Skeletal similarity based automatic joint mapping for performance animation

Author: Steven Weijden, 3683591
Supervisor: Dr. Nicolas Pronost

Universiteit Utrecht

March 2014
MSc Game and Media Technology – Utrecht University

Abstract

Motion capture technology is becoming increasingly readily available to end users. To accommodate this development we aim to create an application that can map an as wide array of characters as possible, using minimal assumptions. For the input multiple (non-)human characters can be used to animate a target character. By using only information that can be derived from the skeletons and no prior learning phase, the approach ensures flexibility. After selecting the input and target characters, users can get a decent mapping without changing any options, though if desired settings can be changed to influence the mapping. The mapping subdivides the skeletons into sections with characteristics and uses these when comparing all the bone matches. These comparisons are done using weighted metrics, which check various aspects to determine which matches have the best matching role and function in each skeleton. By taking the combined weight these metrics, the best matching input bone per target bone can be found.

Contents

1. Introduction

We present an application to create an automatic joint mapping which can interpret a wide variety of characters using minimal assumptions. Using various bipedal and quadrupedal characters as input and target, these can be automatically mapped to easily allow users to create a custom character mapping. We aim to combine this mapping with motion capture systems such as Vicon or Kinect. Characters are expected to be symmetrical and have the y-axis as their forward direction. There are no pose restrictions or learning phases, though extreme poses may influence the mapping. For the automated mapping the user won't need to have a detailed understanding of the application, keeping it easy to use and accessible. Because there can be situations where the automated mapping doesn't find the most desirable result, it's also possible to adjust the matches found in the mapping manually. There are also several options included that allow for further customization, such as selecting multiple input bones for a given target bone or setting a delay on an input bone.
The skeletal structure is analyzed and subdivided in order to derive various characteristics, as is described in chapter 3. Besides comparing individual bones, it is explained how data from groups of bones is stored separately to allow input and target sections to be compared. Information on these sections is gathered from their placement within the skeleton and the bones they encompass.
In chapter 4 the mapping is explained and the methods used to evaluate these matches. This is done through metrics and weights, which control how much of an influence a characteristic has. Characteristics being the information stored per bone or section. The match with the highest total weight is considered the best for each target bone. The metrics that are used were chosen to allow minimizing the assumptions while creating accurate and reliable results. For this reason metrics such as name based matching aren't used. Besides creating a dependency on the naming conventions, this metric is also likely to give poor results when matching significantly different characters which have a different number of bones in their body sections.
The experiments are done in chapter 5. An overview of the character models is given and the automated mapping results are looked at. Then the results are analyzed and the risks and limitations are discussed. In chapter 6 the conclusion is given and the encountered problems and future work are covered.

2. Related work

A lot of research has been done in the field of motion retargeting and there are various ways through which people have strived to create the most desirable motions for their goals. Our approach is to create an application that focuses on accepting an as wide variety of inputs as possible with minimal assumptions while generating decent natural results. In this section we look at some of the already existing work that has been done and how this compares to our own work.
In our approach we take a (non-)human character and find the best mapping to other (non-)human characters, using only the skeletal information. There are various approaches that map to non human characters, but these either make more assumptions or require more information to achieve their result. For our approach we restrict our assumptions to having a character be oriented correctly and the skeletal structure conform to a tree graph. The results for the compared approaches are more natural and realistic, as they retain the naturalness of the input motion with possible adaptations or by simulating physics on the resulting motion. For our approach we aim to include physics in the future, because this allows for the most flexible handling of many situations, such as when mapping from a biped to a quadruped character.

- [SSL13] Creature Features: Online motion puppetry for non-human characters

In order to create the desired motion, human input motions are coupled with predefined non-human target motions. These small number of loosely matched human-creature motion pairs together with approximate body tags are required. With this the direct feature mappings and motion classification can be computed. This allows for a user to create real-time custom animations by using the coupled motions. Because the creature motions are predefined and coupled with the human motions, the results are also very natural. The motions are however limited in flexibility to the reach of the human performing the input as well as the coupled motions available.
The work of [SSL13] focuses on driving non-human characters using human motion input. It has more natural movement and detailed control, but at the price extensive preparation and limitations on the flexibility depending on the coupled motions. In the approach it's interesting to see how the Kinect is incorporated to provide streaming input of character motions to control the target and what problems they encountered.

- [FHXS13] Fast, automatic character animation pipelines

This paper aims to allow users to quickly retarget detailed behavior from a human input skeleton to a human target skeleton with a different morphology, but a similar skeletal hierarchy. The transferred behavior consists of gazing, object manipulation, locomotion without foot sliding, head nodding and more. In order to match the skeletons, named based and topological based matching is used. To get the same initial poses on the skeletons, the local coordinate frames are aligned. Controllers are used to generate motion by means of rules, learned models or procedurally based methods.
In [FHXS13] fast, natural, detailed and controlled motions are achieved with retargeting human input motions to a human target skeleton. Constraints are enforced which allow grasping objects and other exact motions for characters with different morphologies. The price of this is that the approach is limited to human skeletons and requirements are placed on the naming of the bones.

- [BTST12] Automatically Rigging Multi-component Characters

For the target character the application requires only a mesh and the skeleton is generated for it. The input skeleton is then mapped with the animation data to the target skeleton. The focus of the paper lies primarily with the identification and creation of the skeleton. For the mapping procedure an approach similar to our own is used which requires the basic assumptions of a tree graph on the skeleton to function and an initial alignment of the target and input characters. Characters may not be in an ideal T-pose and labels are not used either. The structure of the skeleton can differ, such as a different number of bones in the character limbs. In order to find the best matches between bones on a single branch, the length of all bones in a section is compared to the total length of a section. Using the ratio from this, the set of matches with the lowest differentiating ratios for the section are used. This is the same approach we have taken in the option to avoid duplicate bones.
In [BTST12] for the target character only the mesh is needed as the focus of the paper is to rig multi-component characters. The resulting target skeletons could potentially be used as input for our application. This is possible because neither approach requires labeling of bones or limbs. Other similarities are allowing different morphologies between target and input characters, non standard poses for the characters and the handling of body sections with a different number of bones between input and target. Our approach however focuses more on allowing many different types of input and has been built to deal with more significantly different morphologies, a wider variety of input types (including multiple inputs) and to allow user interaction on the resulting mapping.

- [PS12]A novel template-based automatic rigging algorithm for articulated-character animation

Taking only a mesh as input, a skeleton is derived from it and classified based on a database of templates without any user intervention. After finding the best matching template, the joints from the template skeleton are mapped to the skeleton derived from the mesh. The number of junctions on the symmetry axis within the created skeleton is an important piece of information in the template identification process. Using this information the skeleton can already be subdivided into a subset of character types. Depending on the number of branches from a junction, this may be sufficient to identify the character type. Should a character have a head and tail and the pure skeletal information would be insufficient, then the thickness of the mesh at the head and tail is used.
The focus in [PS12] is to get a rigged character from a mesh. Though a very different goal from our own, to identify the skeleton there are techniques then can be used to help identify sections. However because our approach only uses a skeleton as input, we can’t use the mesh thickness. This information is also used to differentiate between the various digits on hands or feet and to find the forward direction dynamically. Our approach uses the forward direction to determine the left and right side of the character and subsequently which sections are digits on limbs.

- [YAH10] Animating Non-Humanoid Characters with Human Motion Data

The goal is to map non human characters using human motion capture data which conveys expressions and emotions through body language. For this reason the focus however lies much more on realism. To improve the physical realism a small set of corresponding key poses is used together with a physics-based optimization process. The resulting body language is expected to be understandable to human viewers.
Similar to our own work the goal in [YAH10] is to map non human characters, though our primary focus lies in having an as diverse approach as possible. Users only observe the results, while in our

approach the naturalness in part depends on the user input. In this work diversity matters, but only subject to the extent this can be kept natural.

- [HKG06] Splicing Upper-Body Actions with Locomotion

The paper presents a method to produce natural motions using upper and lower body splicing while retaining physical and stylistic correlations by identifying and enforcing temporal and spatial relationships. These are found within the example motions to retain the overall posture of the upper-body action while adding secondary motion details appropriate to the timing and configuration of the lower body.
The paper from [HKG06] provides an interesting insight into the possible approaches that can deal with blending of different character inputs. The focus is on creating new motions from separate upper and lower body motions. It is interesting to see how a natural motion was achieved, though physics were not explicitly kept in mind. As the goal of implementing physics in the future is to achieve more realistic motions, this work may help provide insights.

3. Skeletal hierarchy

For the mapping to be done the skeletons need to be analyzed and their underlying similarities and characteristics identified. The skeletons are subdivided into sections of one or more bones, to which certain characteristics can be assigned. After initial grouping based on the joints, the sections that are of the same type and adjacent to each other, are merged. With help of these sections, bones can be compared both individually and as a group.
Assigning the various types of body sections correctly is essential to creating a proper mapping. These types are explained in 3.1 and the information stored with these is described in 3.2. Other important identifiable information is mentioned in 3.3, such as how the continuation of the centerline is recognized, which is later used to differentiate between the spine, head and tail.

3.1.Skeletal body types

Only the root and spine will always exist. The other sections might not be present, while still retaining a validly interpretable skeleton. When a section exists, it contains one or more bones. A bone is defined by having a (starting) position and orientation. The length is defined by the distance to the next bone. If there is no bone lower in the hierarchy then the current, there should be an end effector position present to derive the length from. Presently these aren't yet available however due reasons explained in section 6.

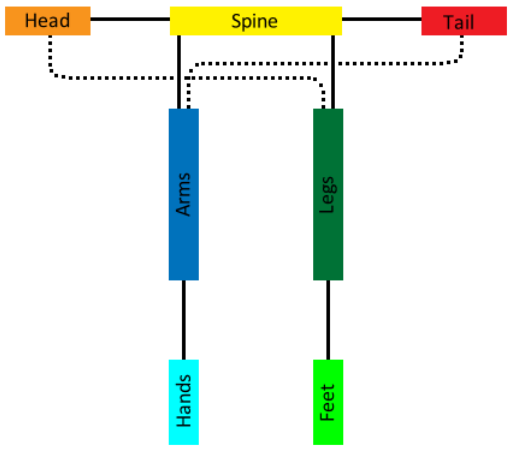


Figure 1: skeletal hierarchy

- **Root**
  This is the origin of the skeleton. Generally part of the spine, it could also be part of the head or tail section as long as it's a section that is part of the center of the skeleton.
  - **Spine**
    The spine is the central section which connects the other sections together. Part of the centerline, it runs from the bone connecting the last pair of legs to the bone connecting the first pair of arms (or possibly legs on a quadruped). The spine will continue beyond points with multiple child nodes, until either it ends or a head or tail section starts. Sections designated as arms, legs, head or tail can be connected to the spine.
    - **Head**
      Connected to the spine, for a biped this section is defined by being ahead of the forward most pair of arms of a skeleton. In this case, if no pair of arms exists, no head section is

defined. For a character with multiple pairs of legs, the forward most pair of arms or legs is used to define the start of the head section. It should be noted that based on the way the head section is created, it's possible for a pair of legs to be attached to this section, though this is unlikely.

- **Tail**
  Connected to the spine, this section is defined by being behind of the rear most pair of legs of a skeleton. If no pair of legs exists, no tail section is defined. It should be noted that based on the way this section is created, it is possible for a pair of arms to be attached to this section. This can happen in the event the tail has extremities such as fins.
- **Arms**
  Connected to the spine or tail, arms are defined as a pair of sections split off from the centerline and their last bones ending above the threshold for designating legs. Arms continue until either the end of the section if there aren't any further junctions, until a junction is hit which contains multiple children with their own junctions or a junction is hit which doesn't have any junctions. Any sections after arms will be hands.
  - **Hands**
    Connected to arms, hands are the end sections below arms. Hand sections most often exist at the end of arms, but if an arm has multiple junctions, it's possible to have hand sections on other locations attached to an arm. This mainly occurs with wings recognized as arms.
- **Legs**
  Connected to the spine or head, legs are defined as a pair of sections split off from the centerline and their last bones ending below the threshold for designating legs. Legs continue until either the end of the section if there aren't any further junctions, until a junction is hit which contains multiple children with their own junctions or a junction is hit which doesn't have any junctions. Any sections after legs will be feet.
  - **Feet**
    Connected to legs, feet are the end sections below legs. Leg sections generally exist at the end of legs, but should a leg have multiple junctions, it's possible to have feet sections on other locations attached to a leg.

3.2.Section characteristics

Various characteristics are used in the mapping to help differentiate bones and groups of bones from each other. These play an important role in the metrics, which are explained in 4.1. Here we display what information is stored in them.

- Body sides

Defining a skeleton into 3 groups, all body sections are assigned to left, center or right. This is an important characteristic for the metrics.

- Body types

This gives a body section the label of arms, legs, hands, feet, spine, head or tail. This is an important characteristic for the metrics.

- Number of bones

This value stores the number of bones in a body section and is used to for traversing body sections.

- Length bones

The combined length of all the bones in a body section is stored here. This is used for the removal of duplicate bones described in 4.2.2.

- List of bones

The bones inside the body section are listed in hierarchical order. This allows traversing the skeleton by jumping to the last bone in the list to access any child sections or taking the parent of the first bone to navigate towards the root.

3.3.Hierarchy details

Continuation of the centerline

In order to find the child bone which continues the centerline of the skeleton, the angles between all child bones on a junction are calculated and compared as shown in figure 2. The bone which has the two most equal angles is then considered to be the continuation bone. Any pair of remaining bones will then be limbs. There is however a small chance that the angles between all children are almost identical and a bad result is found. Alternatives are not necessarily better though, so we've stuck to this approach.

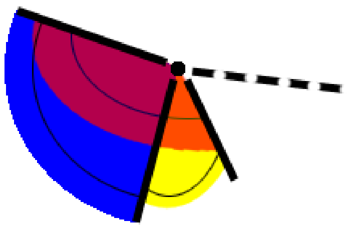


Figure 2: spine continuation

An alternative would be to check the hierarchy of each limb and check the number of bones within, but this would be unreliable as there is a significant chance the continuation bone might have the same number of bones as the other sections. Another approach could be to use the forward direction to compare the bone angles with, but this has a small chance of bad results if the child bones are aligned in the same plane as the forward direction has with the vertical axis. It's also possible to look at the position of the end points of the child bones and check how far along the forward direction these lie. This should be more reliable, but without end effectors this approach also isn't applicable to limbs of length one, as no end point can be derived. As such we've stuck to comparing the angles.

Leg threshold

To differentiate between arm and leg sections, we look at the last bone in a limb. If the position of this bone is above this threshold, then it is considered an arm, otherwise it is set as a leg. The leg threshold can be dynamically changed through in UI.

Assumptions

The application is intended to work with minimal requirements. The only assumptions made are that a model's forward direction is along the y-axis, a skeleton adheres to tree graph and that a skeleton is symmetrical. This last assumption was made to streamline development and should be able to be removed with minor changes in the code.

Tests have been done for biped and quadruped characters, though the mapping has been built to deal with an arbitrary number of limb pairs. This is aided by the joining of sections of the same type during the mapping. This causes each skeleton to at most have one spine, head and tail. A skeleton will always have a spine. A head section is defined by the forward most arms for a biped character or the forward most arms or legs for a multi legged character. The tail is defined by the rear most legs. If the elements required to define a section don’t exist, the mapping will be considered to not have such a section.

4. Mapping

In this section we describe how a user interacts with the application in order to animate the target character and the inner workings of the mapping. The mapping consists of comparing all target and input bones and it retains those that are considered the best matches based on the given criteria. The metrics are first described in 4.1 and are what determine how good a target-input match is. In 4.2 options that make up the criteria are described. Various options affect the mapping before, during and afterwards, while In 4.3 we go into more detail about what makes a match good and which steps are taken to achieve this result.
The user will not need to have any in depth knowledge on how the mapping works and still be capable of getting a decently mapped result. Besides the automated process and the options available to modify this, the mapping can also be manually changed at any time. This allows users to generate their own motion driven character animation for (non-)humanoid characters.

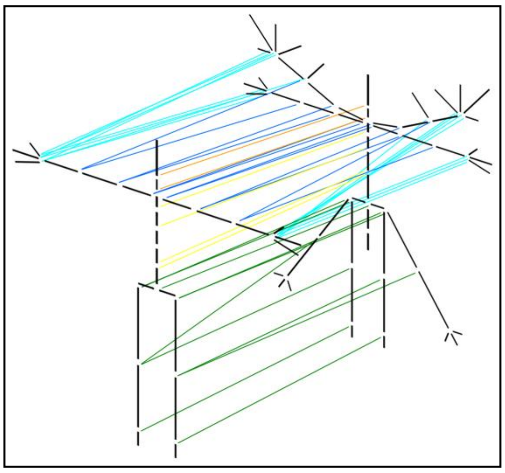


Figure 3: Mapping example

4.1.Weighting metrics

Matches between input and target bones are given points based on how well they conform to the metrics. This gives a total amount of points for each match, of which the highest is considered the best. More points are given based on how significant and reliable metrics are, as shown in table 1. Several of the metrics also use skeletal characteristics as described in 3.2. When metrics apply to body sections as a whole, this means all bones in that section have the same weight added. Other metrics increment weights based on characteristics of individual bones. A match is considered good if the resulting weight is above the mapping threshold.

Table 1: *Assessment of metrics*

Metrics	Importance	Default Weight
Same body side	++ (high)	30
Same body type	++(high)	50
Similar body type	++(high)	15
Number of bones	+/- (slight)	10
Bone orientations	+/-(slight)	15*1 (max)
Bone sequence	++(high)	16
Source continuity	+ (moderate)	5
Source preference	+ (moderate)	10 (optional)

- Same body side

By comparing the stored body side for sections, points can be given for matches on the same side. The metric works per section of a skeleton and when multiple children are encountered from the root down, it is checked if there is a bone that continues the spine. With an uneven number, one child will continue the spine and is assigned as part of the center. The remaining limbs pairs are compared with the forward direction to see if they are on the left or right side. An example of the subdivision this metric creates can be seen on the left of figure 4.
This metric is considered highly valuable because it allows stimulating matches that create good matches and is applied without the risk of false positives. If limbs exist on the same side for both target and input skeletons, then they will be given preference over matches on different sides. A large weight (of 30) is given so that this metric will exceed less reliable metrics.

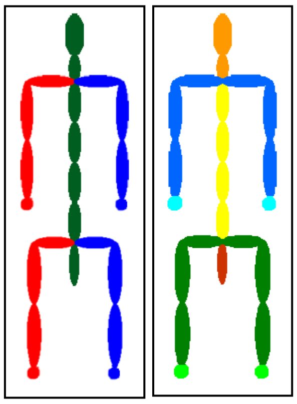


Figure 4: (left) body sections
(right) body types

- Same body type

When comparing the body types of the various sections, this metric gives points to identical type matches. The weight (of 50) is very large, so that when both target and input skeletons have the same type this will be the most significant metric. Because it’s possible for a type not to have a matching counterpart, the similar body type was added which is explained next. An overview of the body types on a humanoid character is given on the right in figure 4.
This metric is considered highly valuable because it plays an important role in getting good matches and has no risk of false positives.

- Similar body type

When comparing the body types of the various sections, this metric controls giving points to similar matches. The weight (of 15) is large enough to be significant in combination with most metrics, but small enough not to interfere with more important metrics such as the same body type matches.
This metric is considered highly valuable because it plays an important role in getting good matches when a section doesn’t have an identically matching type and it has no risk of false positives.

- Number of bones

When comparing sections of target and input skeletons, this metrics compares the number of bones each section has. If this equal a weight (of 10) is added. This is large enough to be noticeable in relation to the less significant metrics such as the source continuity, but small enough to avoid disrupting the similar body type metrics.

This metric is considered only slightly valuable because there is a chance that sections which aren't a good match are also given points. It is even possible if a good match on another skeleton has a deviating number of bones in a section that points are only given to a bad match. The metrics is still included however, because if all other factors are equal, it is preferable that sections of equal length are matched.

- Bone orientations

In this metric the orientations between bones of target and input skeletons are compared. The weight a match is given depends on how similar the two orientations are. The maximum weight (of 15) is given if the difference in orientation is zero. The maximum angle is equal to the maximum weight. Towards the maximum angle, the weight quadratically decreases to zero. The total amount of points assigned can be modified with a separate value.
This metric is considered only slightly valuable because it's possible for matches in orientation to occur between bones which aren't a good match and between bones that are an intended match it's also possible that they deviate more than the maximum angle.

- Bone sequence

By comparing if bones are at the same location within a body section, this metric increases the likelihood that bones are assigned in the same order in the target skeleton as they appear in the input skeleton. A noticeable weight (of 16) is given so that matches are robust against the orientation weight. Because this metric is always applied to a match within a body section, it won't conflict with metrics like the source continuity.
This metric is considered highly valuable because it is the most important metric to stimulate matches to follow the correct order within a section and thus contributes a lot to creating a good mapping. Furthermore the chances of less desirable matches are very low. When target and input sections are of a different length, this metric only influences those bones up to the smallest compared section. To deal with this eventuality, it's also possible to select an option to remove duplicate bones, which is explained in 4.2.2.

- Source continuity

When using multiple input skeletons it is preferable to map target bones within a section from the same input. This metrics exists to avoid target bones in a section from being matched randomly with bones from similar input skeletons, due to a lack of differences between the input sections. In this case the orientation would be the only variable determining the eventual match. The metric checks if the source of the previous match is the same as the current. By giving a small weight (of 5), the orientation difference must be greater than this threshold in order for a consecutive match to be chosen from a different skeleton.
This metric is considered moderate valuable, because the low weight allows it to avoid disrupting more important metrics, while at the same time still making a noticeable difference in the final mapping. Besides this, the metric also improves maintaining the sequence of the input by adding points if the parent of the bone being checked is the previous best match.

- User input source preference

When multiple input skeletons are used that are similar, it's likely that the automated mapping can't differentiate properly between them. This metric was made to allow the user to give a preference to a source. By default no weight is given. However if the default mapping is undesirable and requires manual tweaking, this metric can be used to give an additional weight (of 10) to matches from a specific source. This can be used to reduce the number of bones that need to be manually tweaked. This metric is considered moderate valuable, because in the worst case it doesn't influence the mapping and in the best it allows using the automated mapping to reduce work.

4.2.User options

The options available to users can be split into three sections that influence the mapping. The main interface available to a user is explained in 4.2.1, consisting of the bone overview, the source loading and the animation controls. Options related to the mapping, called the global options, are discussed in 4.2.2. The global options affect the mapping on a larger scale, by changing how parts of the mapping are interpreted. The bone specific options are explained in 4.2.3 and can modify how an input bone affects the matched target. These are applied separately from the rest of the mapping, by manually setting them in the bone overview.

4.2.1. Basic options

This part of the interface is what is minimally needed to use the application and create a mapping. The interface can be subdivided into several sections: the source loading, the animation controls, the bone mapping and the bone overview.

- Source loading

In order to get started, an input needs to be selected. There can be up to three sources loaded to combine multiple character inputs. This could be extended to more, however due to limited space on the UI and a lack of necessity this hasn't been implemented. Currently the only implemented method of loading files uses BVH files. Another intended approach was to use a generated skeleton file together with streaming input.

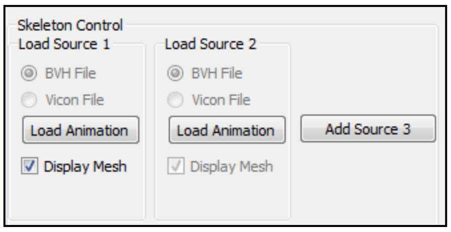


Figure 5: Source loading

- Animation controls

Here the user can load the initial target character. After the bone overview is displayed, the loading and saving of the mapping is enabled. Saved data can be used to load similar skeletons, though the data is loaded in the same order as it was saved. When the current mapping is applied, the options to start/pause and restart the animation are enabled. This allows a user to directly test the automated mapping and adjust it at runtime.

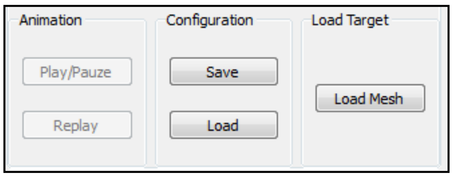
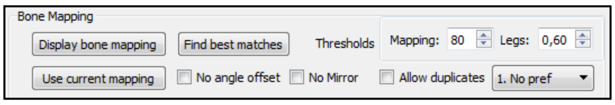


Figure 6: Animation controls

- Bone mapping

Having selected the input and target characters, the user can display the bone overview, change global options, further described in 4.2.2, create an automated mapping and apply the mapping.



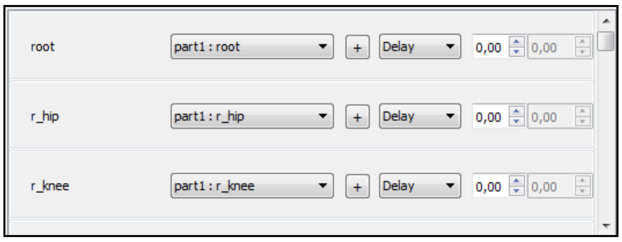
Thresholds Mapping: 80 Legs: 0,60

No angle offset No Mirror Allow duplicates 1. No pref

Figure 7: Bone mapping

- Bone overview

When displaying the bone mapping, a list of target bones is shown with each a list of possible inputs and bone specific options, the last of which is described in more detail in 4.2.3. Additional input bones can be dynamically added or removed. When using multiple input bones on a target bone, alternative bone specific options become available.



root	part1 : root	+	Delay	0,00	0,00
r_hip	part1 : r_hip	+	Delay	0,00	0,00
r_knee	part1 : r_knee	+	Delay	0,00	0,00

Figure 8: Bone overview

4.2.2. Global options

The function of global options is to allow users to change how certain aspects are interpreted by the automated mapping and can be seen in figure 7. The core principle behind the global options is to provide functionality which allows for alternative interpretations and allow a user to judge whether or not this is more desirable.

Table 2: Assessment of global options

Option	Mirror setting	Duplicate bone removal	Mapping threshold	Weight value list
Importance	+ (moderate)	++ (high)	++ (high)	+/- (slight)

- Mirror setting

This option flips the left/right designation of every second pair of legs. This is done to allow using the automated mapping to create a basic gait for multi legged creatures. The weights of the mapping aren't changed.

This option is considered moderately valuable because there are no downsides to it and provides additional functionality. The reason it's not valued more is because it's only relevant in specific cases and requires user observation to verify if the results are more desirable.

- Duplicate bone removal

The function of this option is to improve the mapping when input and target sections have a different number of bones. If there are more target bones, multiple input bones can be matched to bones in a target section, which may lead to exaggerated motions. If there are more input bones, then those input bones beyond the size of the target section are ignored. To remove duplicate bones in the mapping or get the best matches from the entire limb, the ratio of the bone length to the section it's in is taken, as shown in table 3. This ratio is combined for all possible matches that maintain the proper sequence of bones. The resulting set of matches with the best matching ratio's overall is then chosen as shown in table 4. The corresponding input bones are applied as the new matches for the target bones.

Table 3: Example ratio of bones with respect to limbs

Limb bones	A_S	B_S	C_S
1_L	0.2	X	X
2_L	0.4	0.3	X
3_L	X	0.1	0.2
4_L	X	X	0.3

Table 4: Combined weights of sets

Set	Combined weight
1A2B3C	0.7
1A2B4C	0.8
1A3B4C	0.6
2A3B4C	0.8

When there are more target bones, those with the worst ratios will not get an input mapped. Figure 9 shows how the bones between limbs would be mapped for the given example. There are two problems that can occur however. The first is that the mapping works per bone, so there is no guarantee that the bones mapped to a target section all belong to the same input section. Because of this, it is checked how many of the mapped bones belong to the input section which the first bone belongs to. If this is more than half, then this section will be used as input for all target bones. The second problem occurs due to the lack of end effectors, which is explained in more detail in section 5.3.1, Risks and limitations.
The option is considered highly valuable because it provides a significantly different mapping result and is applicable to any existing mapping.

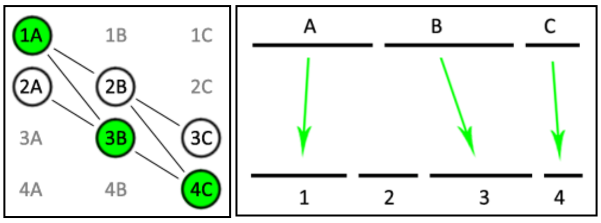


Figure 9: Example match

- Mapping threshold

Whether a match is used is determined by checking if the total weight of the best match exceeds the mapping threshold. The default threshold is set so that the sections with matching body type are all mapped, but those with a similar body type or worse aren't mapped. The function of this option is to allow users to dynamically change the threshold to allow more or less strict matches.
This option is considered highly valuable because it allows users to easily create various mapping results by changing a single variable.

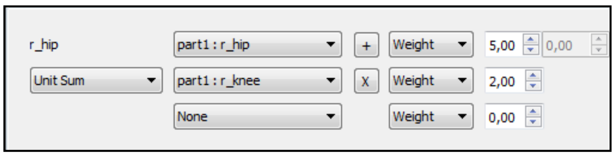
- Weight value list

This option allows a user to choose between various sets of weights for the metrics. Depending on the implemented sets, this can be used to tweak the weights to better match various circumstances. In the current implementation this is only used to allow a preference when using multiple input characters, as the default weight set has proven robust for the various tested characters.
This option is considered slightly valuable because it currently is only useful in a specific situation and depends on predefined weight sets. There is potential for having this option play a more significant

role when creating various weighting sets can have a significant impact, at which point it could become more valuable.

4.2.3. Bone specific options

The automated mapping looks for the best single input to a target bone, but it is possible for the user to set multiple inputs manually. For each input there are several options. In order to show how the application determines what is considered the best result for each target bone and how it is defined, we present an overview of both the one to one mapping options and many to one options available to influence matches between input and target bones. For each input bone the currently selected options and data are stored.



r_hip	part1 : r_hip	+	Weight	5,00	0,00
Unit Sum	part1 : r_knee	X	Weight	2,00	
	None		Weight	0,00	

Figure 10: Multiple input bones

Single input bone

- Delay

Set the time in seconds for the animation of this bone to be delayed.

- Reverse

When rotating the bone, the angle is applied in opposite direction

Multiple input bones

- Weights

When this option is selected, the value assigned to each input bone determines the relative influence it will have on the target bone. If all are 0, then the influence is distributed equally.

4.3. Best matches

For a user it may or may not be easy to identify what would be the most desirable outcome of a mapping. This however depends on many insights that are difficult to translate into metrics and calculations. We try not only to get as close as possible to the most ideal mapping, but also to allow users to steer the automated mapping to improve the result.

Although how good matches are is eventually based upon user interpretation, within the application we consider good matches by comparing each target skeleton bone to all input bones and applying weights to the metrics. Generally the metrics for assigning the sides of the skeleton a bone is part of and the type of section it belongs to have the largest influence in finding the best matches. Combined with the metric to ensure the sequence, these three metrics are the core of the mapping. By varying the threshold for acknowledging a match, a wider selection of bones can be mapped that may have been excluded at first. This however increases the chance for less desirable matches to occur as well.

If the highest weighted match for a target bone is above the threshold, it is considered the best result for that target bone. Is the result lower than the threshold however, two outcomes are possible. Either the target bone is given no mapping when there are still animated bones below the current. Otherwise it's set to be passively animated by physics. Should the removal of duplicate bones option be selected, then it's possible for the resulting matches to still be changed. This option has been described in more detail in 4.2.2.

5. Experiments

Using the current metrics and the default weights, there's been no need to create alternative sets to deal with different skeleton types, as the mapping is robust against the various skeletal structures. The only reason there are multiple sets present at the moment is to display the functionality and to vary the interpretation when dealing with multiple input characters. The most significant variation in results is achieved through changing the mapping threshold, which can be done via the UI. An overview of the various weighting sets is given in table 5.

Table 5: weight sets

Weight Set	Same section	Same type	Similar type	Number of bones	Rotations	Sequence of bones	Source continuity	Source preference
Default	30	50	15	10	15*1	16	5	0
Prefer 1	30	50	15	10	15*1	16	5	10 (source 1)
Prefer 2	30	50	15	10	15*1	16	5	10 (source 2)

5.1.Models

For the testing of the mapping several models have been used. In this section the various models and their characteristics are described. With the method through which models are loaded (Blender to Ogre), it should be noted that no end effectors are stored in the skeleton files. As such bones at the end of a chain do not have a derivable direction vector. This is also the reason why it's impossible to derive the forward direction from the positions of the feet, even if there was the assumption for these to be present. This is further explained in section 5.3.1, Risks and limitations.
An overview of the various models that have been used during testing is shown in table 6, with the number and type of sections as they are recognized by the automated mapping. Below a basic description of the original structure of the models is given.

Table 6: Structure of 3D models

Model / Section	Human (basic)	Human (complex)	Wyvern	Humanoid (Hawkgirl)	Humanoid (Diablo)	Horse	Dragon
Spine	1	1	1	1	1	1	1
Head	1	1	1	1	1	1	1
Tail	0	0	1	0	1	1	1
Arms	1	1	1	2	2	0	5
Hands	0	1	1	0	0	0	1
Legs	1	1	1	1	1	2	2
Feet	0	0	0	0	0	0	1

a. Human (basic) “Stanley_Cubic_Connected”
This is a biped skeleton with a head, one pair of arms and legs. It has no digits.

b. Human (complex) “Rose”
This is a biped skeleton with a head, one pair of arms and legs, including hand digits.

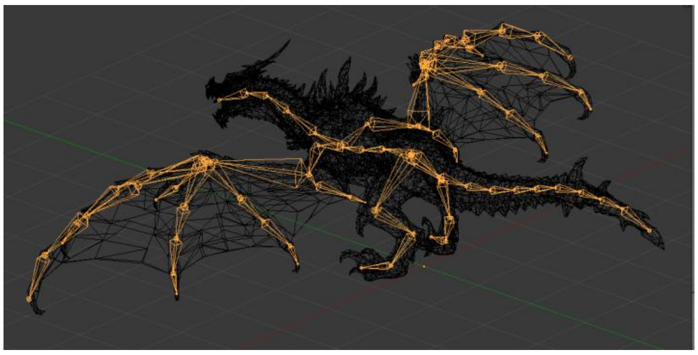


Figure 11: Alduin model

c. Wyvern “Alduin”

This is a biped skeleton with a head, one pair of wings and legs. The wings are recognized as arms. The model also has digits on various positions on the wings and has a tail as shown in figure 11.

d. Humanoid “Hawkgirl”

This is a biped skeleton with a head, one pair of wings, arms and legs. Therefore two arms are recognized. The arms are connected at the same place to spine. It has no digits.

e. Humanoid “Diablo”

This biped skeleton has a head, two pairs of arms, one pair of legs and a tail. The arms are connected to the spine at different positions. It has no digits as is shown in figure 12.

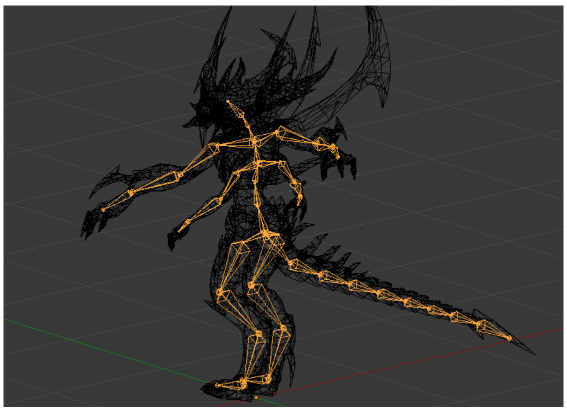


Figure 12: Diablo model

f. Horse “Horse”

This is a quadruped skeleton which has a head, two pairs of legs and a tail. It has no digits.

g. Dragon “Nightfury”

This is a quadruped skeleton which has a head, two pairs of legs, one pair of wings, 4 fins and a tail. The wings and fins are all considered arms. The fins are connected to the tail, with one pair located behind the rear legs, the other 3 sequentially after each other near the end of the tail.

5.2.Testing

All input animations use the human “Stanley_Cubic_Connected” model. This is done both for consistency as well as due to the available input animations, though for the mapping any character used for the output could have been used for the input as well. As input a walking animation is selected, though this doesn’t influence the mapping results. Because the mapping doesn’t incorporate animation data, the focus for the experiments is on using different target skeletons. The main focus lies on one to one mappings, which are shown in 5.2.1.

There is a basic example of a two to one mapping given in 5.2.2 to display the functionality and explain how this is dealt with. However these will more often require manual adjustments, as matches use only the skeletal structure and this leaves a large chance the resulting match won’t be the most desired outcome. To differentiate the second mapped input motion with the first, the second input uses a reaching animation. The results of each mapping can be found in the appendix.

Mapping duration

To give an indication of the time it takes for each mapping to be created a time table is included. To put the values in perspective, the number of bones per skeleton are included as well as the average time per bone. The tests were done with the default weights and with the option for “No duplicates” selected. While testing the same skeleton repeatedly, without running any other applications, random variations in the time were noticed with a size of around 0.06 seconds.

Table 7: Mapping duration and speed per bone for various skeletons

Target skeleton	Bones	Duration (s)	Time per bone (s)
Human “Stanley_Cubic_Connected”	27	0.468	0.017
Human “Rose”	71	1.139	0.016
Wyvern “Alduin”	74	1.186	0.016
Humanoid “Hawkgirl”	42	0.577	0.014
Humanoid “Diablo”	33	0.702	0.021
Horse “Horse”	32	0.546	0.017
Dragon “Nightfury”	187	3.136	0.017

5.2.1. One to one mapping

a. Human (basic) “Stanley_Cubic_Connected”

Both input and target have the same skeleton. All sections and bones are mapped identically.

b. Human (complex) “Rose”

The skeleton has more detail than the input, having hand digits and eye bones that are mapped as arms due to their location. With the location of the eye bones, the spine continues up location where they are connected to the spine. As such the spine section is longer than the input and the neck shorter. By default the hands will not be mapped, though if the mapping threshold is set low enough, they will be mapped using arms from the input as these are designated as a similar body type in the hierarchy. The rest of the skeleton is the same as the basic human skeleton and mapped the same.

c. Wyvern “Alduin”

The wings of the skeleton are recognized as arms, with the digits being considered hands. The spine, head, arms and legs are mapped by their counterparts from the input. The tail doesn’t have matching type, so only if the threshold is low enough will bones from the head or spine be mapped here.

Though visually very different from the human model, the skeletal hierarchy still resembles that of the human model. Significant differences are the presence of digits halfway on the wings and the tail. Furthermore the number of bones and the size of the bones are different.

d. Humanoid “Hawkgirl”

Two arms are recognized by the mapping, one of them being wings. Both pairs are mapped using the same input arms. The spine, head and legs are mapped using their input counterparts. The noticeable differences to the basic skeleton are the extra pair of arms and having 2 bones less in the spine. The size of the bones also differs.

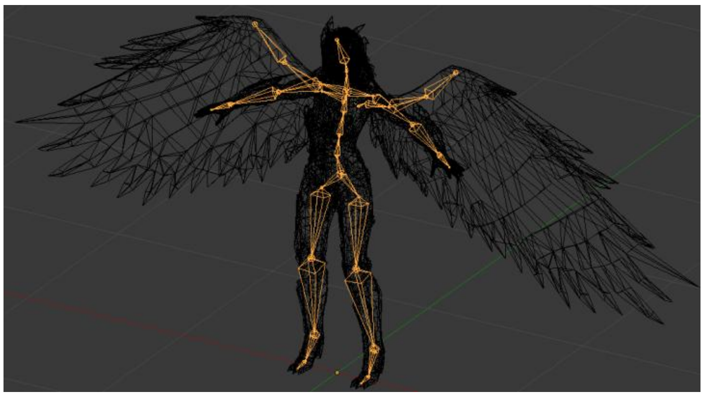


Figure 13: Hawkgirl model

e. Humanoid “Diablo”

Two pair of arms, a pair of legs, a head and a tail are recognized. Besides the tail these are all mapped exactly by their input counterparts, save for the spine which has 3 bones less.

f. Horse “Horse”

Two pairs of legs and a spine, head and tail are recognized. The legs are mapped using the input pair of legs. The head and spine are mapped using their input counterparts. As a quadruped character, the ‘mirror’ option can be used here to get the second pair of legs to be animated with the left and right sides flipped to approximate a basic gait.

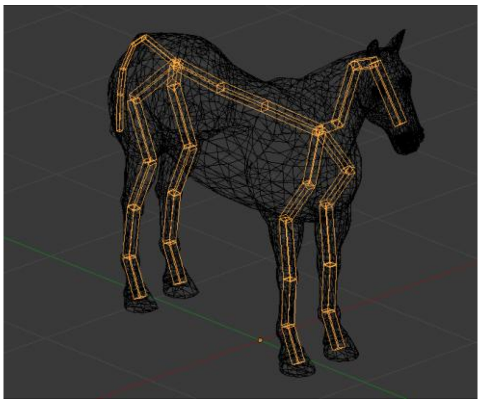


Figure 14: Horse model

g. Dragon "Nightfury"

Two pairs of legs are mapping using the input legs, of which the second pair can be flipped using the 'mirror' option. Five pairs of arms are recognized with various sizes, which are mapped using the input arms. Due to the differences in size, not all input bones can be mapped to target bones in the limbs. The head section includes a joint at the end for the skull and jaw, which in the current mapping isn't anticipated. This doesn't disrupt the assigning of the head type, but can create odd results as it's sequentially handled. More on kind of limitation is described in 5.3.1, Risks and limitations.

The tail and spine are both longer than their input counterparts, so these contain large sections of unmapped bones with the option selected to avoid duplicate bones. Digits on the wings and legs are recognized as hands and feet respectively, but not mapped with the default mapping threshold. As a quadruped character, the 'mirror' option can be used to get the second pair of legs to be animated with the left and right sides flipped to approximate a basic gait.

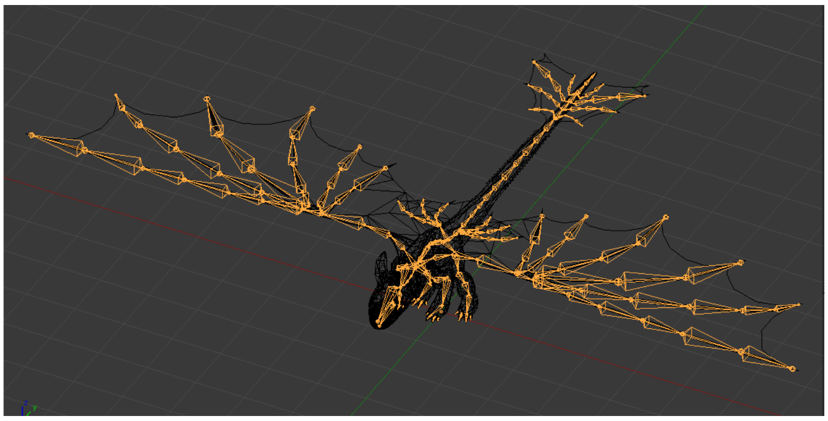


Figure 15: Dragon model

5.2.2. Two to one mapping

h. Human (basic) + Human (basic)

For the second input a reaching animation is selected. Because currently the available input animation data uses the same bones as those of a basic human skeleton, this model is also used for this input as well.

Due to time constraints it was chosen to not edit or create animations to test this in more detail, as the two to one mapping is not significant enough to warrant this effort. Using the same skeleton twice greatly reduces the effectiveness of this test, but it still allows for displaying how this works.

The results of the mapping are as would be expected. The sections are mapped by their counterparts, but due to the lack of difference in the skeletal structure, each section effectively has a 50% chance to come from one source. The source continuity metric does make it likely that the bones within a section are from a single source.

To deal slightly better with the situation where there are two input characters with identical sections in the skeleton, it's possible to set a preference for an input source. This is done by selecting the appropriate weight set.

5.3.Analysis

The core of the mapping consists of properly assigning the body type (i.e. arms, legs, etc) to the different sections in a skeleton. By determining what any given section is, it becomes easier to ensure these are mapped to one another. Together with designating sections to left, middle or right, this allows for robust mapping between sections. In order to get good results within these sections, bones are checked to be sequential and optionally have their length ratio's compared between the selected sections, to further improve the results.

5.3.1. Risks and limitations

During the development choices were made that in some cases brought risks and limitations with them. The most significant ones are described here.

- *End effectors*

For bones in the skeleton, only the starting position is stored. This generally doesn't cause any problems. When retrieving the bone length or direction, the child bone is used to get the result. Not all bones have child bones and normally for this reason end effectors can be used. It turned out however that the end effector positions aren't stored in all situations. When using characters imported from Blender to Ogre, no end effector data is stored and for the already available skeletons the feet end effectors are zero. This meant that solving this problem would not only require changing how the loaded was data, but also creating brand new data. Because of the amount of time and effort this would require, solving this problem was deemed outside of the scope of the current project.
Should this problem be solved in the future, then it could be worth it to derive the forward direction from the feet orientations. This would require the assumption that feet are always present and not oriented completely vertical. The advantage would then of course be that the requirement of characters models having to be oriented facing the y-axis could be dropped.

- *Chance of bad head section designation*

The head section is defined by checking the forward most pair of arms using the forward direction. With only this implemented however it's possible to get a bad mapping under specific circumstances. For example when a character has two pair of arms, connected to different places on the spine and the uppermost pair of arms is behind the second pair, then currently the position of the second pair would be used to determine the head section. It's possible to deal with this, for example by checking if the continuation of the spine has any arms after the current. Should this be implemented however, then it's probably best to also deal with other sections that currently are still be designated as arms, such as when eyes or ears are rigged.

- *Complex sections*

In the current application it's possible for a head, hands or feet section to be given complex hierarchies. This means that the section contains junctions with multiple children, rather than the otherwise linearly structured sections. A complex section for hands and feet occurs when the section designation of the arms or legs ends. The arm or leg section only continues when there is only a single child that can have junctions. If multiple do, then all subsequent the children are set as hands or feet.

For a head section it works slightly different. It is defined based on the last junction with uneven children, where the pairs form limbs. This leaves the head section with the possibility for a junction with an even number of children. An example of this is the dragon character where the head splits at the end between a jaw and skull bone.
These degenerate cases where currently considered outside the scope of the project. A possible approach to deal with these situations could be to take a relatively simple approach to map children running parallel with the same input bone. This would however also need to be adapted to allow it to work together with the removal of duplicate bones option.

6. Conclusion & Future work

Overall the automated mapping generates a decent result quickly and easily. The application in the current form generates a decent mapping and is robust under a wide variety of skeletons. However the naturalness of the resulting motions is still subject to how well the input and target characters match. In order to progress from a dynamic bone mapping to a more versatile and practical approach, the main points to focus on will be to incorporate a physics simulation and streaming input. This will allow users to not only create a fast mapping easily, but also to use it for creating user driven character animations that can be used outside of a testing environment. These and other subjects that are considered points of interest for future work are described next. The main reason for these not being implemented are either due to time constraints or because they were considered outside of the scope of the current project.

- *Physics simulation*

Even though at the moment the physics simulation is outside of the scope of the project, it's been kept in mind during the development that they can be incorporated at a later time. The most noticeable work that was done to allow an easier integration consists of assigning bone sections to be passively animated. Though this currently makes no difference, it will allow the physics simulation to use information from the automatic mapping without having to change it.
The idea for how physics would be simulated and make use of controllers is that for each frame the target characters' pose is subject to gravity. If the generated pose is unbalanced, a smoothed force would then be added on the most relevant bones in order to approximate the original target pose.

- *Streaming input*

For the streaming input the work done to allow easier integration consists of despite currently only using BVH files, which contain animation data, only skeleton information is used. By ignoring the animation data in BVH files the application can give the same result regardless of the input. This allows the development to focus on creating a broad solution, which is also the original goal. To allow this functionality to be incorporated without having to change the UI, the option to select streaming input is already added. The system that's primarily considered for testing is the Vicon system, as it is most likely to give a complete skeleton as a result. Another interesting system capable of streaming is the Kinect, though it's more likely to have problems with occlusion and losing data. In either case the ability to extrapolate motions will need to be implemented.

- *Asymmetrical skeletons*

The goal from the beginning was to have an as flexible as possible automatic mapping, however it soon became clear that at least some assumptions on the input were needed if a workable solution was to be found. In order to build a stable mapping the assumption was made to use symmetrical characters. Though at the current stage it's possible to incorporate asymmetrical characters, this would raise other problems. The most obvious being that there still need to be assumptions to what extent a skeleton may be asymmetrical, before losing the ability to derive anything useful from its structure. Loose sections connected to limbs won't be a problem, having asymmetrical bones on the spine however can currently cause the spine to continue in the wrong direction, if at all. For these reasons it's considered that support for asymmetrical characters should only be added based on requirements for other functionalities such as the streaming input.

- *Offset*

It was intended to align bone orientations between the mapped target and input bones. The option was made to allow for compensating a skeleton which had a default pose where the local axis weren't properly aligned. Because most the used character models are decently oriented, this also wasn't a priority to solve. In order to make it as dynamic and robust as possible, it was only applied to legs. After working on the subject proved to be more complex and time consuming than could be justified, the offset option was left out however.
One of the steps in importing skeleton data was to fix the alignment of the local axis. However the consequence of this step was that all orientation data was set as to unit quaternions and mean no orientation data could be derived from the quaternions. This meant that when trying to apply the offset rotation, the orientation first needed to be calculated using the position. The roll angle couldn't be derived however and it is expected this is a major reason why the expectations didn't match the resulting motions.

- *Expanding supported character diversity*

Currently the only tested characters are biped and quadruped characters. Besides checking if the expected dynamic implementation indeed holds up for more complex models, it may also be important to check how well the application works with simpler characters. This is especially relevant if incomplete skeletons will be incorporated in the future.

- *Expanding type definitions*

To have a more detailed mapping, extra type definitions could be added. To recognize wings, arms can be checked whether they have digits before the last bone of the limb. If they do, then it can be presume the type should be set to wings. This is interesting since it's unlikely such limbs should be mapped from arms when there is a more similar match available. To avoid bad matches when no such type exists on both target and input, wings and arms could be given a higher similarity rating than for example arms with hands. Incorporating different degrees of similarity types would also increase the flexibility and influence of the mapping threshold.
Other types that could be added would be digits connected to the head or tail, such as eyes, ears or fins. If there wouldn't be a matching type to map to, these might be given a moderate preference to hands and a small preference with arms, to allow such sections to be mapped with the most intuitive and dexterous match available.

- *Expanding source continuity*

Currently the source continuity metric only stimulates mapping within one section from the same source. It can be interesting to expand this to ensure that limb pairs are from the same source as well. This can be easily implemented as the tracking of limb pairs is already incorporated.

- *Knee reversal detection*

It might be interesting to add a selectable option to automatically detect if a knee on a leg needs to be reversed. This could be done by using the forward direction and checking the knee joint position relative to the direction of the surrounding bones, though this would require the leg to be bent. A good approach therefore would be to allow manually checking whether reverse jointed knees are present. If legs would be found to have reversed knees, the bone specific options for the given match would be edited and the reverse option set to true.

- *Dealing with root bones*

In the automated mapping the root of the first loaded source is currently used to define the position of the target root, as there is no metric to define which source might be preferable. The movement and rotation of the target root are still coupled, because it hasn't been a priority to allow setting these separately. When physics are incorporated into the mapping it's possible that this will drive the movement instead, as such the handling of the root hasn't been given detailed attention.

- *Disabling input sources*

It could be interesting to add a button to the input source beyond the first to allow disabling the input. This could then exclude the source from the mapping and allow a more flexible use of the mapping. After loading a second resource it would no longer be required to restart the application to switch back to using a single source. As this is a low priority issue and was considered at a late stage, this wasn't added.

- *Unique limb matching*

Currently it's possible that when the input has multiple pairs of arms or legs and the target has the same, that only one pair of input limbs is used to map to all target pairs of limbs. Especially for legs it could be preferable to include a metric which gives a minor preference to map each pair of target limbs with a different pair of input limbs when possible.

- *Flexible section types*

Create an overview of the sections that are recognized after the automated mapping, to allow a user to modify the recognized types for the sections. Though besides requiring a lot of space, this would also require a lot of work internally, as the mapping would need to be able to work not only from scratch, but also allow using previously calculated information.

Bibliography

deadcode3 (2013). *Alduin* [3d model]. Retrieved from TF3DM: http://tf3dm.com/3d-model/alduin-dragon-rigged--76875.html

3dregenerator (2013). *Hawkgirl* [3d model]. Retrieved from TF3DM: http://tf3dm.com/3d-model/hawkgirl-77858.html

3dregenerator (2012). *Diablo* [3d model]. Retrieved from TF3DM: http://tf3dm.com/3d-model/diablo-9798.html

Arthur Athayde (2010). *Night Fury* [3d model]. Retrieved from Artist-3D: http://artist-3d.com/free_3d_models/dnm/model_disp.php?uid=2895

Bitmapworld (2005). *Horse* [3d model]. Retrieved from TurboSquid: http://www.turbosquid.com/3d-models/3d-horse-animals-lightwave-model/269712

[SSL13] Seol Y., O'Sullivany C., Leez J.: Creature Features: Online motion puppetry for non-human characters. *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2013)*, pages 213-221

[FHXS13] Feng A., Huang Y., Xu Y., Shapiro A.: Fast, automatic character animation pipelines *Comp. Anim. Virtual Worlds (2013)*

[BTST12] Bharaj G., Thormählen T., Seidel H.P., Theobalt C.: Automatically Rigging Multi-component Characters. *Computer Graphics Forum*, Volume 31, May 2012, pages 755-764

[PS12] Pantuwong N., Sugimoto M.: A novel template-based automatic rigging algorithm for articulated-character animation. *Computer Animation and Virtual Worlds*, Volume 23, pages 125–141, March 2012

[YAH10] Yamane K., Ariki Y., Hodgins J.: Animating Non-Humanoid Characters with Human Motion Data. *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 169-178

[HKG06] Heck R., Kovar L., Gleicher M.: Splicing Upper-Body Actions with Locomotion. *EUROGRAPHICS 2006*, Volume 25, Number 3

Appendix

Table 8: Experiment results overview

Input bones	**Stanley**	**Rose**	**Wyvern (Alduin)**	**Horse**	**Hawkgirl**	**Diablo**	**Dragon (Toothless)**
root	root	root	root	root	root	root	root
r_hip	r_hip	r_hip	r_hip	r_hip, r_frontuppertorso	r_hip	r_hip	r_shoulderlow, r_hip
r_knee	r_knee	r_knee	r_upperleg	r_knee, r_frontlowertorso	r_upperleg	r_upperleg	r_upperarm, r_upperleg
r_ankle	r_ankle	r_ankle	r_lowerleg	r_ankle, r_upperfrontleg	r_lowerleg	r_lowerleg	r_lowerarm, r_lowerleg
r_subtalar	r_subtalar	r_subtalar		r_subtalar, r_lowerfrontleg	r_subtalar	r_ankle	r_hand, r_ankle
r_foot	r_foot	r_foot	r_talon	r_foot, r_frontleghoof	r_foot	r_foot	r_subtalar
l_hip	l_hip	l_hip	l_hip	l_hip, l_frontuppertorso	l_hip	l_hip	l_shoulderlow, l_hip
l_knee	l_knee	l_knee	l_upperleg	l_knee, l_frontlowertorso	l_upperleg	l_upperleg	l_upperarm, l_upperleg

l_lowerarm, l_lowerleg	l_hand, l_ankle	l_subtalar	spine1	spine2	spine3	spine4	spine5	neck1	head	jaw
l_lowerleg	l_ankle	l_foot		spine1	spine2	spine3		neck1	neck2	head
l_lowerleg	l_subtalar	l_foot		spine1	spine2	spine3		neck1	neck2	head
l_ankle, l_upperfrontleg	l_subtalar, l_lowerfrontleg	l_foot, l_frontleghoof			spineback	spinefront		neck1	neck2	head
l_lowerleg		l_talon			spine1	spine2		neck1	neck2	head
l_ankle	l_subtalar	l_foot	vl5	vl3	vl1	vt10	vc4		vc2	skullbase
l_ankle	l_subtalar	l_foot	vl5	vl3	vl1	vt10	vt1	vc4	vc2	skullbase
l_ankle	l_subtalar	l_foot	vl5	vl3	vl1	vt10	vt1	vc4	vc2	skullbase

r_rearstable41, r_rearstable31, r_rearstable21, r_rearstable11, r_wing1	r_rearstable42, r_rearstable32, r_rearstable22, r_rearstable12, r_wing2	r_rearstable43, r_rearstable33, r_rearstable23, r_rearstable13, r_wing3	r_rearstable44, r_rearstable34, r_rearstable24, r_rearstable14, r_frontstabil, r_wing4	l_rearstable41, l_rearstable31, l_rearstable21, l_rearstable11, l_wing1	l_rearstable42, l_rearstable32, l_rearstable22, l_rearstable12, l_wing2	l_rearstable43, l_rearstable33, l_rearstable23, l_rearstable13, l_wing3	l_rearstable44, l_rearstable34, l_rearstable24, l_rearstable14, l_frontstabil, l_wing4
r_topshoulder, r_botshoulder	r_topupperarm, r_botupperarm	r_toplowerarm, r_botlowerarm	r_tophand, r_bothand	l_topshoulder, l_botshoulder	l_topupperarm, l_botupperarm	l_toplowerarm, l_botlowerarm	l_tophand, l_bothand
r_shoulder, r_wingbone1	r_upperarm, r_wingbone2	r_lowerarm, r_wingbone3	r_hand, r_wingbone4	l_shoulder, l_wingbone1	l_upperarm, l_wingbone2	l_lowerarm, l_wingbone3	l_hand, l_wingbone4
	r_shoulder	r_upperwing	r_lowerwing		l_shoulder	l_upperwing	l_lowerwing
r_sternoclavicular, r_eyeball_joint	r_shoulder	r_elbow	r_wrist, r_eyeball	l_sternoclavicular, l_eyeball_joint	l_shoulder	l_elbow	l_wrist, l_eyeball
r_sternoclavicular	r_shoulder	r_elbow	r_wrist	l_sternoclavicular	l_shoulder	l_elbow	l_wrist
r_sternoclavicular	r_shoulder	r_elbow	r_wrist	l_sternoclavicular	l_shoulder	l_elbow	l_wrist

l_frontstable11, l_frontstable12, l_frontstable13, l_frontstable14, l_frontstable21, l_frontstable22, l_frontstable23, l_frontstable24, l_frontstable31, l_frontstable32, l_frontstable33, l_wing_thumb1, l_wing_thumb2, l_wing_thumb3, l_wing_thumb4, l_wing_thumb5, l_wing_thumb6, l_wing_index1, l_wing_index2, l_wing_index3, l_wing_index4, l_wing_index5, l_wing_index6, l_wing_middle1, l_wing_middle2, l_wing_middle3, l_wing_ring1, l_wing_ring2, l_wing_ring3, l_wing_ring4, l_wing_pink1, l_wing_pink2, l_wing_pink3, l_wing_pink4, l_wing_thing1, l_wing_thing2, l_wing_thing3, l_hand_digit1, l_hand_digit2, l_hand_digit3, l_hand_digit4, l_footdigit1, l_footdigit2, l_footdigit3, l_footdigit4, r_frontstable11, r_frontstable12, r_frontstable13, r_frontstable14, r_frontstable21, r_frontstable22, r_frontstable23, r_frontstable24, r_frontstable31, r_frontstable32, r_frontstable33, r_wing_thumb1, r_wing_thumb2, r_wing_thumb3, r_wing_thumb4, r_wing_thumb5, r_wing_thumb6, r_wing_index1, r_wing_index2, r_wing_index3, r_wing_index4, r_wing_index5, r_wing_index6, r_wing_middle1, r_wing_middle2, r_wing_middle3, r_wing_ring1, r_wing_ring2, r_wing_ring3, r_wing_ring4, r_wing_pink1, r_wing_pink2, r_wing_pink3, r_wing_pink4, r_wing_thing1, r_wing_thing2, r_wing_thing3, r_hand_digit1, r_hand_digit2, r_hand_digit3, r_hand_digit4, r_footdigit1, r_footdigit2, r_footdigit3, r_footdigit4 , tail17, tail18, tail19	tail_01, tail_02, tail_03, tail_04, tail_05, tail_06, tail_07, tail_08, tail_09, tail_10, tail_11, tail_12, tail_13, tail_14, tail_15, tail_16, neck2, neck3, neck4, neck5, neck6, l_shoulderup, l_wing5, r_shoulderup, r_wing5
tail01, tail02, tail03, tail04, tail05, tail06, tail07, tail08, tail09	
tail1, tail2, tail3, tail4, tail5	neck3
l_wingdigit11, l_wingdigit12, l_wingdigit31, l_wingdigit32, l_wingdigit33, l_wingdigit34, l_wingdigit41, l_wingdigit42, l_wingdigit43, l_wingdigit44, l_wingdigit51, l_wingdigit52, l_wingdigit53, l_wingdigit54, l_wingdigit21, l_wingdigit22, l_wingdigit23, l_wingdigit24, l_wingmiddigit1, l_wingmiddigit2, l_wingmiddigit3, r_wingdigit11, r_wingdigit12, r_wingdigit31, r_wingdigit32, r_wingdigit33, r_wingdigit34, r_wingdigit41, r_wingdigit42, r_wingdigit43, r_wingdigit44, r_wingdigit51, r_wingdigit52, r_wingdigit53, r_wingdigit54, r_wingdigit21, r_wingdigit22, r_wingdigit23, r_wingdigit24, r_wingmiddigit1, r_wingmiddigit2, r_wingmiddigit3, tail01, tail02, tail03, tail04, tail05, tail06, tail07, tail08, tail09, tail10	neck3, neck4
l_index0, l_index1, l_index2, l_index3, l_middle0, l_middle1, l_middle2, l_middle3, l_pinky0, l_pinky1, l_pinky2, l_pinky3, l_ring0, l_ring1, l_ring2, l_ring3, l_thumb0, l_thumb1, l_thumb2, l_thumb3, r_index0, r_index1, r_index2, r_index3, r_middle0, r_middle1, r_middle2, r_middle3, r_pinky0, r_pinky1, r_pinky2, r_pinky3, r_ring0, r_ring1, r_ring2, r_ring3, r_thumb0, r_thumb1, r_thumb2, r_thumb3	vt1
Passive	None