Chapter 1

Background Work

In this chapter, we delve into the foundational research areas central to this thesis. We begin by exploring avatars, examining various methods for their representation and animation. Following this, we review the evolution of Sign Language descriptions, tracing their progression from linear glosses to non-linear representations. Finally, we focus on the recent advancements in Sign Language synthesis, discussing various techniques and their implications for the field.

1.1 Avatars

Avatars are digital representations of characters or individuals, often used in virtual environments and simulations. They can be designed to resemble humans, animals, or fantastical beings, and are crucial in fields such as gaming, virtual reality, and digital communication. Avatars serve as a medium for interaction, allowing users to express themselves and engage with others in a digital space.

1.1.1 Skeleton

The skeleton of a digital avatar is a crucial component for enabling realistic movement and animation. It consists of a hierarchical structure of bones, joints, and constraints that mimic the human skeletal system. This section explores several popular skeleton systems used in avatar creation and animation.

Mixamo

Mixamo is a widely used online platform that provides a vast library of pre-rigged 3D characters and animations. Developed by Adobe, Mixamo offers an easy-to-use interface where users can upload their 3D models and automatically rig them using Mixamo's auto-rigging tool. This tool identifies key points on the

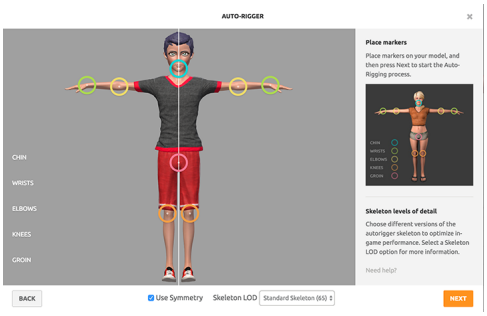


Figure 1.1: Autorigging using Mixamo

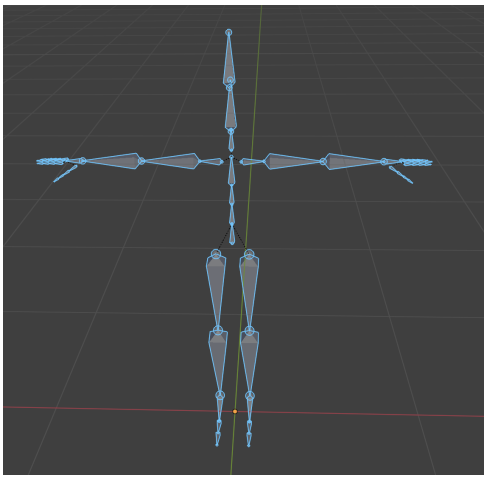


Figure 1.2: Standard Mixamo skeleton structure

model and generates a skeleton with appropriate bone placements. Additionally, Mixamo offers a range of pre-made animations that can be applied to the rigged models, facilitating quick and efficient animation workflows. The platform supports various file formats and is compatible with many 3D software tools. A standard Mixamo skeleton consists of 65 bones, covering major body parts including the spine, arms, legs, hands, and head.

CMU

The Carnegie Mellon University (CMU) Motion Capture Database is one of the most extensive collections of motion capture data available for public use. This database contains thousands of motion capture sequences recorded from real human performers, capturing a wide variety of movements and actions. The heirarchy of a CMU skeleton is shown in figure **??**.

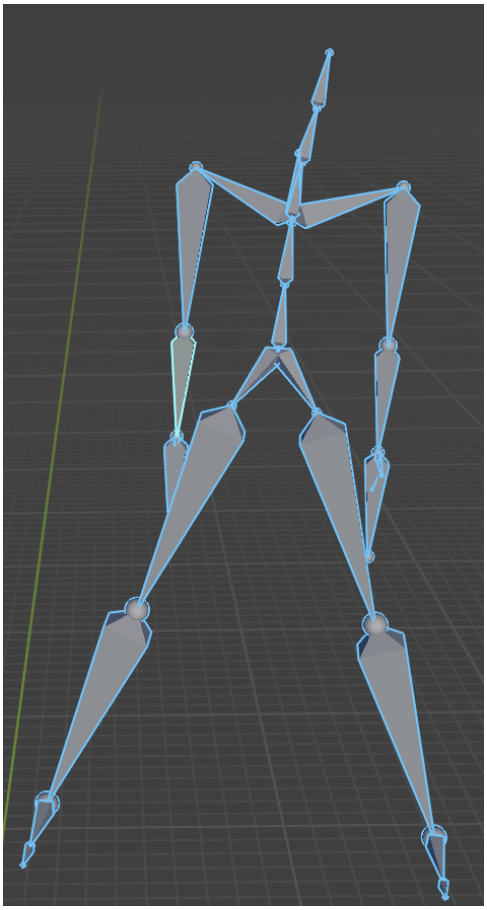


Figure 1.3: CMU skeleton structure

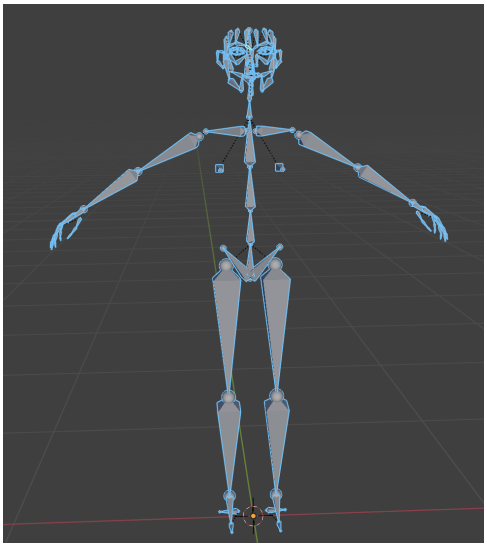


Figure 1.4: Rigify skeleton

Rigify

Rigify is an add-on integrated in Blender. Rigify simplifies the process of rigging by providing a set of predefined rig templates that can be easily customized and applied to 3D models. The heirarchy of a Rigify skeleton is shown in figure **??**.

SMPL-X

SMPL-X (Skinned Multi-Person Linear Model eXtended) is a state-of-the-art parametric model for generating highly detailed and anatomically accurate 3D human avatars. SMPL-X encodes the shape and pose of a character using a low-dimensional space, allowing for efficient representation and manipulation. Thus, a configuration of about 100 numbers can represent the pose as well as the shape of the avatar**??**. The heirarchy of a SMPL-X skeleton is shown in figure **??**.

1.1.2 Mesh

The mesh of a digital avatar is the 3D model that forms the surface representation of the character. It consists of vertices, edges, and faces that define the shape and structure of the avatar. Creating and manipulating meshes is a fundamental aspect of 3D modeling and animation, allowing for detailed and realistic character designs.

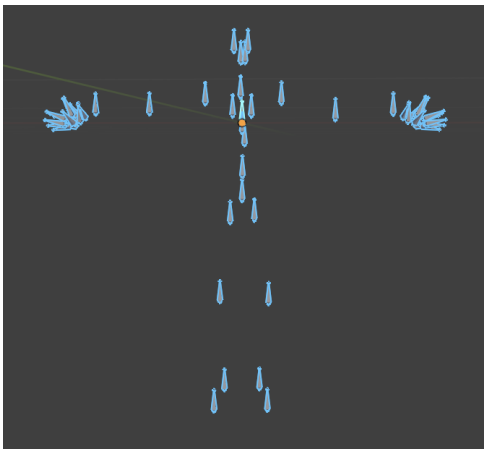


Figure 1.5: SMPL-X skeleton

Figure 1.6: SMPL-X latent space

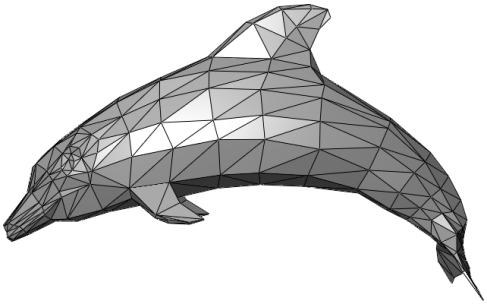


Figure 1.7: Mesh of a shark

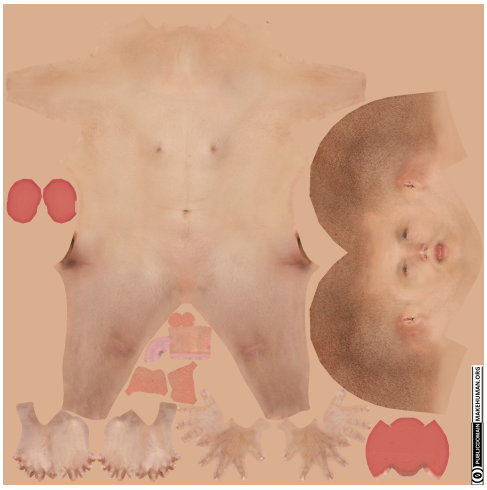


Figure 1.8: Texture map of an avatar

1.1.3 Texture

Textures are 2D images applied to the surface of a 3D model to enhance its appearance and realism. Textures can represent various surface properties such as color, roughness, reflectivity, and transparency. Common types of textures include diffuse maps (color), specular maps (reflectivity), normal maps (surface detail), and roughness maps (surface smoothness). By combining different textures, artists can create visually compelling avatars with intricate surface details and lifelike appearances. These textures are often mapped onto the mesh of the avatar using UV mapping techniques to ensure proper alignment and scaling**??**.

Weight Painting

Weight painting is a technique used to define how much influence each bone in a skeleton has over the surrounding mesh vertices. It plays a crucial role in ensuring that the mesh deforms naturally and realistically when the skeleton is animated. **??** shows an example of weight painting in Blender with the red color indicating high influence and blue indicating low influence.

Face

Modeling and animating the face of an avatar is a complex task that involves creating detailed meshes and intricate animation controls to capture expressions and movements accurately. Blend shapes**??** are used to create different facial expressions by interpolating between multiple versions of a mesh. This technique allows animators to blend between various expressions smoothly, such

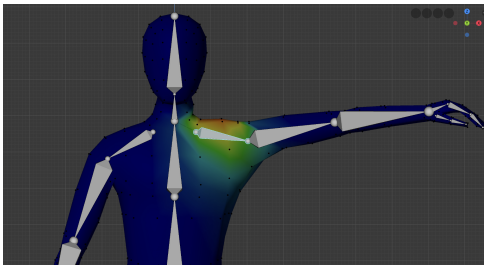
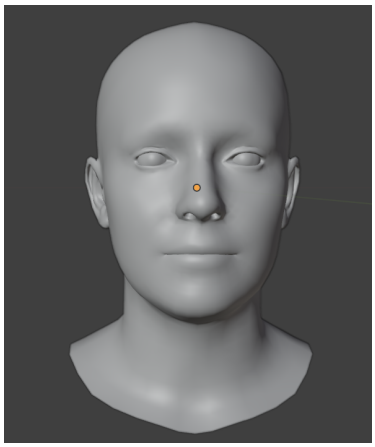
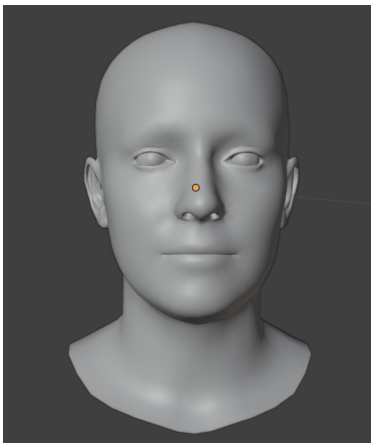
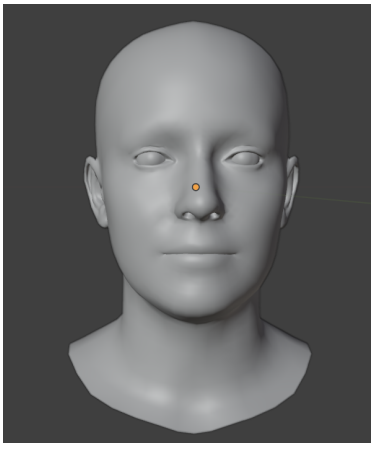


Figure 1.9: Weight painting

as smiling, frowning, or blinking. Adding bones to the facial mesh allows for more granular control over facial movements**??**. Each bone can control different parts of the face, such as the jaw, eyebrows, and eyelids. This method is often used in conjunction with blend shapes to enhance expressiveness.







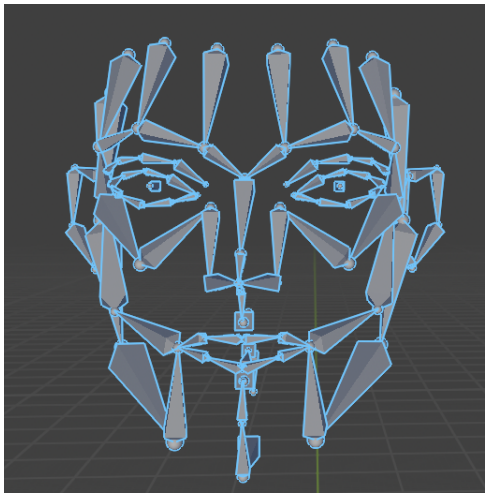


Figure 1.10: Facial bones for eyes, jaw and tongue

Rig

A rig is the final system used by the artists to animate the avatars. This is done by doing all the above systems in order i.e. creating a skeleton, mesh, texture, weight painting, facial blendshapes, implementing IK and FK systems, and adding constraints**??**.

1.1.4 Procedural Avatar Creation

Procedural avatar creation involves the use of algorithms and software tools to automatically generate 3D characters with minimal manual intervention. This approach can significantly speed up the character creation process, allowing for the rapid development of detailed and diverse avatars. This section explores several prominent tools and models used for procedural avatar creation.

MakeHuman

MakeHuman is an open-source tool specifically designed for the rapid prototyping of humanoid avatars. It allows users to create 3D human models through an intuitive interface where parameters such as gender, age, ethnicity, and body proportions can be adjusted using sliders**??**.

SMPL-X or Meshcapade

Just like MakeHuman, SMPL-X can create avatars with parameters suchs as weight, height, age, etc. However, SMPL-X can also be created using images of a person.

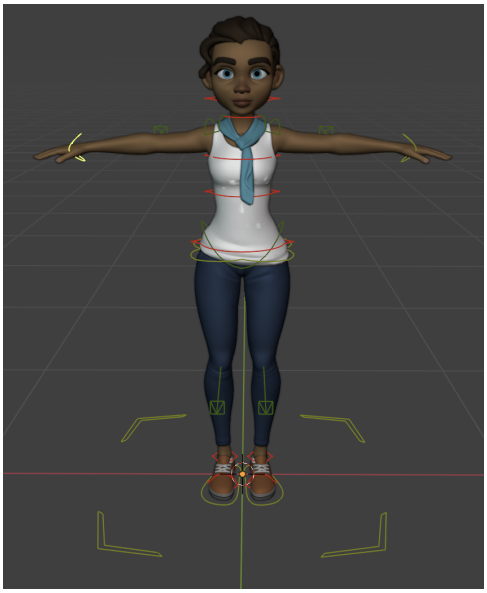


Figure 1.11: Rain rig by Blender studio

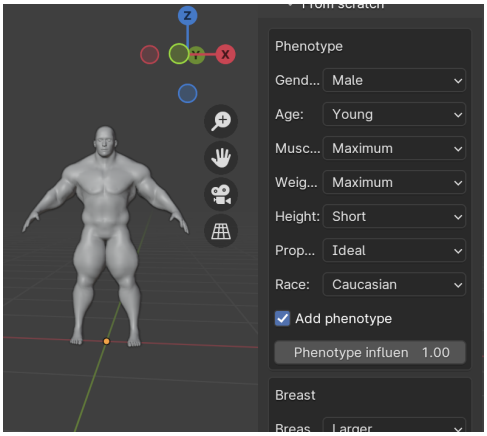


Figure 1.12: Avatar creation using MakeHuman

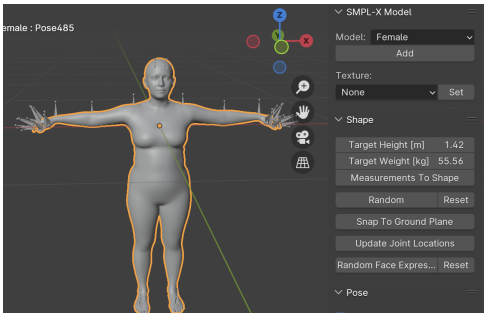


Figure 1.13: SMPL-X avatar creation using Meshcapade

Figure 1.14: Avatar creation using MetaHuman

This allows it to be more realistic than MakeHuman**??**.

MetaHuman

MetaHuman Creator, developed by Epic Games, is a tool for creating ultra-realistic digital humans. The tool's emphasis on realism and detail makes it a powerful resource for creating lifelike characters for games, films, and other interactive experiences**??**. MetaHumans are the most realistic avatars available today, with high-quality textures, detailed facial expressions, and advanced animation controls.

1.1.5 Avatar Animation

The methods used for animating avatars range from manual to automated processes, each offering unique benefits and challenges. This section explores the key methods used in avatar animation.

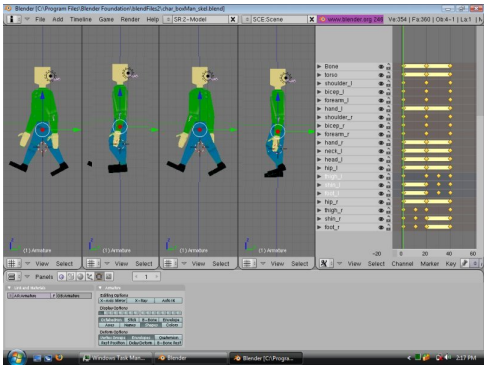


Figure 1.15: Manual keyframing to animate an avatar

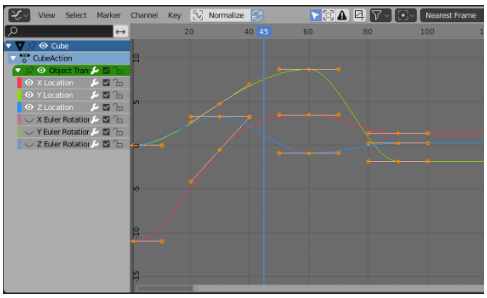


Figure 1.16: Motion curves to finetune an animation

Manual Keyframing

Manual keyframing is a traditional animation technique where animators define specific poses, known as keyframes, at critical points in time. These keyframes mark the start and end points of any smooth transition or movement. The intermediate poses are then calculated using interpolation**??**. This method provides animators with precise control over every aspect of the avatar's movement, making it ideal for achieving nuanced and detailed animations. However, manual keyframing can be time-consuming and requires a high level of skill and expertise to ensure realistic and fluid motion.

Along with keyframing, motion curves[**?**] are used to control the interpolation between keyframes. These curves define the speed and timing of the animation, allowing animators to create smooth and natural movements. Common types of motion curves include linear, ease-in, ease-out, and bezier curves. By adjusting the shape and slope of these curves, animators can fine-tune the animation to achieve the desired effect**??**.

Figure 1.17: Mocap capture and retargeting

Mocap Retargeting

Motion capture (mocap) retargeting involves capturing the movements of a real human actor and applying this data to a digital avatar. This process starts with recording an actor's performance using a motion capture system, which tracks the actor's movements through markers or sensors placed on their body **??**. The recorded data is then mapped onto the avatar's skeleton, a process known as retargeting. Mocap retargeting ensures highly realistic animations by directly transferring the nuances of human motion to the digital character. This technique is widely used in the entertainment industry, particularly in video games and films, to create lifelike animations. While mocap retargeting can significantly speed up the animation process and improve realism, it requires access to specialized equipment and can involve complex data processing to filter noise.

Kinematics

Kinematics is a subfield of physics and mathematics, developed in classical mechanics, that describes the motion of points, bodies (objects), and systems of bodies (groups of objects) without considering the forces that cause them to move. In context of avatar animation, kinematics is used to define the motion of joints in a skeleton, allowing animators to create realistic and dynamic movements. Two key kinematic techniques used in avatar animation are forward kinematics (FK) and inverse kinematics (IK).

Forward Kinematics Forward kinematics (FK) is a method where the position and rotation of each joint in a skeleton are specified explicitly by the animator. This means that to move a hand, for example, the animator must adjust the shoulder, elbow, and wrist joints individually. FK provides precise control over each joint, making it ideal for detailed and deliberate animations. However, it

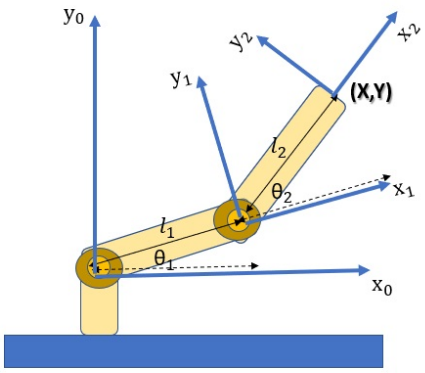


Figure 1.18: Forward kinematics

can be cumbersome for complex movements, as changes to one joint may require adjustments to multiple other joints to achieve a natural pose **??**.

Inverse Kinematics Inverse kinematics (IK) simplifies the animation process by allowing the animator to position the end effector (e.g., a hand or foot) directly, and the software automatically calculates the necessary joint rotations to achieve this position**??**. This technique is particularly useful for tasks like making a character's hand reach a specific point or ensuring that feet remain planted on the ground. IK is widely used in character animation to create natural and realistic movements more efficiently than FK.

IK solvers are algorithms that compute the necessary joint rotations to achieve a desired position of the end effector. Different IK solvers are available, each with its unique approach and strengths. Here are a few popular IK solvers:

- **iTaSc (Instantaneous Task Specification using Constraints)**: iTaSc[**?**] operates by formulating the IK problem as a set of instantaneous task specifications, which are treated as constraints. These constraints can include positions, orientations, and other task-specific requirements. The method solves the IK problem by minimizing a cost function subject to these constraints. It utilizes optimization techniques to handle multiple, potentially conflicting constraints simultaneously, ensuring that the solution adheres to the defined task requirements.

- **CCDIK (Cyclic Coordinate Descent Inverse Kinematics)**: CCDIK[**?**] works by iteratively optimizing the position of each joint in the kinematic chain. Starting from the end effector, the algorithm adjusts each joint to minimize the distance between the end effector and the target position. This is

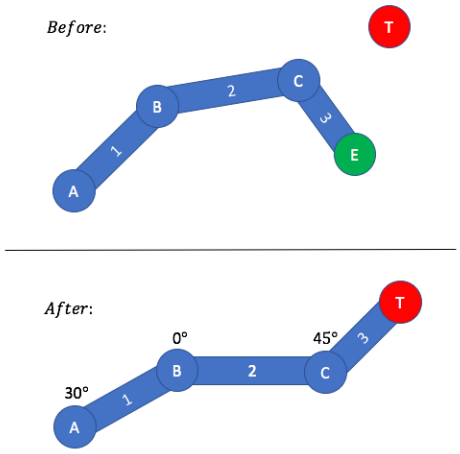


Figure 1.19: Inverse kinematics

done in a cyclic manner, where each joint is adjusted in sequence until the end effector reaches an acceptable proximity to the target. The process is repeated iteratively, ensuring that the adjustments of one joint do not significantly disrupt the adjustments of previous joints.

- **FABRIK (Forward And Backward Reaching Inverse Kinematics)**: FABRIK[**?**] operates through a two-phase iterative process: forward reaching and backward reaching. In the forward reaching phase, the algorithm starts from the base of the kinematic chain and moves towards the end effector, adjusting each joint to align with the target position while respecting joint constraints. In the backward reaching phase, the algorithm starts from the end effector and moves back towards the base, further refining joint positions to ensure convergence towards the target. This alternating process continues until the end effector reaches the desired target within an acceptable tolerance.

Procedural Techniques

Procedural animation techniques use algorithms and rules to generate motion automatically, rather than relying on manual keyframing or motion capture data. These techniques are particularly useful for creating dynamic and responsive animations that adapt to changing conditions or user interactions. Procedural animation can be applied to various aspects of avatar animation, including locomotion, facial expressions, and secondary motion.

TODO add Data-Driven Constraint-Based Motion Editing stuff

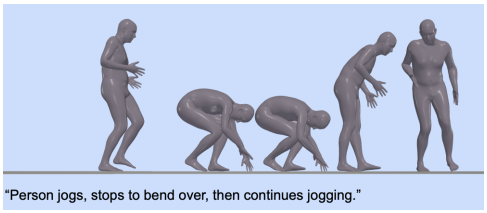


Figure 1.20: Deep Learning based synthesis[**?**]

Deep Learnng based Techniques Deep learning-based techniques have shown significant promise in procedural animation, particularly in the generation of realistic and dynamic movements. Neural networks can be trained on large datasets of motion data to learn complex patterns and generate new animations**??**. For example, diffusion models[**?**] have been used to synthesize human-like motion sequences with high fidelity and diversity from text prompts. Treating character animation as a cross-modal translation task where descriptive sentences serve as inputs to generate corresponding avatar animations. These techniques however require large amounts of data and computational resources to train effectively.

Hybrid

Hybrid animation techniques combine elements of manual keyframing, mocap retargeting, and procedural techniques to leverage the strengths of each method. By integrating these approaches, animators can achieve a balance between control, efficiency, and realism. For instance, an animator might use mocap data as a base and then refine specific movements with manual keyframes to enhance expressiveness. Additionally, procedural techniques can be applied to automate repetitive tasks or to ensure that certain constraints are met dynamically during the animation process. Hybrid methods offer a flexible and powerful workflow, allowing for the creation of complex animations that are both realistic and tailored to the specific needs of a project**??**.

Uncanny Valley

The Uncanny Valley is a concept in the field of robotics and 3D animation that describes the discomfort or eeriness that people experience when they encounter a humanoid figure that is very close to, but not quite, human-like. This phenomenon was first identified by roboticist Masahiro Mori in 1970.

As avatars become more realistic in appearance and movement, they initially become more appealing and relatable. However, there is a point at which the avatar becomes almost, but not perfectly, human-like, causing a sense of unease

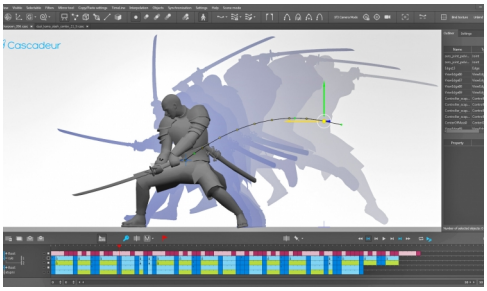


Figure 1.21: Cascadeur hybrid animation tool

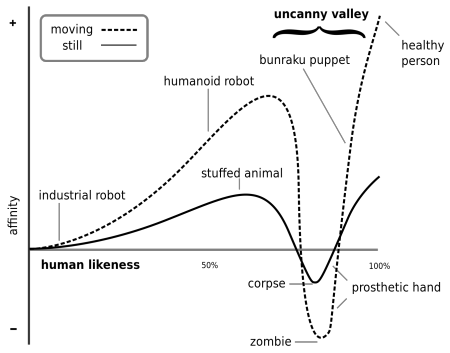


Figure 1.22: Uncanny Valley

or revulsion in the observer. This dip in the graph of familiarity versus human likeness is known as the Uncanny Valley**??**.

The factors contributing to the Uncanny Valley effect often include:

- **Facial Expressions**: Subtle imperfections in facial expressions, such as unnatural eye movements or lifeless smiles, can make the avatar appear creepy or unsettling.
- **Movement**: Non-human-like movements, jerky motions, or slight deviations from natural human motion can heighten the uncanny effect.
- **Texture and Skin Tone**: Inconsistent skin textures, overly smooth surfaces, or unrealistic lighting and shading can make an avatar appear artificial.
- **Eye Contact**: The eyes are critical in conveying emotions and life. Any deviation in eye movement, blink patterns, or focus can make an avatar look eerie.

Understanding and mitigating the Uncanny Valley effect is crucial for animators and developers aiming to create realistic and relatable avatars. One way

to avoid uncanny valley could be to use stylized avatars, which are intentionally designed to deviate from realism and have a unique aesthetic**??**. Another way is to keyframe more information in the animations itself.

1.2 Describing Sign Language

A key challenge in Sign Language synthesis lies in the description of a Sign Language discourse itself. A description of a discourse can answer questions such as:

- What body motions make a sign?
- How do these signs and their respective motions connect to form a discourse?
- What are the grammatical and syntactical rules governing these signs?
- How can non-manual signals (such as facial expressions and body posture) be integrated with manual signs?
- How can non-manual signals (such as facial expressions and body posture) be integrated with manual signs?

In this section, we first study the descriptive languages used to formalize sign languages and how they describe body motions.

Descriptive languages are systems developed to transcribe and analyze sign languages. They provide a framework for representing the complex, multi-dimensional nature of sign language communication. Some notable descriptive languages include:

1.2.1 Lexical Descriptions

Lexical descriptions focus on individual signs and their components, such as handshapes, movements, and locations. These descriptions are essential for understanding the building blocks of sign language and how signs are produced. Some popular lexical descriptions include:

Stokoe Notation

Stokoe Notation[**?**] was one of the first systems developed to transcribe American Sign Language (ASL). It breaks down signs into three main components:

location, handshape, and movement. It is a simple and effective notation for basic transcription, but focuses primarily on the manual components of signs.

A lot of subsequent systems were discovered inspired by Stokoe's approach. These systems are more complex and detailed, and they aim to capture the nuances of sign language more accurately.

SignWriting

SignWriting is a writing system developed by Valerie Sutton just after DanceWriting[**?**] in the 1970s to represent sign languages visually. It was later included as a part of the MovementWriting[**?**] system. It is a featural script, meaning that it represents the features of signs, such as handshapes, movements, and locations, rather than phonetic sounds. SignWriting is designed to be easy to read and write, with symbols that resemble the gestures they represent. The script is written in two dimensions, with symbols arranged in a grid-like structure to capture the spatial and temporal aspects of signs. SignWriting has been used to transcribe over 40 sign languages worldwide and is recognized by the International SignWriting Alphabet (ISWA) organization.

HamNoSys

The Hamburg Sign Language Notation System (HamNoSys) is a phonetic transcription system for documenting sign languages globally, developed in 1985 at the University of Hamburg. Unlike Stokoe's notation, which was specifically created for American Sign Language (LSF), HamNoSys aims for broader application, transcending national sign language boundaries. While Stokoe notation later adapted to other sign languages, HamNoSys was designed from the outset to accommodate the diversity found in global sign languages.

The notation system includes nearly 200 symbols categorized into five main types, which together depict the parameters of a sign. These symbols can include subscripts, superscripts, and diacritics. A sign is described through a series of these symbols, listed in a specific order:

- **Handshape**: Describes the hand configuration, including the number and position of extended fingers, degree of finger bending, and thumb location. Common symbols include ovals and lines to signify finger positions and curves to represent bending.

- **Hand Orientation**: Indicates the orientation of the hand and direction of the fingers or palm. Symbols include ovals with thick sides and small arrows or carets for advanced angles.

- **Hand Location**: Specifies where the hand is positioned relative to the body, using symbols that often resemble the body parts involved, with a black square marking the feature's orientation.
- **Movement (Optional)**: Details any motion in the sign, such as direction and type of movement (e.g., straight, wavy, circular). These are represented by complex symbols with arrows, lines, and brackets to show sequential or unified movements.
- **Symmetry Operator (Optional)**: Indicates if both hands are used or specifies the use of the non-dominant hand, usually represented by two dots.
- **Non-Manual Marker (NMM) (Optional)**: Denotes facial or vocal actions accompanying the sign, such as lip-pursing or eyebrow-raising.

HamNoSys offers a detailed representation of the nuanced components in sign language discourse rather than serving as a practical writing system. Its main purpose is linguistic, used primarily by linguists to analyze the specific features of individual signs. However, it has also found it's applications in synthesis**??** as well has sign detection**??**.

1.2.2 Arranging Lexicals

A sign language description is represented as an arrangement of lexicals or glosses. This arrangement can be sequential(linear) or non-linear.

Sequential Arrangement

Sequential arrangement of lexicals is a linear representation of sign language discourse. In this approach, signs are listed in the order they appear in the discourse, with each sign described using lexical notation. This method is straightforward and easy to follow, making it suitable for basic transcription and analysis. However, it may not capture the full complexity of sign language, particularly the non-linear aspects of signing such as simultaneous movements and facial expressions.

P/C Model

Unlike HamNoSys, the Prosodic and Chronemic Model (P/C Model) **??** is a non-linear framework for describing sign language. This model focuses on capturing the rhythmic and temporal aspects of signing, recognizing that the timing and flow of signs are essential for their meaning. The P/C Model is particularly useful

for understanding the nuances of sign language that go beyond simple hand movements, including the speed, intensity, and pauses that convey additional meaning. However, it has not been adopted for sign language synthesis.

Canadian stuff

https://github.com/PhonologicalCorpusTools try to define sign language phonology based on articulatory features but has no synthesis applications.

AZee

doesnt assume sequence doesnt use stokoe

The AZee model is based on the notion of production rule, which associates a meaning to a set of observable forms. The model is based on the idea that the meaning of a sign can be decomposed into a set of features, which can be represented as a set of production rules. These rules are then used to generate the form of the sign, including the handshape, movement, and location. The AZee model is designed to be flexible and extensible, allowing for the representation of a wide range of signs and sign languages. It is particularly well-suited for sign language synthesis, as it provides a systematic and structured approach to describing signs and their components.

1.3 Sign Language Synthesis

Sign Language synthesis is the process of generating sign language animations from linguistic input. This process involves resolving a linguistic description into the anatomy of the avatar such as handshapes, movements, facial expressions, etc. on an avatar. Sign Language synthesis is a complex and interdisciplinary field that draws on linguistics, computer science, animation, and human-computer interaction. Several approaches have been developed to synthesize sign language animations, each with its unique strengths and challenges.

1.3.1 JASigning

JASigning is an advanced sign language avatar system developed within the scope of the ViSiCAST and eSIGN projects[**?**]. It enables the automatic generation and animation of sign language through a predefined set of gestures and animations. These gestures can be dynamically combined to form coherent sign language sentences. JASigning is designed to support multiple sign languages and integrates with text-to-sign translation systems. The system uses HamNoSys

in the form of an XML representation[**?**] to describe signs and their components. JASigning has been used in various applications, including educational tools, communication aids, and virtual avatars.

1.3.2 EMBR

TODO

Unlike the previous models, EMBR (Embodied Agents Behaviour Realizer) Script **??** is a scripting language used to control virtual characters in general and is not restricted to sign language avatars. However, EMBRScript allows for animation specification through a sequence of key poses, each defined at specific time points for a precise specification of body movements, facial expressions, and other actions. This also makes it usable to describe signs with synthesis using an avatar in mind.

EMBR (Embodied Agents Behaviour Realizer)[**?**] is a real-time animation engine engineered to produce expressive gestures and sign language for virtual avatars. EMBR facilitates precise control over avatar movements, encompassing hand shapes, facial expressions, and body postures critical for accurate sign language depiction. The framework supports scripting and integrates with various input modalities, such as text or motion capture data, to generate realistic and contextually appropriate sign language animations. EMBR's flexibility and adaptability make it a valuable tool in research settings, particularly for studies focused on the nuances of non-verbal communication and the development of nuanced, context-sensitive gestures.

1.3.3 Sign3D

The Sign3D project developed tools for high-fidelity 3D recording of French Sign Language (LSF) using an optical motion capture system with head-mounted oculometers. They annotated captured data with Elan software, segmenting signs and adding meta-data. A visual composition interface was created for assembling new sentences from the database. These motion capture chunks were concatenated and optimized for smooth transitions. The process concluded with rendering a 3D virtual signer for accurate motion playback.

1.3.4 Synthesis based on generative models

Generative models have shown promise in sign language synthesis by learning the underlying structure of sign language data and generating new animations. These models can be trained on large datasets of sign language videos to cap-

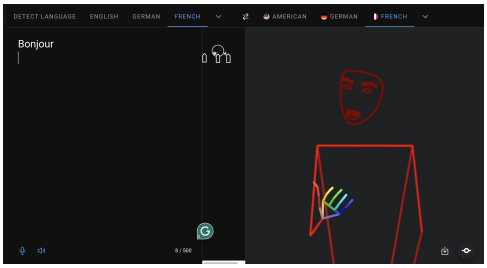


Figure 1.23: Synthesis using SignWriting as an intermediate representation

ture the complex relationships between linguistic input and sign language output. Some popular generative models used in sign language synthesis include:

Gloss based

Sign language animation from text focuses on generating realistic sign language videos from written text. The process involves text-to-gloss conversion, gloss-to-pose generation, and pose-to-sign rendering. Recent approaches utilize neural networks and generative models to enhance animation fluidity and realism. One method uses dictionary examples and a codebook of signs to create continuous sign language sequences, incorporating both manual and non-manual features[**?**]. However, challenges remained in annotating the data and training the models effectively, particularly in the production of natural hand and facial movements.

Sign Writing based

Another approach[**?**] used the SignBank dataset, which contains 220k parallel samples of SignWriting and spoken language text. SignWriting served as an intermediate representation, translating spoken text into SignWriting before converting it into human poses**??**. Neural machine translation (NMT) architectures with techniques like byte pair encoding (BPE) segmentation were used. Evaluations show over 30 BLEU scores in bilingual settings and over 20 BLEU in multilingual settings, validating the approach and highlighting the effectiveness of SignWriting in NLP for sign language processing.

SignAvatars

The SignAvatars dataset introduces a large-scale, multi-prompt 3D sign language motion dataset with 70,000 videos and 8.34 million frames from 153 signers. It includes isolated and continuous signs, annotated with 3D meshes and biomechanically-valid poses. The dataset supports tasks such as 3D sign language recognition

and production from text scripts, individual words, and HamNoSys notation. A novel annotation pipeline ensures accurate 3D holistic annotations. Evaluation metrics indicate significant improvements in generating natural and consistent 3D sign language animations from diverse textual inputs using models like Sign-VQVAE.

Sgnify

Recent advancements in sign language processing involve synthesizing sign language animations from textual descriptions. The process includes text-to-gloss conversion, gloss-to-pose generation, and pose-to-sign rendering. One notable approach uses neural networks to convert text into SignWriting, which is then transformed into human poses. SGNify utilizes linguistic priors and SMPL-X for accurate 3D hand pose, facial motion, and body pose from monocular video. Techniques such as filtering in the frequency domain and resampling ensure natural rhythm and prosody, overcoming challenges like motion blur and occlusion, resulting in more comprehensible and natural sign language animations.

1.3.5 AZee based

Paula

The system extends the Paula[**?**] avatar using AZee descriptions. AZee encodes both form and functional linguistic aspects, while Paula ensures smooth human motion. The approach leverages a hierarchical model to specify movements using larger linguistic structures, improving naturalness. Key innovations include embedding geometric constraints and utilizing procedural techniques for dynamic, realistic animations. Proform placements are optimized by factoring semantic functions and applying common forms across multiple productions. This method addresses limitations in previous models, such as the lack of supporting torso motion and dynamic differences, leading to more fluid and expressive sign language animations.**??**

Fabrizio

This work presents a bottom-up(or low-level generation using constraint optimization) synthesis solution for the AZee system using off-the-shelf IK solvers. The approach generates procedurally computed animations from AZee's symbolic descriptions, leveraging Blender's 3D editor and iTaSC solver. This method handles constraints as IK problems, translating skeletal poses into keyframes. Despite inherently robotic motion, it serves as a low-level fallback for the existing top-down system. Preliminary results show effective static pose generation

and a trade-off between precision and computation time, enhancing the flexibility of sign language avatar systems by integrating procedural synthesis with predefined animations.**??**

TODO visualize sites like -> https://mosh.is.tue.mpg.de/

1.4 Conclusion

Throughout this chapter, we have introduced, defined, and discussed a variety of concepts from three research areas that are key to this thesis: *concept1*, *concept2*, and *concept3*. In the rest of the manuscript, we heavily rely on many of these concepts to investigate ways in which models for sign language synthesis can progressively learn and improve from new data, even well after their initial development phase. In other words, we put a special focus on the production phase of these machine learning models, where data is usually unlabeled and available progressively as time passes. In particular, *concept2* and *concept3* are two subjects that appear regularly to help introduce more complex ideas.

Therefore, in Chapter **??** and Chapter **??**, we study and exploit task-specific neural representations, especially for *specific-task*, which are crucial in the context of *application1*. Additionally, the studies presented in Chapter **??**, Chapter **??**, and Chapter **??** dive deeper into *concept3* for sign language-related applications, while borrowing key concepts from *concept2* as they are described in this chapter.

The background work specific to the various tasks and applications we address throughout the manuscript is presented and discussed in the corresponding chapters. Previous work on *task1* and *task2* is presented in Chapter **??**. *Task3*, and in particular *subtask1* and *subtask2*, are discussed in Chapter **??**, while *task4* is discussed in Chapter **??** and Chapter **??**.

TODO - linear systems bad - non-linear systems give good coverage - paula azee like system is good - but hand crafted aniamtions bad for coverage/labour intensive - infinite things so cannot have coverage - need a low-level synthesis - but azee's low-level was incomplete - complete azee to do more stuff

and more?