Motiovation or Objectives and Scpoe?

- Another important Motivation was to provide an easy to access and open source tool for motion retargeting, all widely used retargeting tools either require payment or an account login. Notibly there do no exist solid free motion retargeting Solutions.

- no basic tool for simple customizable motion retargeting

- while ik is already a common tool for animators to quickly get a desired pose, a well implemented and accessable motion retargeting can further improve an animators workflow by posing as a starting base for a desired pose using other motion editing tools

A deeper look into existing tools for these Problems reveals that many of them are sub-optimal or require some form of payment. Either in form of Currency or User Data.

1.2 Objectives and Scope

To facilitate the option to use a large set of Motion Data with Rigged Characters popular Tools like Mixamo use standardized Human like Skeleton to simplify the Process by moving the Motion Retargeting Problem to a Auto-Rigging Problem. Thus for a scalable system, the underlying Skeleton should be abstractable and independent of Motion Data. This is however not easy.

The primary Goal is a Tool which automates or streamlines the process of creating a Virtual Character just from a Scan. This includes the Implementation of Interfaces to easily add new methods for Autorigging and Motion Retargeting.

To further support Scalability for Future use. The proposed Tool should be interactive in order to test and compare algorithms more easily for correctness and potential drawbacks.

1.3 Summary of the Work

Firstly we will go over all Related Works in Chapter 2. This includes a Recap of how Computer Animation works and their basics. Then we go over Inverse Kinematics, Constraints up to Motion Retargeting and AutoRigging in Chapter 2.

In Chapter 3 the Design and Implementation of the Automation Tool is explained. As well as details specific Implementations of Motion Retargeting and Autorigging Methods or API interfaces.

Chapter 2

Related Work

2.1 3D Animation Basics (0.5 Basics)

Prior to examining the literature pertinent to this thesis, it is essential to define the fundamental principles of skeletal animation in computer graphics, establish consistent nomenclature, and establish a foundation to prevent confusion. In the field of cross-paper naming, it is not uncommon for different designations to be used for the same concept or for separate concepts to be merged into a single term.

Furthermore, many papers adopt a clear and consistent naming convention prior review.

The most prevalent form of humanoid animation is skeletal animation. The majority of graphics engines are capable of supporting this type of animation due to its inherent simplicity. This has led to its early adoption as a standard feature in hobby engines, with numerous motion editing tools in the industry also built around it.

2.1.1 Skeletal Animation

- simillar to how a animals in the real world have a rigid bones connected to a skeleton and moved with muscles, a similar analogy developed in computer graphics in a bionics manner

TODOm 2.1.1: source? figure 2.1

A skeleton is comprised of multiple bones arranged in a hierarchical structure, typically a tree-like configuration. These bones are associated with a length attribute. Joints represent the connection points between bones and are characterized by a rotational degree of freedom.

In addition to joints connecting two bones, root and end effector joints are of particular interest.

A root joint has no parent. Any transformation applied to this joint is reflected in the actor's global movement. In animation, this joint is often translated in conjunction with a walking animation, ensuring that the actor does not remain stationary while walking. While this could be achieved through the use of a scenegraph, it facilitates the unification of motion playback across applications by circumventing the necessity for an additional abstraction.

explain chain

- chain

In implementations Bones and their parent joints are often combined. Since the parent joint describes the rotation of th

Bones are usually not explicitly defined in implementations and are implicitly included in their parent joint

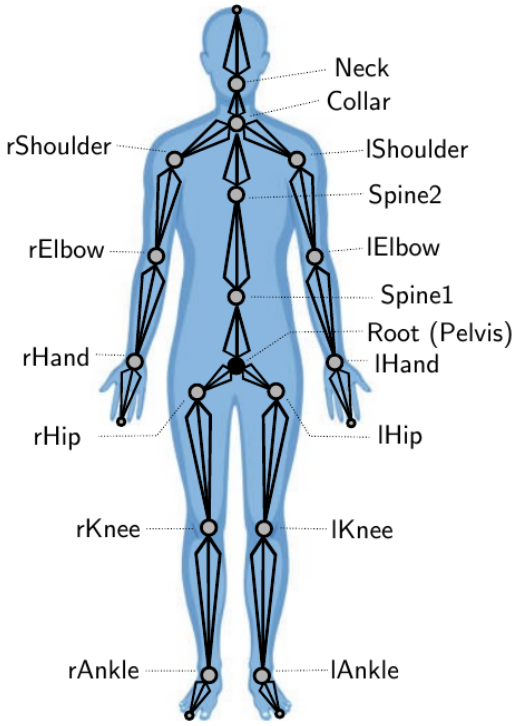

FIGURE 2.1: example of human skeleton, note that bones and their parent joint are combined, this can cause confusion, in this example the root and collar joint have multiple bones.

2.1.2 Pose Space vs. Work Space

Established common Spaces in the Graphics Pipeline include Window Mapping (NDC and Camera space), but more importantly for this work, World Space and Object Space. Object Space in regards to Skeletal Animation means the Space of the character in restpose.

later important expl

- In order to visualize a skeleton or parent other objects in worldspace to joints, for example a tool to simulate some kind of work. We need to know the position of a desired Joint in pose θ.

As discussed previously joints describe rotation of their child bones. To determine Position of Joints relative to Object Space, all kinematic chains from the root bone have to be propagated.

TODOm 2.1.2: merge sec with Forward Kinematics?

2.1.3 Forward Kinematics

- forward kinematics describes the process of computing the working space from pose space parameters Let F be the forward Propagation of the kinematic chain and θ the current pose configuration, object space position and rotation t of the endeffector can be computed as:

$$t = F(\theta)$$

explain affine matrix multiplication (rotation + translation)

- for affine transformation the propagating the chain results in an global rotation and translation

chain loops

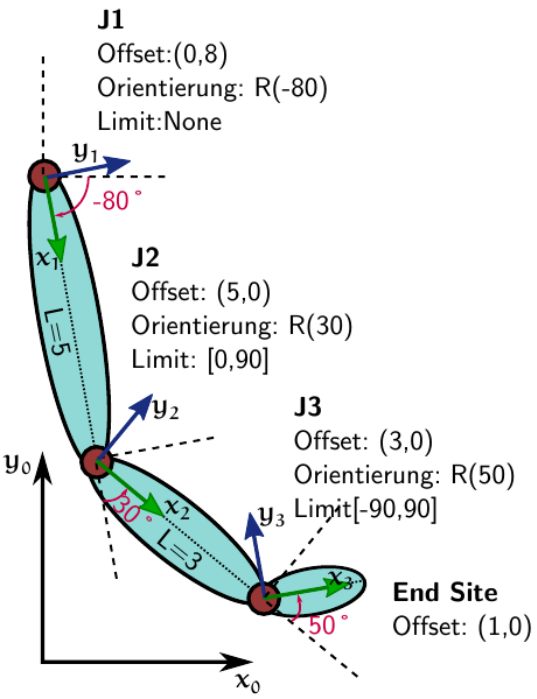

FIGURE 2.2: shows an example of a joint chain and their local coordinate systems

2.1.4 Restpose and Bind Pose Matrix

Because Character Modellers or Scans have to define the surface of a Virtual Character in an existing pose, Bones have to be placed correctly in that Character. Joint rotations of a Motion are then applied relative to the restpose angle of that joint.

TODOm 2.1.3: later first?

This also suggest that motion transfer between skeletons poses already a challenge when restposes are different.

explain what term rig mean beforehand

The Bind Pose Matrices are assigned per Joint and describe the transformation from the Object Space Koordinate system of the rigged character to the corresponding Joint in Restpose.

TODOm 2.1.4: keywords cursive?

The Inverse Bind Pose Matrix, as the name implies, does the opposite of the bind pose matrix, in various paper and code sources this is also commonly referred to as Offset Matrix.

Both Bind Pose and Offset matrix are defined with the skeletal hierarchy and their restpose once for a character. The Offset Matrix is essential part for efficient Linear Blend Skinning.

2.1.5 Skeletal Skinning

- for now we have a skeletal definition, but what was initialy wanted was to animate a character mesh easily - the Idear of Skeletal Animation is to abstract parts of the body away into joints, this is to reduce the complexity by defining motion of every single surface vertex manually. For Skeletal Animation, Vertices of the character surface, also called Skin, is abstracted to a bone.

TODOm 2.1.5: skinning example?

This is done by assigning which vertex is affected by which bone. Furthermore, because Flesh is deformable and not rigid, there is a need to interpolate vertices near the joint of two bones, for a 2 bone example and a vertex inbetween them.

- depending on what kind of cloth a character is wearing, there is a need to define vertex weights. Vertex weights have been hand authored by weight painting or tools like

blender automatic weight computation, nearest bone name

- The most common used Skinning method is Linear Blend Skinning - there are many more skinning methods which try to fix artefacts of linear blend skinning, but this is not in the scope of this thesis

TODOm 2.1.6: earlier?

TODOm 2.1.7: in CForge?

- for linear blend skinning, the offsetmatrix moves the weighted vertices of a joint in object space to the center of the coordinate system, so that local rotations of a joint are applied correctly. Together the joint transformation chain with the offset matrix are combined into the skinning matrix, which then gets send to the vertex shader. There it is combined

explain math rotation and translation

2.1.6 Motion Data

For Motion Playback, Rotational, Translation and Scale values, per Joint. One pose configuration in an Animation is called Keyframe. A Motion consists of multiple keyframes played sequentially. Timepoints per Keyframe determine at which time of an Animation a given Pose should be displayed.

The Sampling rate determines how many Keyframes per second are contained in the animation.

explain bvh

- a common trick for gait motion is to use the sampling rate to create a variable amount of walking speeds from one animation without having to create or capture gait motion for every desired speed - nearest neighbor interpolation between keyframes would result in choppy animation playback, to get a smooth playback at lower sampling rates linear interpolation is an quick, ease and sufficient enough for pleasing results

TODOm 2.1.8: F-Curves, shortly?

2.2 Inverse Kinematics (1. Inverse Kinematics)

check

- forward kinematics desribed at we have joint angles and lengths, with which we can compute each subsequent joint starting point to get the endeffector position - inverse kinematics describes the need to get joint angles with which rigid joint lengths and a target position, the endeffector matches the target position

In the previous Section we learned that Forward Kinematics takes Input from the Configuration Space of a Rigged Model and gives us Working Space Coordinates we can use to Render a Skinned Mesh. But we could also do Collision test. or parent further objects a character could hold onto joints.

For an dynamic grabing motion a natural desire would be to know a Configuration to target any Point in Working Space.

formulation

- Definition IK - ik goal to find joint configuration where endeffectors move to desired targets, while movement should be smooth fast and accurate

Inverse kinematics (IK) is the process of determining a joint configuration that satisfies various working space conditions, such as reaching a target or avoiding specific regions in space.

formulation

- Animators use Inverse Kinematics to intuitively animate characters without having to rotate each bone individually

Inverse Kinematics pose a fundamental tool for Motion Editing, its not only used for Automating Processes or real time interactive applications, but by 3D animators themselfs as a helpful tool to model a desired pose more easily and quickly.

formulation

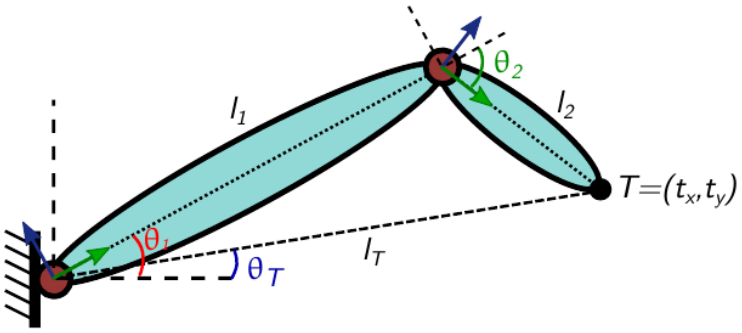

FIGURE 2.3

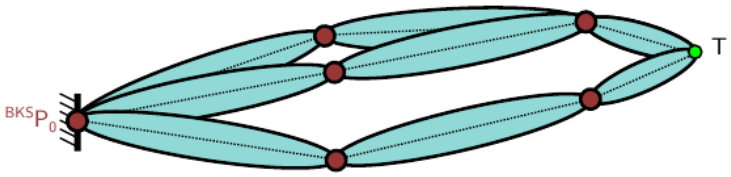

FIGURE 2.4

- very useful in animation be it movies and games as well as robotics - Inverse Kinematics widely used in Animation and Robotics industry

2.2.1 The IK Problem

reuse explaination of basics

The ideal approach would be to find a inverse mapping of the Forward Kinematics Mapping F so can get a pose configuration θ for a given target direction t:

$$\theta = F^{-1}t$$

TODOm 2.2.1: explain chain transform multiple solutions here?

2.2.2 Analytical Methods

The analytical approach tries to solve the system of equations spanned by inverting the Forward Kinematics formula of the corresponding armature.

- Lander [1] explained the analytical method simple for beginners.

main expl

While this solution would be Ideal because it is very fast and numerically perfect. Solving the system for more than two Joints becomes with each additional Joint harder.

TODOm 2.2.2: example durchgehen 2.3

This is beacause the Inverse Kinematics Problem can not be solved unambiguously.

2D example infinite solution

In 2D, a chain of more than two joints yield an infinite amount of solutions for a reachable Point in space. This already happens in 3d for chain length of two.

- multiple solution if chain length == target distance to chain root 1 sol - if target outside, no solutions

[2] has expl

2.2.3 Jacobian Methods

The Jacobian Inverse Method for solving Inverse Kinematics falls into the category of numerical solvers and represents the first Iterative Approach developed.

check

The primary challenge associated with inverse kinematics lies in the fact that pose space and working space are not linearly dependent. This implies that there exist multiple mappings of $F^{-}1$ that could potentially satisfy t. Consequently, determining the optimal solution becomes a complex task.

Furthermore, it is not uncommon for F to lack direct invertibility. This further complicates the determination of a unique and well-defined inverse function, or even the existence of such a function across the entire workspace.

go into detail with 2.5

When a joint is rotated, the resulting endeffector moves in a circular motion. This indicates that the forward kinematics function outputs a non-linear space in which the endeffector moves. 2.5 visualizes this difference for an endeffector.

However, it can be observed that this non-linear space can be approximated by a linear space for small amounts of movement:

Let J be a linear space mapping such that for a small movement of θ:

$$\Delta t \approx J(\theta)\Delta(\theta)$$

The Jacobian Matrix J is defined as the rate of change on Vector t when we turn angles of Joints in θ in each respective Dimension for a small amount Δ.

- Explicit values of J can then be evaluated by changing the corresponding angle of the armature by Δ and using the Forward Kinematics Function to determine the change of endeffector direction relative to its old position in object space.

For rate of change a common definition of the Jacobian Matrix is representing it using derivatives:

fill

$$J = \left(\frac{\partial F(\theta)_i}{\partial \theta_j} \right)$$

where i are respective Dimensions in which the target moves for each changeable angle θ_j.

$$\Delta\theta \approx J^{-1}(\theta)\Delta(t)$$

cite https://www.youtube.com/watch?v=wC and replace image with own

Buss [3] provides a more in-depth Introduction to the Jacobian Inverse Kinematics Method and how the Jacobian Inverse works.

put rigid explaination of simplifying calculation into chapter 3

2.2.4 Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD) were the first heuristic approaches to solving IK. Kenwright [4] wrote a great article which summarizes the History Workings and Constraints. There he stated that, due to its simplicity, it is not certain who published, but Wang and Chen [5] are credited.

- in order to reach a target point with an endeffector, each joint will be rotated so that the current vector from current joint position to endeffector points to the target

expl more in depth + picture

- there are two variants of CCD, one which starts rotating joints from the endeffector joint back to the root, and one that starts from the root and rotates the endeffector last

break condition?

Algorithm 1 CCD

Require:

TODOm 2.2.3: list of algorithms after index?

ccd algo

TODOm 2.2.4: use algo directly from other papers too? or write own version?

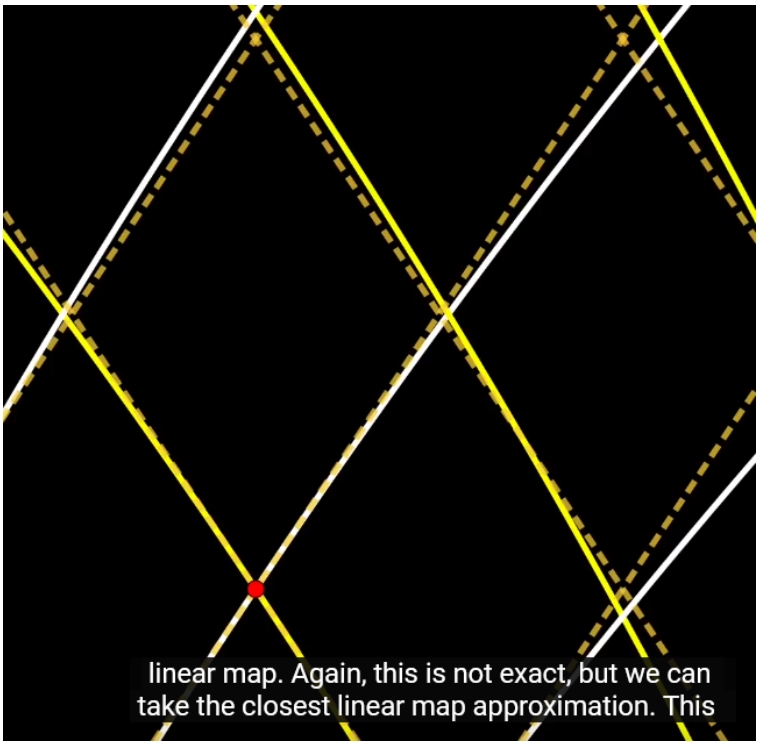

FIGURE 2.5

2.2.5 FABRIK

address CCD problems in CCD sec

- In order to improve performance and the rolling and unrolling Problem of CCDs, Aristidou and Lasenby [6] came up with Forward And Backward Reaching Inverse Kinematics (FABRIK)

- builds and optimizes upon ccd - fabrik noted producing more natural results, avoiding rollung and unrolling of ccd and moving the whole chain like jacobian inverse

TODOm 2.2.5: use algo directly from other papers too? or write own version?

Algorithm 2 FABRIK

Require:

fabrik algo

2.2.6 Other Methods

2.2.7 IK Surveys

TODOm 2.2.6: title? subsection-Comparison of IK methods

Aristidou et al. [2] present various Inverse Kinematics techniques in depth for general applications.

Boulic et al. [7] survey different Inverse Kinematics techniques to correct noisy and incomplete motion capture data from vision Input.

`https://zalo.github.io/blog/inverse-kinematics/#properties-of-various-ik-algorithms`

TODOm 2.2.7: seem to be only using custom methods?, dont go more into that?

complete

table to visualize comparison of ik methods like

2.2.8 Existing Tools

- Jacobian Methods Impl Because of the Mathematical complexity of Jacobian Methods, implementations are hard to find.

TODOm 2.2.8: into related or impl?

TODOm 2.2.9: Exisiting Tools section in review or impl? would be more relevant for going over drawbacks, so Id say implementation?

- CCD Impl
TODO Tex
- Fabrik Impl
- a - Final IK ik collection for unity - http://www.root-motion.com/finalikdox/htm
- paid - no source code

formulate

formulate

comparison from FABRIK paper

2.3 Constraints (2. Constraints)

2.3.1 Constraint Types

TODOm 2.3.1: only in impl? (Motion Retarget Editor) eher nicht oder?

2.3.2 Jacobian Constraints

2.3.3 CCD Constraints

2.3.4 FABRIK Constraints

2.3.5 iTASC

2.4 Motion Retargeting (4. Motion Retargeting)

2.4.1 Available Tools

2.4.2 Naive Retargeting

2.4.3 Limb based Retargeting

TODOm 2.4.1: category?

[8] - describes the problem to be hard to solve mathematically because of how to define the quality of a motion - require basic features of motion identified as constraints

Limb based Motion Retargeting approaches abstract Joints into Joint Chains, where each Chain is retargeted individually.

[9]

2.4.4 Jacobian based

TODOm 2.4.2: is Limb based?

Choi and Ko use Inverse Rate Control, which is the Jacobian Inverse Method of Inverse Kinematics, and extend it to be applicaple to tree structures instead of chains without branches. [10]

Choi and Ko have also showed a way to imitate joint angles of the source motion by incorporating them as a secondary goal.

2.4.5 Machine Learning Approaches

2.4.6 Other approaches

2.5 Automated Rigging (5. Autorigging)

2.5.1 Machine Learning Approaches

2.5.2 Thinning Approaches

TODOm 2.5.1: genauer anschauen für mögliche impl?

2.5.3 Skin Matching Approaches

2.5.4 SMPL fitting

2.5.5 Re-Meshing

Chapter 3

Motion Retarget Editor (6. Editor)

methodisches vorgehen hier

3.1 Chosen Tools

- Also having an open source foundation opens up community improvements and helps CrossForge mature by protyping features and incorporating them if deemed useful - because CrossForge is a relatively small Framework compared to Unity or Unreal Engine, many tools like Scene management, User Interfaces or Picking had yet to be implemented - for a fully automated pipeline, there is a need to keep various parts interactive for interactive testing to verify correct implementation of algorithms

user interface section?

3.2 Classes and Scene Management

TODOm 3.2.1: picking in scene management, UI before scene management?, picking uses smart pointers, easy to explain reasoning, but scene uses also smart pointers

3.3 User Interface

imguizmo needs to be before picking, thus ui should be before picking

3.3.1 Picking

3.4 Animation System

CrossForge already provided an implementation for skeletal animation playback using Linear-Blend-Skinning.

ref assimp

3.4.1 CrossForge format

For this feature CrossForge implements a direct approach. Assimp, the C++ library used for importing and exporting to various 3D formats. Provides the Inverse Bind Pose matrix. The purpose of this matrix is to transform the joint from global to local space so that local transformation of that joint are applied localy to the weighted vertices when doing linear blend skinning.

3.4.2 Sequencer

3.4.3 Editing Tools (Restore Restpose, apply Transform etc.)

3.5 Inverse Kinematics Implementation

While various Inverse Kinematics Implementations exist, , they are usually imple-

list impl

mented across various Programming Languages or use different 3D Engines, resulting in vastly different and complex APIs.

To reduce complications, various inverse kinematics algorithms proposed in section Inverse Kinematics (1. Inverse Kinematics) are re-implemented using CrossForges Animation Controller interface.

3.5.1 Jacobian Method

3.5.2 CCD

3.5.3 FABRIK

3.6 Motion Retargeting

subsection Combined Retargeting Methodologies (TODO eigenanteil in extra chapter)

3.7 Skeleton Matching

- while testing the new motion retargeting implementation, limbs were matched manually with a popup user interface - could define matching by limb names, but want to autogenerate armature

3.8 Constraints Implementation

section Combined Constraint System (TODO eigenanteil in extra chapter)

3.9 Import and Export

3.9.1 Model Data

3.9.2 Animation Data

3.10 foreign tool Integration

title

3.10.1 Rignet

Chapter 4

Conclusion and Future Work (7. Future)

4.1 Editor Improvements

4.2 Utilizing Skinning Alternatives

4.3 Other Useful Tools

4.4 Clothing

4.5 Motion Blending

4.6 Blender Addon

Bibliography

[1] Jeff Lander. "Oh My God, I Inverted Kine! 09/98: Graphic Content". In: (1998).

[2] A. Aristidou et al. "Inverse Kinematics Techniques in Computer Graphics: A Survey". In: *Computer Graphics Forum* 37.6 (Sept. 2018), pp. 35–58. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.13310. URL: https://onlinelibrary.wiley.com/doi/10.1111/cgf.13310.

[3] Samuel R Buss. "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods". In: ().

[4] Ben Kenwright. "Inverse Kinematics – Cyclic Coordinate Descent (CCD)". In: *Journal of Graphics Tools* 16.4 (Oct. 2012), pp. 177–217. ISSN: 2165-347X, 2165-3488. DOI: 10.1080/2165347X.2013.823362. URL: http://www.tandfonline.com/doi/abs/10.1080/2165347X.2013.823362.

[5] L.-C.T. Wang and C.C. Chen. "A Combined Optimization Method for Solving the Inverse Kinematics Problems of Mechanical Manipulators". In: *IEEE Transactions on Robotics and Automation* 7.4 (Aug. 1991), pp. 489–499. ISSN: 1042296X. DOI: 10.1109/70.86079. URL: http://ieeexplore.ieee.org/document/86079/.

[6] Andreas Aristidou and Joan Lasenby. "FABRIK: A Fast, Iterative Solver for the Inverse Kinematics Problem". In: *Graphical Models* 73.5 (Sept. 2011), pp. 243–260. ISSN: 15240703. DOI: 10.1016/j.gmod.2011.05.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S1524070311000178.

[7] Ronan Boulic et al. "Evaluation of On-Line Analytic and Numeric Inverse Kinematics Approaches Driven by Partial Vision Input". In: *Virtual Reality* 10.1 (May 2006), pp. 48–61. ISSN: 1359-4338, 1434-9957. DOI: 10.1007/s10055-006-0024-8. URL: http://link.springer.com/10.1007/s10055-006-0024-8.

[8] Michael Gleicher. "Retargetting Motion to New Characters". In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '98*. The 25th Annual Conference. Not Known: ACM Press, 1998, pp. 33–42. ISBN: 978-0-89791-999-9. DOI: 10.1145/280814.280820. URL: http://portal.acm.org/citation.cfm?doid=280814.280820.

[9] Yann Pinczon Du Sel, Nicolas Chaverou, and Michaël Rouillé. "Motion Retargeting for Crowd Simulation". In: *Proceedings of the 2015 Symposium on Digital Production*. DigiPro '15: The Digital Production Symposium. Los Angeles California: ACM, Aug. 8, 2015, pp. 9–14. ISBN: 978-1-4503-3718-2. DOI: 10.1145/2791261.2791264. URL: https://dl.acm.org/doi/10.1145/2791261.2791264.

[10] Kwang-Jin Choi and Hyeong-Seok Ko. "On-Line Motion Retargetting". In: (1999).