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 15, 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

parts starting with "-" need to be rewritten

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 each Chapter, fundamentals are explained. Popular and recent novel techniques will be discussed and available tools will be evaluated, not primairly in performance but also in availability, open-ness and ease of use.

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

Chapter 2

Related Work

2.1 3D Animation Fundamentals

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

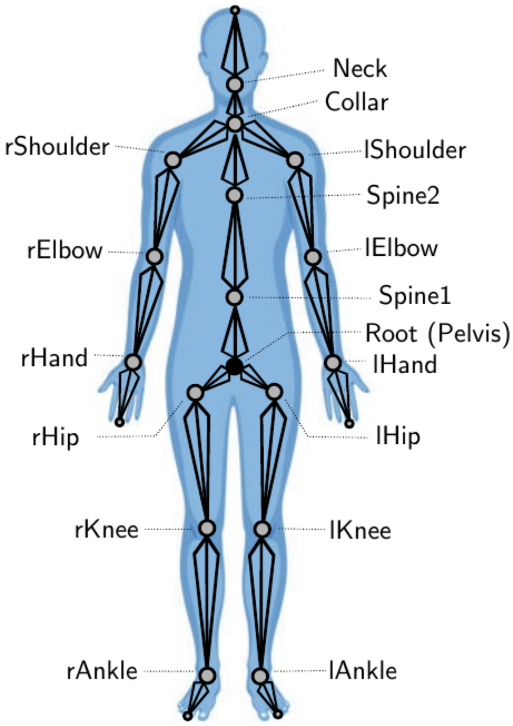

FIGURE 2.1: Example of human skeleton, note that bones and their parent joint are combined, this can cause confusion, in this example the root and collar joint have multiple bones. Image taken from [1]

Bones (sometimes called links) in Animation represent rigid objects inside virtual character - associated with a length attribute.

Joints represent the connection points between bones and are characterized by a rotational degree of freedom. - joint is the component concerned with motion;

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.

- endeffectors represent bones without children - depending on application sometimes it isnt clear if the endeffector is joint itself or a bone, having no joint at its tip, due to these discrepencies some systems having additional joints defined at it tips to ensure conversion between different formats happen seamlessly

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

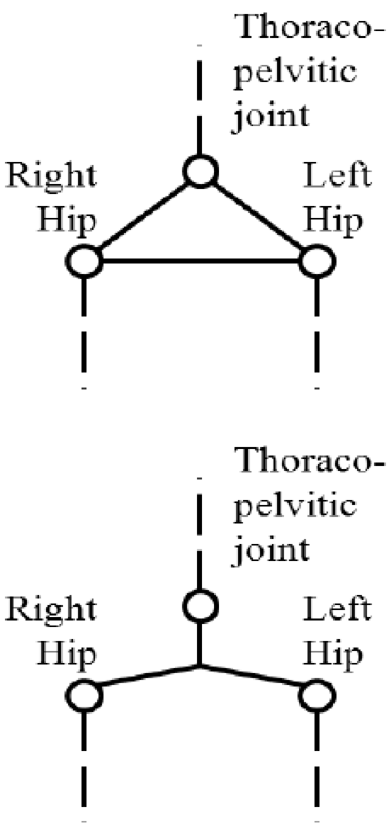

FIGURE 2.2: Visualization of an Armature with a loop, depending on implementation the tree becomes a Graph. Image taken from [2]

2.1.2 Skeletal Hierarchy

TODOm 2.1.2: tree-like?

A skeleton is comprised of multiple bones arranged in a hierarchical structure, typically a tree-like configuration.

explain chain

- a joint chain represents a link of multiple joints where each joint has at most 1 child

- branching happens when a joint has more than one child

TODOm 2.1.3: visualize? + position in TEX?

chain loops, extend

Closed loops - while tree structures are most often found, some systems allow for circular structures - using smartly placed bones, one can enforce constraits, for example ensuring 2 bones have always - harder to implement

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

meaning

? - useful for centre of rotation correction - not allowed in some systems, e.g. blender

2.1.3 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 2.1.4: merge sec with Forward Kinematics?

2.1.4 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:

$$t = F(\theta)$$

2.3 visualizes this process, translation, rotation (and sometimes scale) of each joint determines the transformation of all child joints.

math formula example

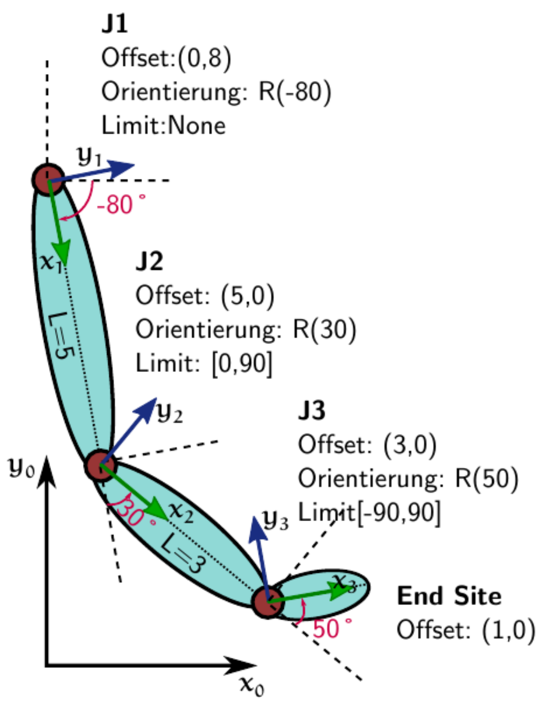

FIGURE 2.3: Example of a joint chain with respective local coordinate systems visualized, notice that the global transform of Joint J2 depends on J1, and J3 on J2 and J1. Image taken from [1]

explain affine matrix multiplication (rotation + translation)

- for affine transformation the propagating the chain results in object space relative rotation and translation

2.1.5 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 2.1.5: 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 2.1.6: 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.

2.1.6 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 2.1.7: 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

TODOm 2.1.8: earlier?

- 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 2.1.9: 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

TODOm 2.1.10: unnecessary? or later?

- Box-Based or Spherical Skinning - Dual Quaternion Skinning (DQS) - Delta Mush

2.1.7 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 2.1.11: F-Curves, shortly?

2.1.8 Other Animation Approaches

TODOm 2.1.12: unnecessary?

- Morph Targets - vertex animation textures (Vertex Shader Animations)
-> no one as flexible as skeletal animation due to abstraction of skeletal animation

2.2 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

- 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?

- The primary challenge associated with inverse kinematics lies in the fact that pose space and working space are not linearly dependent.

2.2.2 Reachability

- This is beacause the Inverse Kinematics Problem can not be solved unambiguously. - figure 2.4 shows that in 2D a chain of more than 2 joints yields an infinite amount of solutions for an reachable point

cases

- 3 cases - target is outsite of reach, no solution - depending on desired behavior, chain should be - target is at distance of of chain length ik is applied to - exactly one solution - easily identifyable, but rare occurance

TODOm 2.2.2: illustrate?

- target is within bounds of chain length - still problem that depending on skeletal definition sometimes points inside chain length are not reachable (e.g. long and small joint)

In 2D, a chain of more than two joints yield an infinite amount of solutions for a reachable Point in space, illustrated in Figure 2.4. This already happens in 3d for chain length of two.

2D example infinite solution

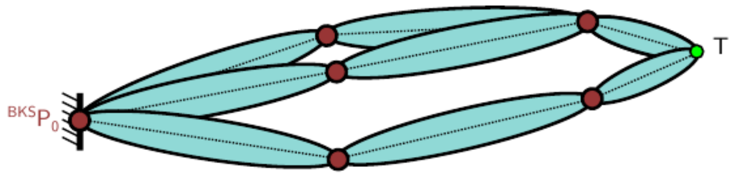

FIGURE 2.4: In 2D, for Chains with more than 2 Joints, there will be an infinite Amount of Configurations satisfying reaching the Target, when $|T - P_0| < |J1| + |J2| + |J3|$. Image taken From [1]

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

[3] has expl

2.2.3 Analytical Methods

- The analytical approach tries to solve the system of equations spanned by inverting the Forward Kinematics formula of the corresponding armature. Because they find solutions reliably, they are called Closed form solutions.

- Lander [4] explained the analytical method simple for beginners.

main expl

- Figure 2.5 illustrates relevant Variables for solving a 2 Joint chain. l_T, l_1 and l_2 span a triangle, because all lengths are given, trigonometric functions can be used to determine angles θ_1 and θ_2.

TODOm 2.2.3: example durchgehen 2.5

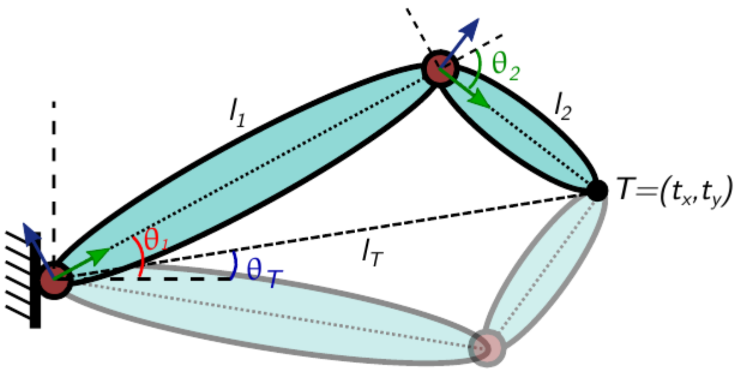

FIGURE 2.5: Visualization of relevant Variables for solving θ_1 and θ_2 analytically to reach point T using Trigonomic functions. A Second Solution is Visualized with less opacity below, Image taken from [5]

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

TODOm 2.2.4: correct? any source?

- but the computational increase with longer chains is not the only problem. providing either all Solutions or predetermined ones, finding solutions that produce plausible results related to temporal locality is even more difficult

2.2.4 Jacobian Methods

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

- also called inverse rate control

check

TODOm 2.2.5: explain chain transform multiple solutions here?

TODOm 2.2.6: formulation

Previously 2.2.1 the Problem of multiple pose space configurations satisfying target position constraints. 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 2.6

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. 2.6 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? and replace image with own

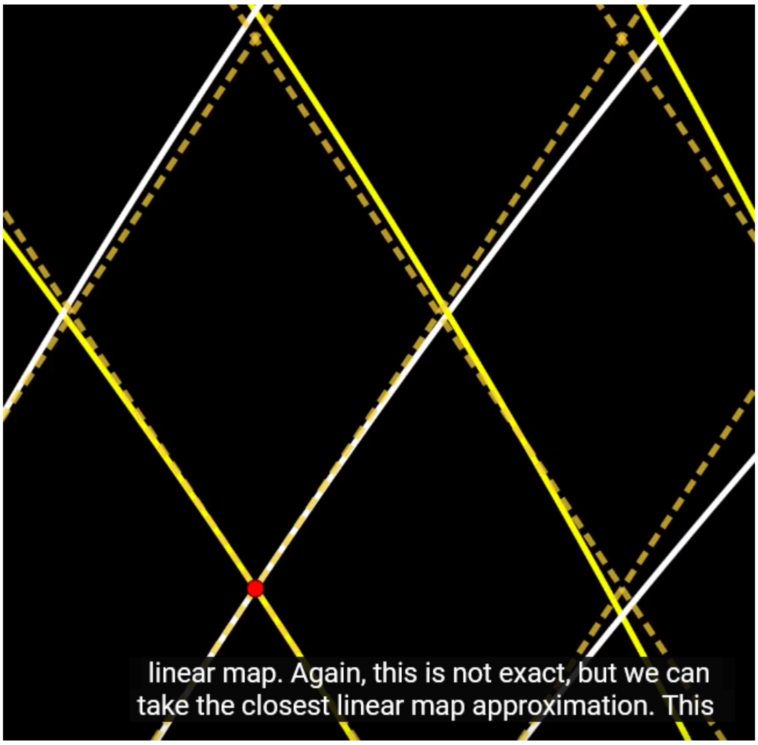

FIGURE 2.6

Buss [6] 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

- still lacking resources on how to implement Jacobian IK

2.2.5 Cyclic Coordinate Descent

Cyclic Coordinate Descent (CCD) were the first heuristic approaches to solving IK. Kenwright [7] 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 [8] 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?

TODOm 2.2.7: isnt this very inefficient?

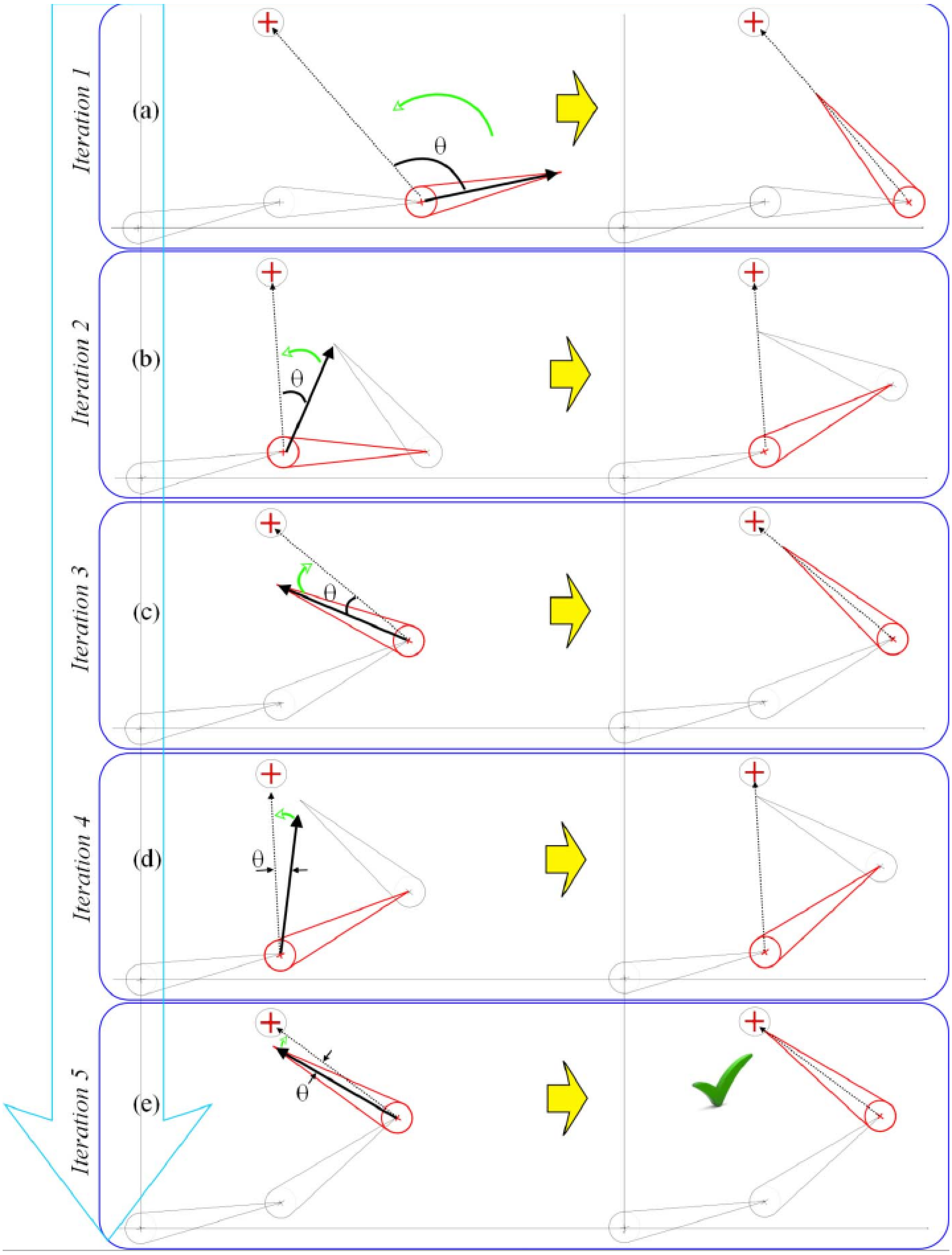

FIGURE 2.7: Three Joint Chain example of CCD, in each Iteration the Current Joint is rotated so that the Vector from current Joint to target and current Joint to endeffector align. Image taken from [7]

TODOm 2.2.8: list of algorithms after index?

Algorithm 1 BackwardCCDIK Algorithm, Taken From [7]

procedure BACKWARDCCDIK
  Input: e ▷ threshold
  Input: $k_{\max}$ ▷ max iterations
  Input: n ▷ link number (0 to numLinks-1 chain)
  $k \leftarrow 0$ ▷ iteration count
  while $k < k_{\max}$ **do**
    for $i = n - 1$ **to** 0 **do**
      Compute u, v ▷ vector $P_e - P_c$, $P_t - P_c$
      Compute ang ▷ using Equation 1
      Compute axis ▷ using Equation 1
      Perform axis-angle rotation (ang, axis) of link i
      Compute new link positions
      if $|P_e - P_t| < e$ **then** ▷ reached target
        return ▷ done
      end if
    end for
    $k \leftarrow k + 1$
  end while
end procedure

2.2.6 FABRIK

address CCD problems in CCD sec

- In order to improve performance and the rolling and unrolling Problem of CCDs, Aristidou and Lasenby [9] 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

- fabrik simplifies the

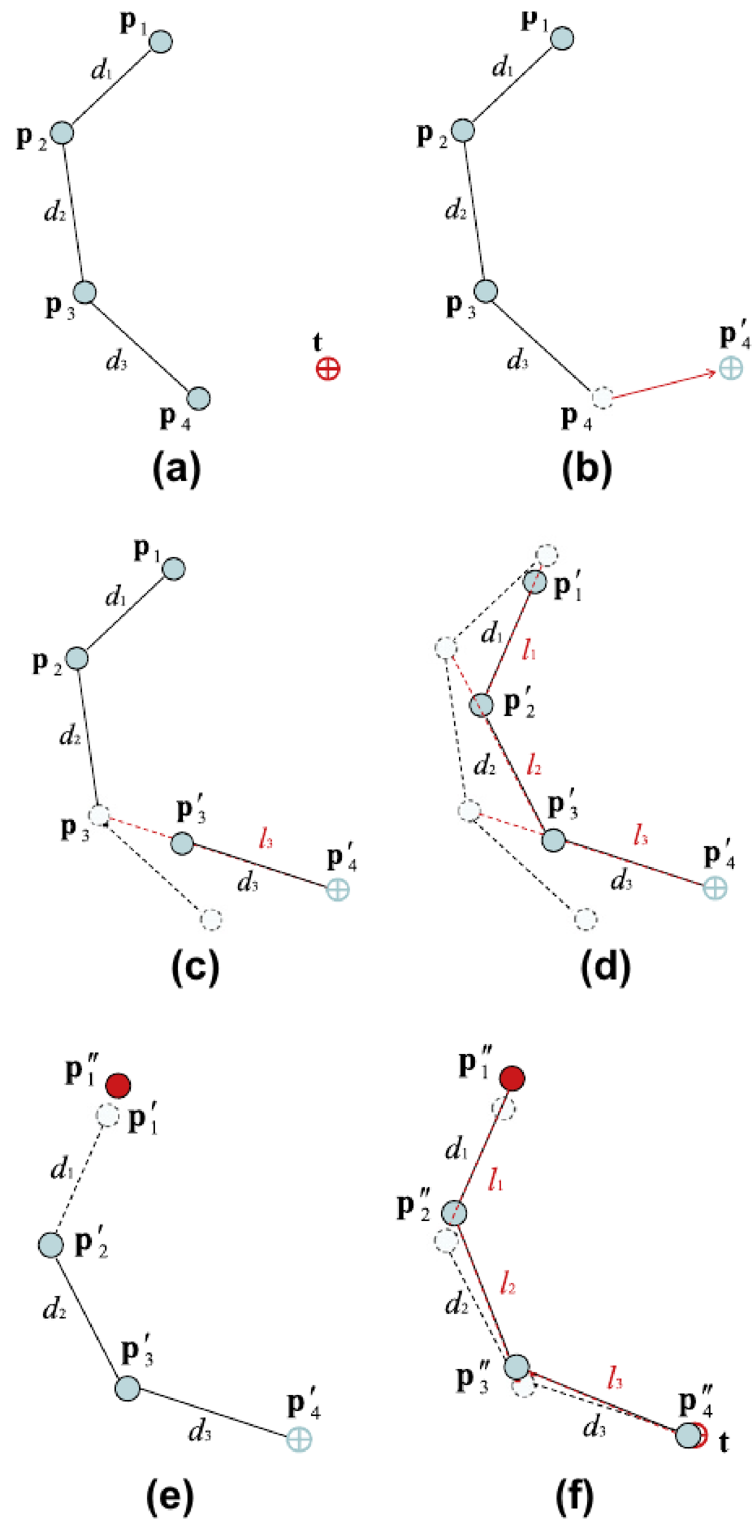

FIGURE 2.8: Image taken from [9]

Algorithm 2 A full iteration of the FABRIK algorithm, Taken from [9]

Require: The Joint positions p_i for $i = 1, \ldots, n$,
Require: target position t,
Require: distances $d_i = \|p_{i+1} - p_i\|$ for $i = 1, \ldots, n-1$
Output: New joint positions p_i for $i = 1, \ldots, n$
dist $\leftarrow \|p_1 - t\|$ ▷ The distance between root and target
▷ Check whether the target is within reach
if dist > $d_1 + d_2 + \ldots + d_{n-1}$ **then** ▷ The target is unreachable
  for i = 1 to n-1 **do** ▷ Find the distance r_i between the target t and the joint position p_i
    $r_i \leftarrow \|t - p_i\|$
    $k_i \leftarrow \frac{d_i}{r_i}$
    $p_{i+1} \leftarrow (1 - k_i)p_i + k_i t$ ▷ Find the new joint positions p_i.
  end for
else
▷ The target is reachable; thus, set as b the initial position of the joint p_1
  $b \leftarrow p_1$
  $dif_A \leftarrow \|p_n - t\|$
▷ Check whether the distance between the end effector p_n and the target t is greater than a tolerance.
  while $dif_A > tol$ **do**
▷ STAGE 1: FORWARD REACHING
    $p_n \leftarrow t$ ▷ Set the end effector p_n as target t
    for i = n-1 down to 1 **do**
▷ Find the distance ri between the new joint position p_{i+1} and the joint p
      $r_i \leftarrow \|p_{i+1} - p_i\|$
      $k_i \leftarrow \frac{d_i}{r_i}$
      $p_i \leftarrow (1 - k_i)p_{i+1} + k_i p_i$ ▷ Find the new joint positions p_i .
    end for
▷ STAGE 2: BACKWARD REACHING
    $p_1 \leftarrow b$ ▷ Set the root p_1 its initial position.
    for i = 1 to n-1 **do**
▷ Find the distance r_i between the new joint position p_i
      $r_i \leftarrow \|p_{i+1} - p_i\|$
      $k_i \leftarrow \frac{d_i}{r_i}$
      $p_{i+1} \leftarrow (1 - k_i)p_i + k_i p_{i+1}$ ▷ Find the new joint positions p_i.
    end for
    $dif_A = ||p_n - t||$
  end while
end if

2.2.7 Other Methods

There exist many more Methods for solving Inverse Kinematics, but - fail due to high cost

Newton Methods explained by [9] treat IK as a minimization problem but are slow and hard to implement.

TODOm 2.2.9: citation okay?, cite section in paper?, ref book in fabrik hard to understand

Mass Spring Models, is a IK method proposed by Sekiguchi and Takesue [10]. A numerical method which uses the virtual spring model and damping control. Suited for redundant robots, robots which have many DOF.

DOF Degrees of Freedom explain in basics

explain mass spring

expl Particle IK

TODOm 2.2.10: title? subsectionComparison of IK methods

2.2.8 IK Surveys

Aristidou et al. [3] present various Inverse Kinematics techniques in depth for general applications.

Boulic et al. [11] survey different Inverse Kinematics techniques to correct noisy and incomplete motion capture data from vision Input.

TODOm 2.2.11: seem to be only using custom methods?, dont go more into detail?

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

complete

table to visualize comparison of ik methods like

TODOm 2.2.12: into related or impl?

2.2.9 Existing Tools

TODOm 2.2.13: Exisiting Tools section in review or impl? would be more relevant for going over drawbacks, so Id say implementation?

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

- CCD Impl

TODO Tex

- Fabrik Impl

- a - Final IK ik collection for unity - http://www.root-motion.com/finalikdox/html - paid - no source code

formulate

formulate

comparison from FABRIK paper

2.3 Constraints

- In the previous section we looked at Inverse Kinematics abstractly as a motion editing tool, because IK is dynamic in nature and only considers Skeletal strucutre

Compared to the real world, we have yet to model DOF limiting factors of our Skeleton Bones like neighboring tissue like muscles, organs, fat or Connective tissue, as well as physical limits related to the atanomy and structure of the bones themselfes located at joints.

These are essential for Inverse Kinematics and its appliences in order to already avoid a set of self interpenetration Issues as well as non plausible poses to improve realism.

- many papers describe constraints specifically for an inverse kinematics method

- integration tied to a specific inverse kinematics method allows for optimization potential

- problematic is to incorporate constraints in a way that a global solution will still be found

2.3.1 Tree Structures

- in order for

- multiple inverse kinematic chains that share the same joint

- Jacobian Inverse Kinematics solves this naturally by incorporating not just one chain and a target, but all joints and targets into tha jacobian matrix.

2.3.2 Skeletal Constraints

- Aristidou et al. [12] have described six most common anthro- pometric Joint constraints, visualized in Figure 2.9.

- dependin on tpye various types of movements allowed

ball-and-socket joint - ball moves within a socket - limits angular rotation in the direction of parent joint

- hinge joint - simplest type of joint; - elbows, knees - motion only in one plane/direction about a single axis

- pivot Joint - only rotation on one axis, used in neck - for a given target, the head orientates towards it, - the target point has to be projected on the axis, and the rotation constraint has to be enforced

condyloid - ovoid articular surface that is received into an elliptical cavity - permits biaxial movements, that is, forward-backward and side to side, but not rotation

saddle - convex-concave surface, treated same as condyloid, e.g. thumbs - different angle limits, allowable bounds - no axial rotation

plane joint - also gliding joint, only sideways/sliding movements - requires IK rule relaxiation in form of joints are not connected anymore - done by projecting target onto joint plane bounds in algo

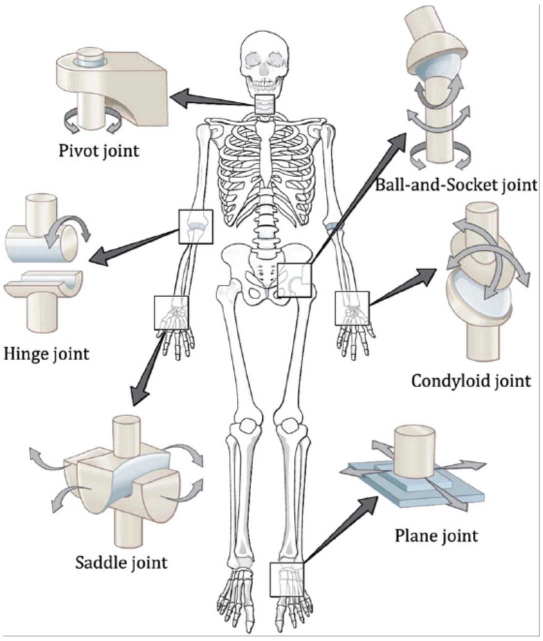

FIGURE 2.9: Various Constraint Types Visualized and where they could be used in a Virtual Human Skeleton. Image taken From [12]

2.3.3 Other Constraints

- Wilhelms and Gelder [13] proposed Reach cones, using spherical polygons to specifying a region for allowable joint movement.

- other constraints types that can be useful for motion editing

- distance constraints can ensure that either specific spaces are avoided or should be reached, for example elbow movement

self intersection

- the same way a constraint could be incorporated that checks for self intersection

TODOm 2.3.1: only in impl? (Motion Retarget Editor) eher nicht oder? (Jacobian, CCD, FABRIK)

- enforcing various constraints is difficult because it can affect an algorithms ability to converge

hinge limits

swing twist limit

2.3.4 Jacobian Inverse Constraints

complete

[14]

2.3.5 CCD Constraints

- While weighting CCD for multiple endeffectors can be intuitive Hecker [15] explained how to utilize priorities by averageing desired angles at branches.

- To ensure CCD doesnt run into a local minima under influence of constraints simulation annealing is used [7]. - Simulation annealing for CCD tries to jump out of a local minima by randomly rotating joints.

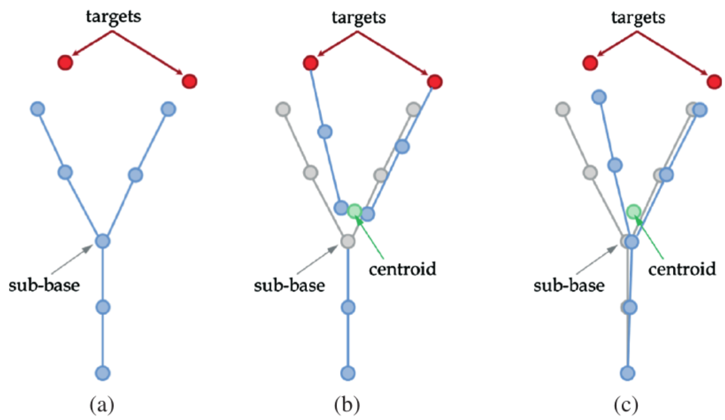

FIGURE 2.10: Image showing, Image Taken from [12]

2.3.6 FABRIK Constraints

- [12] mentioned multiple various constraint types, but lacks in detail on how to exactly implement these, referring to [9].

- While mentioned in [9], Aristidou et al. [12] explained more in detail how to solve a FABRIK armature for multiple endeffectors.

- all chains of the armature are propagated from endeffector until the sub-base joint, which is equivilant term for branch.

Multiple Endeffectors - 2 Stages
- first Normal (forward) - appliy FABRIK starting from each endeffector moving inward - moving until sub-base: - (sub-base == joint with 2 or more child chains) - sub-base is joint that connects two or more joints - -> so apply fabrik until sub-base is reached - on subbase joint, each position of joint on fabrik iter is stored - -> centroid is calculated wich is the mean of all pos - - FABRIK iter contiued from new subbase pos (centroid) - second stage backward: - algo normal applied until sub-base - then applied seperately for each chain

multi endeff cases: - neither reachable - just apply straight line - TODOf optimized version lowers to one iteration - one reachable - 2 solutions: - attain one target, leave other away - both away but closer - TODO option to interpolate with weights - both reachable - 2 cases: - both can reach target in configuration - only one can reach target, again interpolatable with weights

caption

- option to smooth out noisy input data **4.4.2. Joint Control between Two True Joint Positions.**

- true root and end eff pos, noisy inter joints - if out of reach, straight line - if in reach fabrik applied from end eff (first forward) then backward

check source

- also posibility to run into deadlock - strict constraints cause not reaching config because of locality in algo (no parent, child consid) - solution - first check if reachablie: - yes -> if dist not smaller each iter - -> backward step first iter: bend by increasing degrees away from target - allows joints to bend more - bending till 360 degrees, then no solution if not reached
- [12] also showed Self-collision Determination

2.3.7 iTASC

Instantaneous Task Specification and Control (iTASC) [16] - multiple constraints

iTASC blender pos

- it Implemented as an optional Inverse Kinematics Solver in Blender but incomplete, having various issues, highlighting its implementation complexity.

2.4 Motion Retargeting

- Motion Retargeting is the process of transfering motion data for a skeleton with a specific hierarchy and joint lenghts to another skeleton which either differs in one or both.

2.4.1 Naive Retargeting

- The naive retargeting approach is to simply transcribe motion applied to a joint from the source character to a target character by defining joint correspondences between source and target character

list problems

- this approach can cause various problems, including but not limited to ground penetration, self interpenetration, wrong directions due to restpose differences, footsliding and more

TODOm 2.4.1: limb based MoRe, or better name?

2.4.2 Limb based Retargeting

TODOm 2.4.2: category?

[17] - 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.

[18]

2.4.3 Jacobian based

TODOm 2.4.3: is Limb based?

Choi and Ko use Inverse Rate Control, which is the Jacobian Inverse Method of Inverse Kinematics, and extend it to be applicape to tree structures instead of chains without branches. [19]

- Choi and Ko have also showed a way to imitate joint angles of the source motion by incorporating them as a secondary goal. The primary task tracks given end-effector trajectories and the secondary task is to imitate the joint angle trajectory θ, as best as possible.

- Input trajectories are a continuous input of constraints which applied to the target produce coherent motion.

inverse rate control

2.4.4 Machine Learning Approaches

continue

- due to the complexity of the motion retargeting Problem, machine learning approches are a popular...

Aberman et al. [20] - using skeletal pooling, which reduces skeletons to a common primal skeleton by a sequence of edge merging, to archieve retargeting between different skeleton hierarchies

TODOm 2.4.4: list various more MoRe paper?

- Skinned Motion Retargeting with Residual Perception of Motion Semantics & Geometry
- Unsupervised Motion Retargeting for Human-Robot Imitation
- https://arxiv.org/pdf/2402.05115v1

- HMC: Hierarchical Mesh Coarsening for Skeleton-free Motion Retargeting
- https://arxiv.org/pdf/2303.10941v1

- ==Correspondence-Free Online Human Motion Retargeting==
- https://arxiv.org/pdf/2302.00556v3

- OKR: Joint Keypoint Representation for Unsupervised Cross-Domain Motion Retargeting
- https://arxiv.org/pdf/2106.09679v1

- Skinned Motion Retargeting with Dense Geometric Interaction Perception
- https://arxiv.org/pdf/2410.20986v1

- Self-Supervised Motion Retargeting with Safety Guarantee
- https://arxiv.org/pdf/2103.06447v1

- Flow Guided Transformable Bottleneck Networks for Motion Retargeting
- https://arxiv.org/pdf/2106.07771v1

- MoCaNet: Motion Retargeting in-the-wild via Canonicalization Networks
- https://arxiv.org/pdf/2112.10082v2

- Hierarchical Neural Implicit Pose Network for Animation and Motion Retargeting
- https://arxiv.org/pdf/2112.00958v1

TODOm 2.4.5: correct?

- while machine learning approaches can offer good quality retargeting, there is a lack of interactively changing retargeted motion

- new features often require models to be retrained

2.4.5 Other approaches

2.4.6 Available Tools

2.5 Automated Rigging

2.5.1 Machine Learning Approaches

2.5.2 Thinning Approaches

TODOm 2.5.1: genauer anschauen für mögliche impl? (Future?)

2.5.3 Skin Matching Approaches

2.5.4 Re-Meshing

Chapter 3

Motion Retarget Editor

methodisches vorgehen hier

TODOm 3.0.1: goals from related work?

- current research focuses on machine learning - despite ik / limb based methods existing for a long time, there exist no standalone free open source tools or plugins for blender

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

- Noteably, there is a lack of Open Source Implementations of more complex Motion retargeting algorithms and especially frameworks in order to compare and improve motion retargeting.

- Furthermore the process of creating a usable virtual human for various applications remains tedious - goal creating for creating an autonomous virtual human

pos

- TODO MetaHuman (UE5) provides an excellent quality with facial and hand rig - but creation restricted to existing toolset provided by environment - clothing has to be recreated - cant use scan

user interface section?

CrossForge [21], developed by Tom Uhlmann at Chemnitz University of Technology, is a A C/C++ Cross-Platform 3D Visualization Framework using OpenGL. - design allows you to use the available CrossForge modules, modify them, or completely replace them with you own OpenGL based implementation and GLSL Shaders. - This flat design, simplicity and direct approach, CrossForge is well suited for educational purposes and computer graphics research.

formulation

- CrossForge allowes for quick implementation of various ...

- while CrossForge already has LinearBlend Skinning and an simple Animation Controller Implemented, it is lacking in many Features, notebly a User Interface for Keyframe Control, Joint Visualization, a Picking System, which had yet to be implemented and will be discussed in the following sections

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 Animation System

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

ref assimp

3.3.1 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.

3.3.2 Sequencer

- A sequencer is a powerful tool in game engines and animation software used for creating and editing cinematic sequences, for

3.3.3 Editing Tools

In order to

- construct restpose - update restpose - apply transform to Mesh

3.4 Inverse Kinematics Implementation

While various Inverse Kinematics Implementations exist, , they are usually implemented across various Programming Languages or use different 3D Engines, resulting in vastly different and complex APIs.

list impl

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

3.4.1 Jacobian Method

- Various Sources for Jacobian Inverse Kinematics lack in detail on what specific entries of each cell mean.

- this is due to what the input means

3.4.2 CCD

3.4.3 FABRIK

3.4.4 Comparison of IK Methods

requirements for good IK

multiple endeffectors

survey table comparison

3.5 User Interface

For the User Interface Cornut's ImGui [22] is used. It provides a - large and flexible set of Widgets - very easy integration - many plugins written for it

TODOm 3.5.1: Zenodo, to get DOI of github repo? upload others?

3.5.1 Scene Control

In Order to apply transformations to picked objects, a gizmo is needed. The term "gizmo" is typically used to refer to a small device or gadget that has been designed for a specific purpose. It often signifies a tool that is capable of performing a particular task in an innovative or efficient manner. The term is informal and can apply to various types of devices.

In the context of graphics programming, gizmos facilitate the manipulation of objects within 3D space. They are widely used in graphics editors to visually represent and control object transformations, most commonly position, rotation, and scale. However, they also cover various other types, such as camera manipulation or mesh editing. They provide intuitive controls that enhance user interaction with the 3D space.

ext

ImGuizmo [23] is a easy to integrate Gizmo Plugin for ImGui.

3.5.2 Picking

- in order to interact with scene objects, a picking system is needed -
- TODO Matrix seperation

3.6 Motion Retargeting

subsection Combined Retargeting Methodologies (TODO eigenanteil in extra chapter)

- import / export armature

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

- only inital guess, user will be able to check matched joints - TODO visualize joint chains with JointPickable

3.8 Constraints Implementation

section Combined Constraint System (TODO eigenanteil in extra chapter)

3.8.1 Target Weighting

- while not mentioned by Aristidou et al. [12], Target priorities can be archived by lineary interpolating centoids between optimal sub-base position depending on their Weight.

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)

4.1 Editor Improvements

4.2 Blender Addon

4.3 SMPL fitting

4.4 Utilizing Skinning Alternatives

4.5 Clothing

4.6 Motion Blending

4.7 Other Useful Tools

Bibliography

[1] Thomas Kronfeld. "Themenschwerpunkte Informatik: Virtual Humans 3. Kinematik". 2022.

[2] Sébastien Moya and Floren Colloud. "A FAST GEOMETRICALLY-DRIVEN PRIORITIZED INVERSE KINEMATICS SOLVER". In: ().

[3] 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.

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

[5] Thomas Kronfeld. "Themenschwerpunkte Informatik: Virtual Humans 4. Inverse Kinematik". 2022.

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

[7] 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.

[8] 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/.

[9] 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.

[10] Masanori Sekiguchi and Naoyuki Takesue. "Fast and Robust Numerical Method for Inverse Kinematics with Prioritized Multiple Targets for Redundant Robots". In: *Advanced Robotics* 34.16 (Aug. 17, 2020), pp. 1068–1078. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2020.1780151. URL: https://www.tandfonline.com/doi/full/10.1080/01691864.2020.1780151.

[11] 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.

[12] Andreas Aristidou, Yiorgos Chrysanthou, and Joan Lasenby. "Extending FABRIK with Model Constraints". In: *Computer Animation and Virtual Worlds* 27.1 (Jan. 2016), pp. 35–57. ISSN: 1546-4261, 1546-427X. DOI: 10.1002/cav.1630. URL: https://onlinelibrary.wiley.com/doi/10.1002/cav.1630.

[13] Jane Wilhelms and Allen Van Gelder. "Fast and Easy Reach-Cone Joint Limits". In: *Journal of Graphics Tools* 6.2 (Jan. 2001), pp. 27–41. ISSN: 1086-7651. DOI: 10.1080/10867651.2001.10487539. URL: http://www.tandfonline.com/doi/abs/10.1080/10867651.2001.10487539.

[14] Kris Hauser. *Robotic Systems (Draft)*. University of Illinois at Urbana-Champaign. URL: https://motion.cs.illinois.edu/RoboticSystems/InverseKinematics.html.

[15] Chris Hecker. "My Adventure with Inverse Kinematics".

[16] Joris De Schutter et al. "Constraint-Based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty". In: *The International Journal of Robotics Research* 26.5 (May 2007), pp. 433–455. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/027836490707809107. URL: https://journals.sagepub.com/doi/10.1177/027836490707809107.

[17] 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.

[18] 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.

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

[20] Kfir Aberman et al. "Skeleton-Aware Networks for Deep Motion Retargeting". In: *ACM Transactions on Graphics* 39.4 (Aug. 31, 2020). ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3386569.3392462. arXiv: 2005.05732 [cs]. URL: http://arxiv.org/abs/2005.05732.

[21] Tom Uhlmann. *CrossForge: A Cross-Platform 3D Visualization and Animation Framework for Research and Education in Computer Graphics*. 2020. URL: https://github.com/Tachikoma87/CrossForge.

[22] Omar Cornut. *ImGui*. URL: https://github.com/ocornut/imgui.

[23] Cedric Guillemet. *ImGuizmo*. 2016. URL: https://github.com/CedricGuillemet/ImGuizmo.