Animation Models for Interactive AR Characters

Author(s):
Çimen, Gökçen

Publication Date:
2019

Permanent Link:

https://doi.org/10.3929/ethz-b-000372660 →

Diss. ETH No. 25899

Animation Models for Interactive AR Characters

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Gökçen Çimen
MSc in Computer Science, Bilkent University, Turkey
Born on 19.02.1988
Citizen of Turkey

accepted on the recommendation of
Prof. Dr. Robert W. Sumner, examiner
Prof. Dr. Markus Gross, co-examiner
Prof. Dr. Metin Sezgin, co-examiner

2019

Abstract

Technological advances in virtual and augmented reality (AR) have enabled new ways for users to interact with digital characters. The use of virtual characters in our lives by means of new digital assistants, avatars, virtual pets or digital toys is rapidly increasing. Besides the visual fidelity of virtual characters, motion plays a crucial role in interactive applications. However, the characters in AR often have limited or no responsiveness to the user's presence and to other stimuli— i.e., the user's movements or changes in the real-world environment. In this thesis, we investigated several approaches in modeling the characters' motions. The focus was particularly on the methods that could enable virtual characters to give human-like responses during the interactions with users and real-objects.

Building character animations is a well-established field in the film and gaming industries. Characters are designed by artists, rigged to define joint movements and then animated with careful timing to match animation scripts. However, unlike in traditional video games or animation movies, the character's surrounding and the user input may not be predefined in AR applications. Integrating physics-based approaches into characters' motion models has a particular advantage in interactive AR applications as they can generate motions that are online and responsive to environmental changes and user inputs. In this thesis, we explore methods for motion models that allows virtual characters human-like responses to interactions with users and real-objects.

Augmented reality has the power to turn our physical environments into digital gaming platforms by combining real and virtual objects. Besides focusing on the physical aspects of the motion that supports the interaction with real objects and natural responses to perturbations, perception capabilities like vision-based human pose estimation, object recognition and reconstruction are investigated. A first step in the direction of interactive AR characters is taken in this thesis that can understand and react to environmental changes and user's behaviors. The user can use his or her own body movements and physical objects in the surrounding in the most natural and intuitive way to interact and play with AR characters. Finally, two frameworks are presented that incorporate digital characters and costumes into selfie settings in AR, which allows virtual characters to mimic the user's pose, or the user to wear characters' costumes. The approaches utilize

the latest advancements of deep learning for 2D pose estimation in the wild, combining with a projection onto the 3D subspace to find the closest matching 3D pose and minimum parametrization, assuming selfie scenario— which enables mobile devices to be used.

Zusammenfassung

Technologische Fortschritte in der virtuellen und erweiterten Realität (AR) eröffnen dem Benutzer neue Möglichkeiten, mit digitalen Charakteren zu interagieren. Die Verwendung virtueller Charaktere in unserem Leben, die als neue digitale Assistenten, Avatare, virtuelle Haustiere und digitale Spielzeuge genutzt werden, nimmt rapide zu. Neben der visuellen Genauigkeit virtueller Charaktere spielt Bewegung in interaktiven Anwendungen eine entscheidende Rolle. Die Charaktere in AR haben jedoch oft eine eingeschränkte (oder keine) Reaktion auf die Anwesenheit des Benutzers und auf andere Reize - wie etwa Bewegungen eines Benutzers oder Änderungen in der realen Umgebung. In dieser Arbeit untersuchen wir Methoden für Bewegungsmodelle, mit denen virtuelle Charaktere menschenähnlich auf die Interaktionen mit Benutzern und realen Objekten reagieren können. Das Erstellen von Charakteranimationen ist ein etablierter Bereich in der Film- und Spieleindustrie. Charaktere werden von Künstlern entworfen, manipuliert, um gemeinsame Bewegungen zu definieren, und werden dann mit einem sorgfältigen Timing animiert, damit sie den Animationsskripten entsprechen. Im Gegensatz zu herkömmlichen Videospielen oder Animationsfilmen ist es nicht möglich die Umgebung des Charakters und die Benutzereingaben in Augmented-Reality-Anwendungen vorzudefinieren. Die Integration von physikbasierten Charakteransätzen in die Bewegungsmodelle von Charakteren hat einen besonderen Vorteil in den interaktiven AR-Anwendungen, da sie online Bewegungen erzeugen können, die auf die Umgebungsänderung und die Benutzereingaben reagieren. Wir zeigen die Verwendung von Physik-basierten Zeichenanimationen bei der Leistungsverfolgung, bei denen der Benutzer die Bewegungen digitaler Zeichen mit unterschiedlichen Morphologien steuert, die für interaktive AR-Szenarien erweitert werden können.

Augmented Reality bietet die Möglichkeit, physische Umgebungen in digitale Spieleplattformen zu verwandeln, indem reale und virtuelle Objekte kombiniert werden. Nebst dem Fokus auf die physischen Aspekte der Bewegung, welche die Interaktion mit realen Objekten und die natürlichen Reaktionen auf Störungen unterstützen, untersuchen wir Wahrnehmungsfähigkeiten wie die auf Sicht basierende Schätzung der menschlichen Haltung, die Objekterkennung und die Rekonstruktion. Wir machen einen ersten Schritt in Richtung interaktiver AR-Charaktere, die Umgebungsveränderungen und das Verhalten des

Benutzers verstehen und darauf reagieren können, während der Benutzer seine eigenen Körperbewegungen und physischen Objekte in der Umgebung auf natürlichste und intuitivste Art und Weise zur Interaktion verwendet und mit AR-Charakteren spielt. Schließlich stellen wir einen Rahmen vor, welcher digitale Charaktere und Kostüme in die Selfie-Einstellungen von AR integrieren, wodurch virtuelle Charaktere die Haltung des Benutzers nachahmen können oder der Benutzer die Kostüme von Charakteren trägt. Unser Ansatz nutzt die neuesten Fortschritte des Tiefenlernens für die Schätzung von 2D-Posen in freier Wildbahn, kombiniert mit einer Projektion in den 3D-Unterraum, um die am besten passende 3D-Pose und minimale Parametrisierung zu finden, unter der Annahme eines Selfie-Szenarios, das die Verwendung mobiler Geräte ermöglicht.

Acknowledgments

Undertaking this PhD within the Computer Graphics Lab, ETH Zürich, and working with truly passionate and creative people in animation and games lab at Disney Research Zürich was an honor. Foremost, I would like to express my sincere gratitude to my advisor, Prof. Bob Sumner, who made this possible. Without his encouragement, support and continuous optimism, this thesis would hardly have been completed. I couldn't have imagined having a better advisor and mentor for my Ph.D journey helping me grow academically and personally.

Second, I would particularly like to single out my friend and mentor, Martin Guay, who had a great impact on my PhD pursuit. I remember feeling overwhelmed by the complexity of work shortly after I started my PhD. I am grateful to him for keeping me on track, even during tough times— thank you for your patience and guidance.

I would like to thank all of my co-authors, Prof. Markus Gross, Prof. Bob Sumner, Prof. Stelian Coros, Prof. Kenny Mitchel, Martin Guay, Mattia Ryffel, Loïc Ciccone, Llogari Casas, Christoph Maurhofer, Ye Yuan, Pablo Wiedemann and Matthoas Fauconneau, for their wonderful collaboration. It was fantastic to have the opportunity to work with you. I would like to offer my special thanks to the amazing digital artists, for their generous support: Maurizio Nitti, for the beautiful designs of the digital characters, and Alessia Marra, for creating the incredible poster for *AR Poser* work. A very special gratitude goes out to *DISTRO Network* for providing collaborative research and training community, and all collaborators who made whole experience very positive.

I am also very grateful to my all friends who have been a major source of emotional support— a very special thanks to my fabulous flatmates, Merve, for always having a listening ear when things get a bit discouraging, and Bellatrix, for being a good cat and great company during the process of writing my thesis. Living with you is amazing and fun with all the silly things we've done together. To Pelin, thank you for your support and the special coffee times which were the sometimes-all-to-needed breaks. To Egeyar, encouraging me to start learning german together, which was fun and kept me motivated, even though progressing along together was not necessarily at the same pace. To my dear friends far away but still making the effort to stay in touch: Gizem, Bengü, Sinan, Elif and Seher,

for being supportive and caring all these years. To Patrizia and Anina, thanks for your extremely kind friendship and also joining me to awesome music concerts. And to everyone else that was around and encouraging me, thanks for all the good times.

And finally, last but by no means least, I am deeply thankful to my parents and sister for their never ending love, constant support, and sacrifices, which I would not have come this far.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Augmented Reality (AR) and Mixed Reality (MR) blur the line between our physical and digital worlds by truly merging the interactions within them. Naturally, mixing realities provides a new dimension in human-computer interaction and its power comes from the use of the real environment and entities as a new medium of interaction. At present, rapid advances and the easy accessibility of features like cameras, gyroscopes, and accelerometers made mobile devices a popular platform for AR applications. As a result, we can see many applications in a lot of area including marketing and advertising, education, industrial and medical procedures, gaming and entertainment, shown in Fig. 1.1

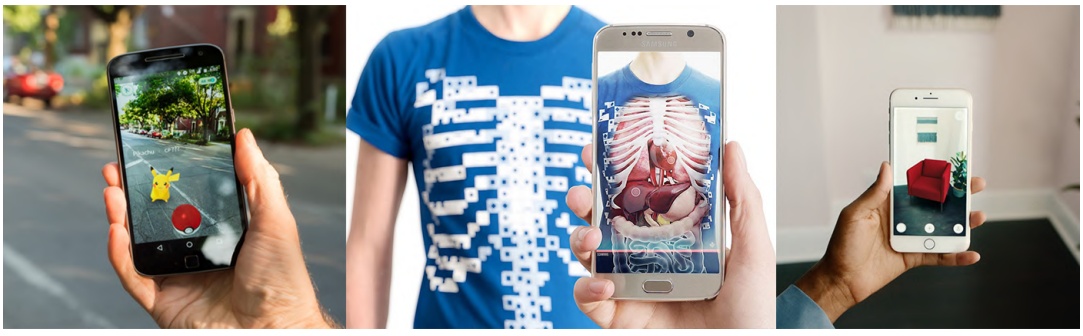

Figure 1.1: *AR is a powerful tool applied to entertainment, education, marketing and other fields.*

Interaction between humans and virtual characters covers a wide range of disciplines including computer vision, artificial intelligence besides conventional character animation techniques in computer graphics. Interactive virtual characters embodied in an environment— in physical or digital form

—should have the ability of generating actions online that are responsive to the environment change and the user input. The superiority of AR and MR applications comes from making virtual characters an integral part of the real world and directly allowing natural physical interactions with users, yet these actions should also be able to show a physical nature and accuracy. The overall goal of this thesis is to investigate different real interaction aspects between users and *interactive AR characters*. It describes the steps towards designing and implementing motion models for virtual characters that can understand the user's environment and freely move and interact in it by applying animation and interaction techniques combining the following two research areas:

- Augmented Reality
- Physics-Based Character Motion

1.1 Overview

The frameworks and interfaces presented in the thesis demonstrates that an effective combination of advantageous features of the aforementioned research domains yields a closer integration of virtual characters into our physical environment. This chapter makes a brief overview of the individual domains and summarizes challenges and contributions.

Augmented Reality

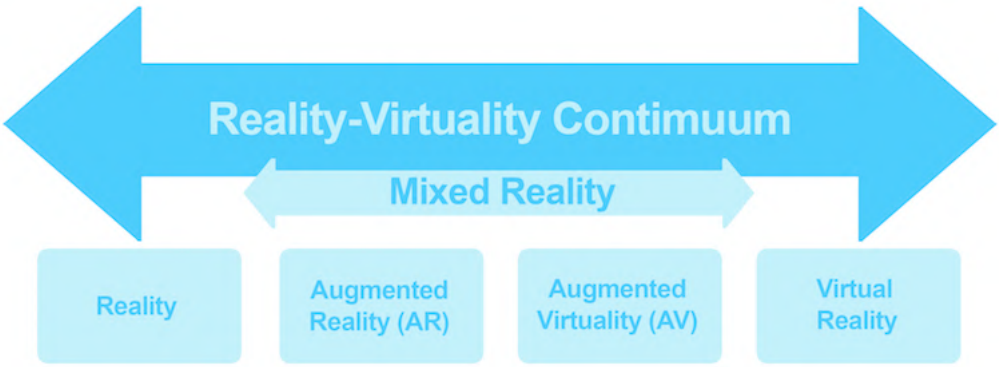

Figure 1.2: *Milgram's Reality Virtuality (RV) continuum. Adapted from [Milgram and Kishino, 1994].*

The Reality-Virtuality Continuum is first introduced by Paul Milgram and Fumio Kishino in 1994 [Milgram and Kishino, 1994] with a new reality as

"environment as one in which real-world and virtual-world objects are presented together within a single display" — *Mixed Reality (MR)*. Milgram's RV continuum draws the continuous transition of real world from the left up to virtual world on the right. As an intermediate case in the spectrum, *Augmented Reality (AR)*, refers to predominantly real-world spaces where virtual elements, e.g. virtual objects or characters, are dynamically integrated into, and can interact with, the physical world in real time .

At one extreme, *Virtual Reality (VR)* is a practice to tricking brain into experiencing a completely alternative reality from our physical world. The biggest difference between VR and other realities is that it totally immerses a user inside a synthetic environment that may or may not imitate real-world properties, such as the physical laws governing space, time, gravity and etc. In VR, the connection of the user with the physical world is through sensory feedback that typically tracks the head and hands of the user. A significant effort is put into creating accurate representations of virtual objects from their physical counterparts. Similarly, the believability and co-presence of a virtual character interacting with a user depends on their human-like responses to users' movements. Recently, Estudiofuture's Vivo Technology presented a platform that allows creating VR characters that are aware of the the user's movements and can give reactions to them with a technique of blending key-framed animations with physical behaviours [Ruiperez and Ruiperez, 2018]. It presents a use case example to show the importance of AI interactivity in VR characters.

Figure 1.3: *VIVO allows animators to create responsive VR characters that can react to a user's reaching out behaviour and his or her touches [Ruiperez and Ruiperez, 2018].*

The main advantage of AR is its direct exploitation of the physical world. In AR, there is no need for the virtual representations of real objects or users while interacting with virtual characters. Therefore, it requires more intuitive and natural ways of interactions, since users and physical objects in the surrounding environment remain visible.

According to the seminal work "A survey of Augmented Reality" by Ronald

T. Azuma [Azuma, 1997], the following three parameters defines an AR system:

- Combines real and virtual
- Interactive in real time
- Registered in 3-D

Rather than completely replacing it, AR supplements reality by combining virtual characters and real objects into the same space with the user. The parameters specifically ensure the role of the interactivity itself in AR. Hence, simply imposing a virtual character upon the real space that is standing but not interacting with a user is not enough for a good AR experience. Instead, animation models for virtual characters can exploit the input mechanisms (e.g., sensors and cameras) of AR systems that collect user's real-world interactions to interpret and process. This thesis presents several AR applications taking full advantage of the physical world, especially focusing on the user tracking and the bodily interactions between virtual characters and users. While concentrating on animating them in a sufficiently believable and natural way, it explores the interactions of virtual characters that can realize and mimic users' movements and intelligently inhabit the real world merged with virtual and real objects.

Even though the type of the tracking equipment depends on the device, there are three common tracking classes in current AR systems: user, object and spatial tracking, illustrated in Fig. 1.4. Markers or spatial tracking to detect surfaces allows properly positioning the virtual character and other contents into the environment. Object recognition and tracking depends on feature points extracted from the structure of the object through scanning. The frameworks presented on this thesis target handheld devices, such as mobile phones, with vision-based tracking methods. That is, recognition and tracking of objects and/or user are achieved by the analysis of the camera image. The user tracking comprises not only the position and orientation of the user, which is solely based on device tracking, but also the estimation of the user's 3D skeletal pose directly from camera image.

Physics-Based Character Motion

Animation models for interactive virtual characters that can give lifelike responses to the unexpected disturbances in a dynamically changing environment in real time requires motion control. The physics-based character control uses physics (equations of motion) and operates at the mechanical level of the motion. Just like real-world, the characters are controlled via forces

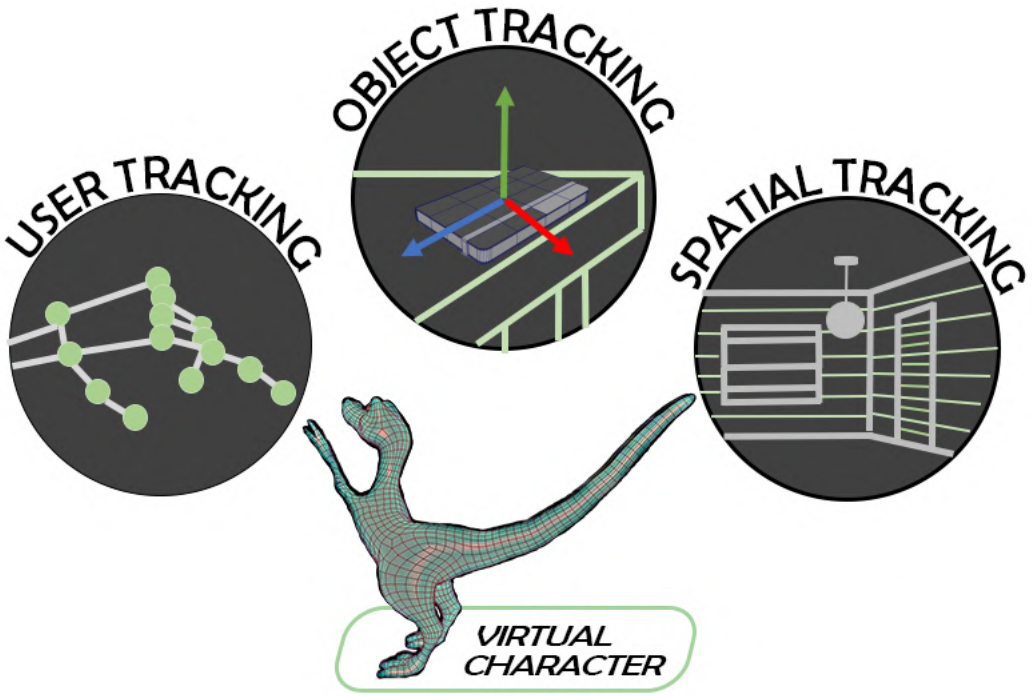

Figure 1.4: *User tracking for detecting users' positions together with their body poses, object tracking for recognizing real-world objects, and spatial tracking for understanding the environment, including surface detections are essential tools to enhance interactivity in AR.*

and torques which can be external (e.g. gravitational forces or contact forces) or internal (e.g. joint limits). Further, we summarize the three essential parts of a physics-based character control framework: physics-based character, controller and physics simulator.

Physics-Based Character can be regarded as a set of connected rigid bodies which are linked to each other via character joints. In order to make the control easier and increase simulation performance, the character models are often simplified as in Fig. 1.5. In general, rigid bodies are represented with primitive geometries such as cylinders, spheres and boxes. A joint defines and constraints the movements of the bodies, also known as Degrees of Freedom (DOF). The commonly used joint types defined after human and animal body mechanics are 3 degree-of-freedom ball-and-socket joint, 2 degree-of-freedom universal joint and 1 degree-of-freedom hinge joint. For example, while a shoulder or wrist joint can be a ball-and-socket joint, a knee is well represented with a hinge joint for a humanoid character.

The process of modeling a physics-based character includes defining kine-

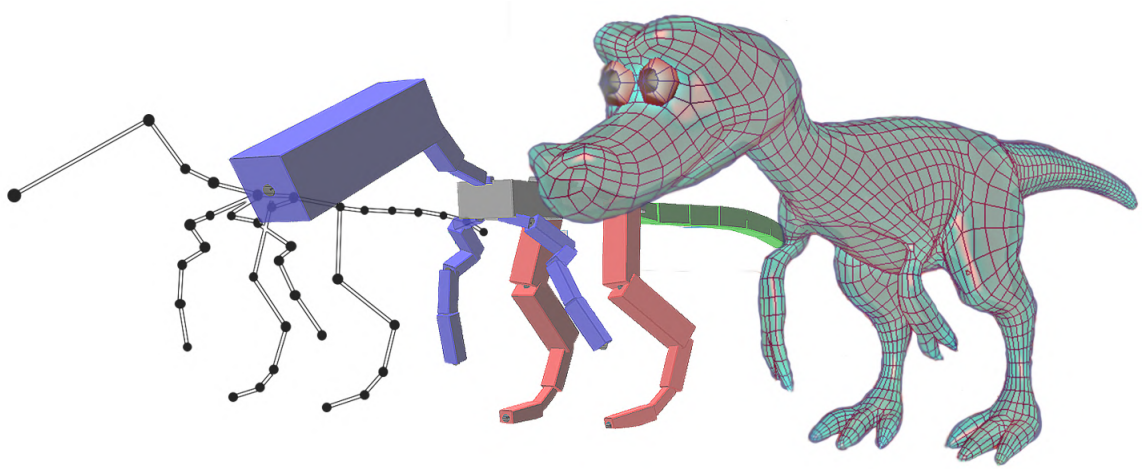

Figure 1.5: *A physics-based character model: (left) the skeletal design of the character, (middle) the construction of the character's articulated body entirely out of rigid bodies with physical properties such as mass, moment-of-inertia, (right) the mesh of the character.*

matic and physical properties of rigid bodies and joints, that is, providing correct mass/density and geometry information (e.g., length and width of the body) for each body, as well as limits for the joints. While the length of a rigid body is chosen (or automatically defined) according to the anatomical design of the character's skeleton, the width might be associated with the mass of the rigid body. The mass is an important property as it determines how quickly it reacts to the external forces applied on it, such as gravity. Another essential property of a rigid body is the moment of inertia which is calculated from the rigid body density, how the mass is distributed relative to its centre of mass. It defines the resistance of the body to angular acceleration and affects the total angular momentum of the body. Taken together, the mass distribution of a character has an impact on the overall motion of the body and balance.

Physics Simulator is the fundamental requirement for physics based animation. It updates the state of simulated objects in the virtual environment at each simulation step using Newton-Euler laws of dynamics. The update of rigid bodies is performed in two steps in a physics simulation: *Forward dynamics*— linear and angular accelerations of each object are computed based on the internal constraints and external forces applied on it; *Numerical integration*— positions and velocities of each objects are calculated based on accelerations. Natural reactions occur during and after collision of objects in real life, but collision issue needs to be handled in virtual world. There-

fore, *collision detection* which determines whether two or more objects penetrate or are in contact is an important part of the simulator. Bullet [bulletphysics.org], ODE [ode.org], PhysX [nvidia.com/physx] and Newton [newtondynamics.com] are some well-known and commonly used physics engines.

Motion Controller is the part that is responsible for generating joint torques in a physics-based character control framework, which then fed into the physics simulator. Fig. 1.6 depicts a general architecture of a physics-based character animation and control system. A motion controller determines the joint torques based on the goal of the character's motion. Hence, it can be dedicated to specific tasks such as balance and locomotion and can adapt a purely physical approach inspiring from successful biomechanics models like *Inverted Pendulum (IP)* [Kenwright, 2010]. Notwithstanding, a motion controller can also be modeled specifically to track a reference human motion.

A motion controller allows a user to interact with physics-based characters via control parameters, such as desired body speed, direction and posing (e.g., positioning of the character's end-effectors). Besides allowing user to control the speed and direction of physically simulated character's motion, it can also allow users to control a full body pose of the character from a motion capture data.

The earliest and simplest control algorithm for pose tracking is *Proportional Derivative (PD) Controller* [Hodgins et al., 1995]. The PD controller of a joint can generate torque to track the desired joint angle coming from the reference motion by minimizing error between the current pose and the target pose. It mimics a spring-damper system, where a force is generated by a spring to move to its rest position. However, the downside of this approach is that it has no knowledge on the underlying equations of motion and the character's physical properties, which can cause the an accumulated error and unpredictable results. Furthermore, it requires fine tuning of the PD control parameters for every different characters and motions.

Alternatively, methods that directly integrate multi-body (character) dynamics into constrained optimization can ensure a better and accurate results for motion tracking [Abe et al., 2007]. In this approach the equations of motion representing the dynamics of the character is incorporated into the optimization as a constraint, while the desired control for the character is defined through objectives. Then the joint torques are calculated and updated online at each simulation step. However, the biggest drawback of

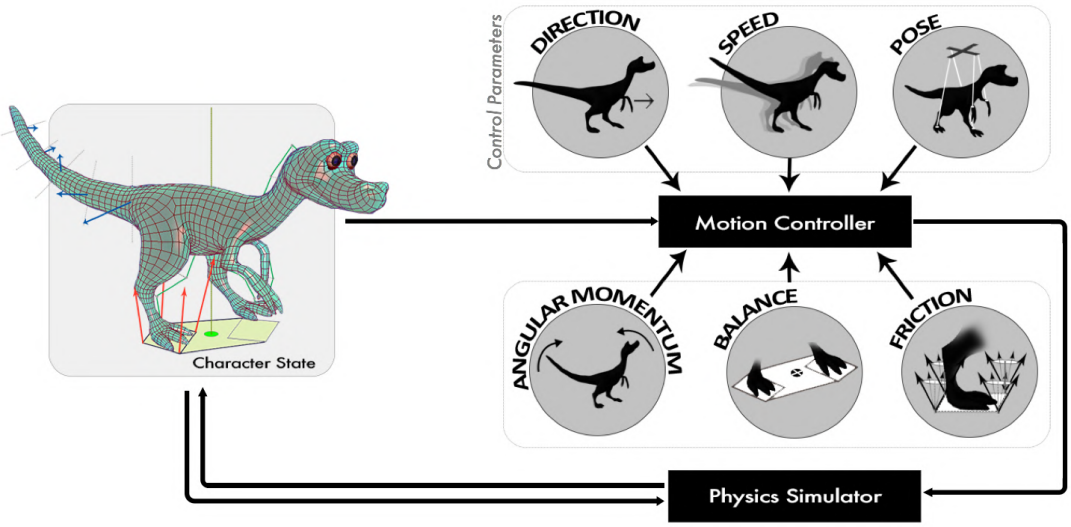

Figure 1.6: *A single simulation step of a physics-based motion control system: the controller is provided with information about the character's state (e.g., joint orientation, centre of mass (COM), ground reaction forces (GRF)) and control parameters from the user (e.g., desired speed, direction, end-effector positions). It, then, generates forces and torques for the character's joints considering a set of physics-based constraints, such as for angular momentum, balance and friction regulations. Finally, this actuator data is fed into the simulator to estimate the new character state*

these methods is their difficult implementation which require animators to have knowledge on multi-body dynamics.

A physics-based motion controller naturally also include regulators exploiting features inspired from biomechanics for improving the balance and stability of the character's motions. For instance, a balance regulator (or objective) can be responsible for keeping *Center of Mass (COM)* of the character directly within the convex polygon defined by the outline of the its feet—known as support polygon. *Angular momentum (AM)* is another important aspect for balance, employing full body regulation in the presence of disturbances due to ground forces and gravity, or other external forces [Macchietto et al., 2009]. Role of an AM regulator can be controlling the the rate of change whole body AM about the COM. Furthermore, another regulator can ensure that the resulting Ground Reaction Forces (GRFs) is within and do not exceed friction limits.

Interactive AR Characters

Interactive virtual characters play a significant role to increase the immer-

siveness in both virtual and augmented realities. In contrast to the traditional characters whose behaviors depend on virtual inputs from a mouse and keyboard, intelligent characters in AR can be controlled with new interaction channels by exploiting qualities of the physical environment. Naturally, the realism of the motions during interactions with the environment greatly increases the believability of the character and the sense of its coexistence in our world.

In virtual or augmented reality applications including games and storytelling the motion trajectories of a virtual character are traditionally handcrafted by skilled animators with key frame animation. The pipeline consists of the artist setting up a skeleton and a rig for the character model. The skeleton, which is a hierarchical set of interconnected bones, is used to position the character in poses. The animation is created by interpolating between character poses that are saved into key frames. Since creating these animations is a labor-intensive task, animating characters by adapting captured performances from human actors is a commonly used alternative. However, the ability of using captured human movements introduces a key challenge— motion retargeting especially for animating non-humanoid characters. In addition, animation models solely utilizing existing motion dataset (either captured or hand-crafted) have a disadvantage of being restricted by the contents of the database.

Animation becomes even more challenging during interaction with the user and real objects in AR; autonomous virtual characters may need to react to unpredictable user interactions and adapt their behaviors accordingly. Overall, the realism of a character's motion during interaction with a user depends on several aspects; for example, its awareness of the real world and environmental attributes including the user's movements, its responsiveness to the dynamically changing environment and the appropriateness of its motions regarding the context of interaction. To sum up, we can investigate the animation model of an interactive AR character from the perspective of the character model itself, its perception of the environment and interaction with it, as in Fig. 1.7.

Character Realistically modeled motions respect the body structure and dynamics of the character. Otherwise, any anomaly in the motion can distort the existence the virtual character in an alternate reality. The anomalies can be implausible or repetitive movements and unresponsiveness to perturbations. They are usually associated to the generated motion's disrespect of the laws of physics. Moreover, characters can also be non-anthropomorphic where their limb proportions and even topology may be different from hu-

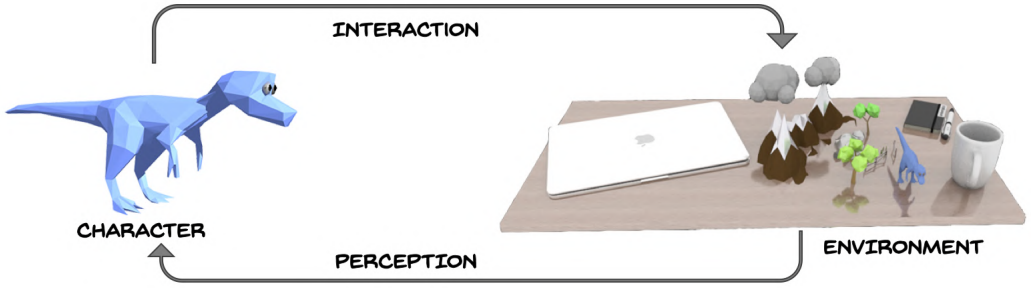

Figure 1.7: *An intelligent AR character can draw their perceptions from the state of virtual and real objects. The behaviour of the character include interacting with and reacting to the real (natural) environment, extending the communication with the user.*

mans. Nevertheless, a virtual character possessing physical characteristics of motion as much as possible and respecting mechanical constraints of its body has a high level of embodiment in the real world.

Perception Perception of the environment is an integral task for the character in order to interact with it. A virtual character's perception of its environment depend on the type of interaction which can happen between the character and the objects in the environment, or the user and the other virtual characters. A character's awareness of the physical entities in AR consists of tracking their kinematic features from visual sensor like camera, e.g., objects' position and orientation; or the user's skeletal pose.

Interaction Interaction, which is coupled with perception, depends on the character's capacity of interactivity. The capacity is defined as the character's ability of conveying natural reactions to the environmental changes. These changes can occur due to a user using its physical space and changing positions of the real objects in the physical world, or certain behaviors of other virtual objects in virtual world. In addition to the physical validity of the character's motions (e.g., joint trajectories) and skeletal poses, the responses of the character to the interactions like collisions in AR should have a regard for physical factors, such as force and mass.

1.2 Challenges

A virtual character is an effective means to involve humans in interactive environments; performing as a guide, tutor, actor, or even an adversary, virtual

characters play a significant role to enrich a user's experience. The overall goal of this thesis is to develop motion models for virtual characters with different morphologies that are able to interact with a user as well as other elements in real world space specifically utilizing augmented reality technology. Naturally, motion models are crucial to increase the believability of the characters during their interactions with human viewers. Even though great advances in AR technology and their accessibility with easy-to-reach digital devices opened up possibilities for the direct physical interactions between users and virtual characters, there are a set of challenges to tackle:

- Traditional key-framing is not only a labor-intensive task, but also not suitable for interactive applications where unexpected environmental changes and accordingly character behaviors are met. In addition, exploiting solely human mocap data to animate characters results in well known artifacts like foot-skating, shape interpenetration, or simply a perceived lack of realism. As an alternative, physics-based approach generates motions online that are responsive to non-predefined environment and user inputs, making it better suited for the AR and other interactive applications. However, using physics-based approaches are complicated as they require at least some knowledge of multi-body dynamics, control and optimization theory. Existing methods use intense preprocessing step with manual parameter tuning or offline parameter optimization specific to the character.
- The approaches for physics-based character control are specific to character's shape and task. That is, the same motion cannot be performed exactly with a new character because of the different physical characteristics (e.g., distribution of mass over the body). In addition, we can expect a virtual character interacting with the user in various scenarios to have different body shapes including non-humanoid bipedal or even quadruped. This also applies to the cases that a motion controller inferred from a human performance data. This specific interaction case where a virtual character is mimicking the movements of the user like a puppet requires a motion model that is general enough to adapt the motions performed by an actor to a similar motion performed by the character. Therefore, motion retargeting is another problem to address.
- Physics-based character animation is computationally much more expensive than kinematics-based alternatives. Computational efficiency in an AR environment is critical since a virtual character must create responsive motions in real time to the dynamically changing

environment. Therefore, interactive characters in augmented reality settings require motion models incorporating physical constraints to character's animation models while balancing computational cost with kinematic strategies. Overall, animation models for AR characters need to be robust to different character types, perturbations from user and other objects from the interactive environment as well as giving emergent and physically plausible responses as much as possible.

- In order for the virtual characters to interact in an intelligent and realistic manner, their understanding the real-world environment is crucially important. The real objects and the user that are part of the physical environment are the elements for the character to interact within AR. By taking advantage of the cameras and leverage state-of-the-art computer vision and machine learning techniques for data acquisition from real world, a virtual character should be equipped with a perception that can recognize and track poses of the objects and the user. While an 3D object recognition followed by an 3D reconstruction is required for interactions with real objects, a 3D pose estimation is necessary for the virtual characters to interact with the user directly.

1.3 Contributions and Thesis Outline

The work with the resulting contributions presented in this thesis are slightly diverse due to the exploratory nature of my research path. In essence, our goal is to a large extend to leverage interactivity between users and digital characters using AR. We focus on the animation models for interactive and intelligent AR characters connecting several exciting research topics: augmented reality, physics-based character motion, artificial intelligence and image processing.

Taken together, AR provides an environment that virtual characters would seamlessly be integrated into reality, so that the interaction with users and physical objects in the environment itself would be in the most natural and intuitive way. Modeling the characters and their behaviors using physics-based solutions makes the characters interactive and responsive to the user input and environmental stimuli like force and terrain changes. Finally, object recognition and pose estimation techniques completes the character's abilities to perceive the real world.

Further in this section, a brief introduction to our contributions in the aforementioned domains are outlined.

1.3.1 Physics-Based Character Control Interface

Driving behaviors of a virtual creature using human motion data is quite daunting task when the creature's shape and morphology can significantly differs from human body structure, as is often the case, because it requires motion retargeting in real-time. Even though the motion capture is used to communicate a large portion of the motion, using solely traditional inverse kinematic techniques cannot synthesize a motion that respects the structure and dynamics of the imaginary character as well as leaving free limbs, such as tails, without motion, which are not possible to map from human body.

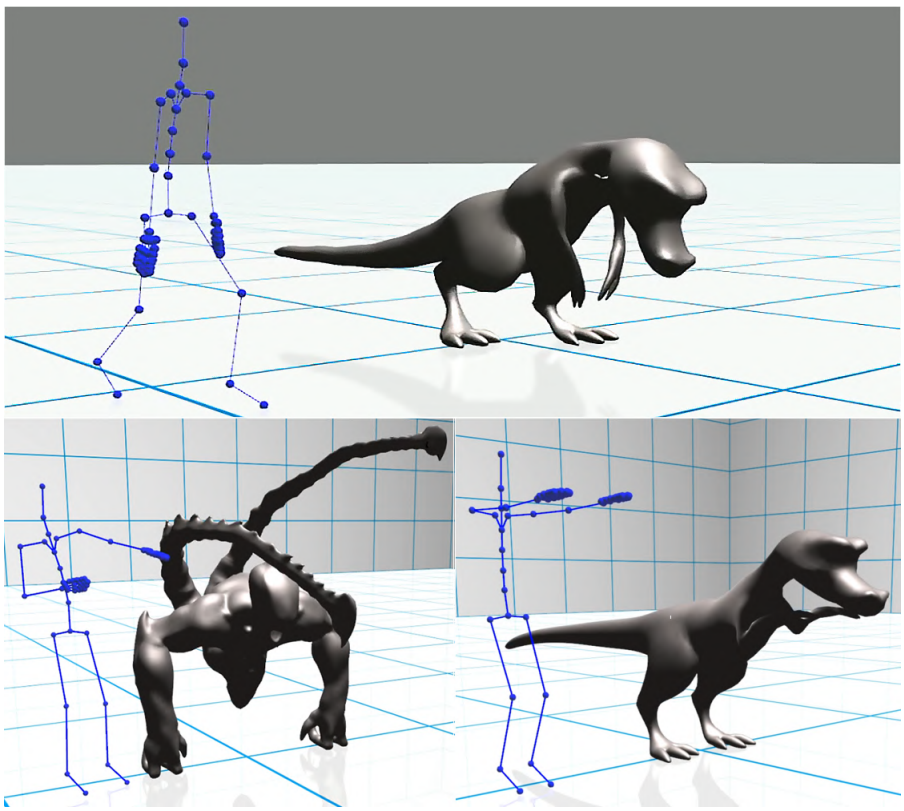

Figure 1.8: *Example frames showing resulting character animations together with the corresponding performance animations obtained with our physics-based character control system.*

We developed a physics-based character control interface that allows a virtual character to track a human actor in real-time. The biggest drawback of the physics-based character control is its complex nature which requires

knowledge of dynamics and control theory. We address this challenge with an intuitive interface that allows both quickly designing the mechanical system of any character (e.g., dinosaurs) as well as setting the controller parameters automatically via a limb-based abstraction. The main goal is enhance the generality and accessibility of this simulated control mechanism to novice users while ensuring physically accurate and emergent character motions. In addition, the system can be extended to be used in AR applications because the generated character animations at interactive rates (See Fig. 1.8).

1.3.2 Interacting with intelligent AR Characters

Thanks to augmented reality technology, virtual characters can be in a heterogeneous environment combined with virtual and real objects, but the full potential of interactions between them has not been fully explored yet. Having characters capture the context of the user's environment and react with an awareness of physical interactions with real object and a user remains a challenge.

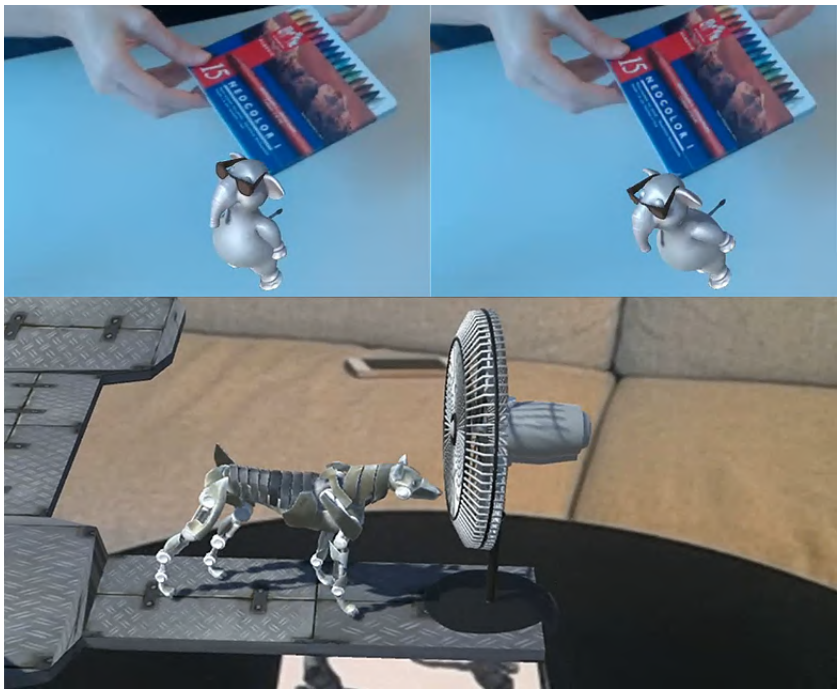

Figure 1.9: *Examples of user-AR character interactions in our AR framework with both real and virtual objects.*

In this work we develop an AR system which allows investigation of different natural physical interaction models between intelligent characters and

real-virtual objects. Our vision is seeing animated AR characters walking around on your table, adapting their motions to changes in environment (e.g., slopes made up of real-world objects) and reacting disturbances in their surrounding (e.g., pushed away by real book or stopped by a fan prop), as shown in Fig. 1.9.

To enable a character to intelligently react to its environment, we introduce a character animation model composed of four main layers— parametric locomotion modeling, kinematic motion control layer, IK-based terrain adaptation, ragdoll physics layer— from bottom to top, respectively. Given only the skeleton of the character to be animated as input, the parametric locomotion model automatically generates periodic motion segments necessary for the character's locomotion. We ensure a physical feasibility in the generated motions by integrating several physical constraints (e.g., full-body torque regulation and friction) in the optimization of motion parameters together with a set of kinematic objectives which still give users a flexibility to define motion style. Kinematic motion control unit is responsible for animating the character in real-time. Finally, an IK-based terrain adaptation and short-lived ragdoll physics approach for handling perturbations are layered on top.

We use an approach for the reconstruction of the 3D environment by utilizing off-the-shelf scanning solutions and Vuforia's image recognition technology [Vuforia, 2017] with pre-defined objects that the character can recognize and perform accordingly.

1.3.3 AR Selfies

Incorporating digital avatar capabilities into entertaining scenarios in AR — *AR Selfies*— unlocks new possibilities for communication and interactions with virtual characters, considering Snapchat [SnapInc, 2018], which heavily relies on AR. Inspired by this, we developed a framework, AR Poser, that augments a person in a camera image with a digital character imitating the person's pose in a selfie setting through augmented reality.

To enable a digital character to interpret and reproduce the pose, we propose a solution for 3D pose estimation approach that relies only on the RGB information from a monocular image. A depth camera, such as a Kinect [Shotton et al., 2013], is an alternative for 3D pose estimations, but not as widespread as monocular cameras on mobile devices and not always reliable in outdoor conditions. Our approach utilizes the latest advancements of deep learning for 2D pose estimation in the wild, combining with a projection onto a small

Figure 1.10: *(Top) AR Poser: 2D pose estimation of the person, followed with its augmentation with a digital character. (Bottom) AR Costumes: 3D costume pose applied to person's pose (middle column), leaving several parts of the person visible. After shape estimation together with in-painting of the person's body, we manage to fit costume without artifacts.*

3D pose space, assuming selfie scenario, to find the optimal matching 3D pose for the character— which enables mobile phones to be used.

Once body pose estimation from monocular RGB image is achieved, it is possible to augment digital costumes onto the person taking selfies. Following to *AR Poser*, we described a new method to fit 3D *AR Costumes* onto the person's pose that is compatible with mobile devices, as shown in Fig. 1.10. For this, we needed a more precise 3D pose estimation to match 3D costume shape closely in pose and proportions. We achieve this with an additional refinement optimization that adjusts the global scale based on root position and exactly matches the limb directions with small local modifications. Still, the overlaid costume shape may leave cloth or skin of the person visible behind it. To remove this artifacts we use image processing techniques including 2D masking using *Grabcut* [Rother et al., 2004] and *Inpainting* [Guillemot and Le Meur, 2014].

The remainder of this thesis is organized as follows. The next chapter, Chapter 2, discusses a literature review in the fields of motion retargeting, performance tracking with simulated characters, augmented reality (AR) and interaction with characters in AR and human body augmentation. Chapter 3 describes a framework for controlling morphologically different, physics-

based characters in a simulated environment, based on performance tracking in real-time from a human actor. The framework enables virtual characters to mimic the characteristics of the performer's motions, while maintaining physical correctness. We start by describing the intuitive user interface for modelling the character's body and associating it to human actor's skeleton via a limb-based abstraction. Further, we lay out the details of the motion retargeting process together with the control objectives and constraints. Chapter 4 explores the ways of interaction between users and intelligent AR characters via a motion model combining kinematic and physics-based approaches. We show examples of interaction with a quadruped AR character adapted to real-world conditions. The subsequent Chapter 5 presents the frameworks *AR Poser* and *AR Costumes* that investigates the problem of 3D pose estimation and augmentation in an entertaining setting— *AR Selfies*. Finally, Chapter 6 concludes the thesis with a summary and discusses limitations along with potential directions for future research.

Publications Over the course of three years, the following peer-reviewed publications have been accepted:

[Çimen et al., 2017] G. CIMEN, M. GUAY, S. COROS and R. SUMNER. An Intuitive Interface for Human Performance Tracking with Simulated Characters. *11th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing*, 2017.

[Cimen et al., 2018b] G. CIMEN, Y. YUAN, R. SUMNER, S. COROS and M. GUAY. Interacting with Intelligent Characters in AR. *Proceedings of the 1st Workshop on Artificial Intelligence Meets Virtual and Augmented Worlds (AIVRAR) in conjunction with SIGGRAPH Asia*, 2017.

[Cimen et al., 2018a] G. CIMEN, C. MAURHOFER, R. SUMNER and M. GUAY. AR Poser: Automatically Augmenting Mobile Pictures with Digital Avatars Imitating Poses. *12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing*, 2018.

[Maurhofer et al., 2018] C. MAURHOFER, G. CIMEN, M. RYFFEL, R. SUMNER and M. GUAY. AR Costumes: Automatically Augmenting Watertight Costumes from a Single RGB Image. *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 2018.

[Casas et al., 2018a] L. CASAS, L. CICCONE, G. CIMEN, P. WIEDEMANN, M. FAUCONNEAU, R. SUMNER and K. MITCHELL. Multi-reality Games:

An Experience Across the Entire Reality-virtuality Continuum. *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, 2018.

CHAPTER 2

Related Work

This chapter lists the related work on kinematic motion retargeting techniques and physics-based characters in the context of performance tracking (Section 2.1), Augmented Reality (AR) technology and their applications in the domain of interaction between users and virtual characters (Section 2.2), and human body augmentation within the concept of *AR selfies* (Section 2.3).

2.1 Performance Tracking with Physics-based Characters

This section starts with a summary of skeleton-based motion retargeting solutions that are mostly rely on Inverse Kinematics (IK). Following, we discuss a related subject— motion puppetry— which is based on a mapping of pose correspondences between source human pose and target character pose (see Section 2.1.2). Finally, the remaining subsection, Section 2.1.3, presents previous work on physically simulated character animation and physics-based motion retargeting, respectively.

2.1.1 Motion Retargeting

For the cases when a target character has different size and structure, naively transferring only the joint angles results in a violation of constraints, such as the character's body intersecting with itself or the environment (e.g., ground). Therefore, in a scenario like performance tracking where an actor guides the movement of a virtual character through his/her performance, motion retargeting is necessary.

First motion retargeting techniques emerged to adapt and reuse motion data (i.e., motion capture, key-framing) created for one character on other characters. When the source and target character have different body sizes and proportion (e.g, arms, legs), it is not possible to apply a mapping to directly transfer joint angles to the character. Therefore, the common approach used in the prior work on motion retargeting is to adapt a feature-based inverse kinematics (IK), where a set of important motion features are formalized as constraints. The constraints tell the optimization to preserve the important qualities of the motion.

[Gleicher, 1998] first addressed the motion retargeting as a spacetime optimization problem, where a set of kinematic constraints, the spatial-temporal relationship among body segments and the environment (i.e., the feet must be planted when in contact with the ground), are specified via a preprocessing step. [Choi and Ko, 1999] adopted an online motion retargeting approach that uses inverse kinematics based on the Jacobian. Their approach can imitate the joint angle changes from the reference motion while tracking a set of end-effector positions by utilizing the kinematic redundancy of the articulated figure. While Choi et al. pointed out that the end-effector positions are more important motion features than joint angles, [Shin et al., 2001] argued that what is important can change by the context of the motion. [Tak and Ko, 2005] further integrated dynamic constraints into their constrained-based motion retargeting technique to ensure physically plausible motions. They applied sequential filtering that optimizes the pose on every single frame based on the kinematic and dynamic constraints designed by the animator.

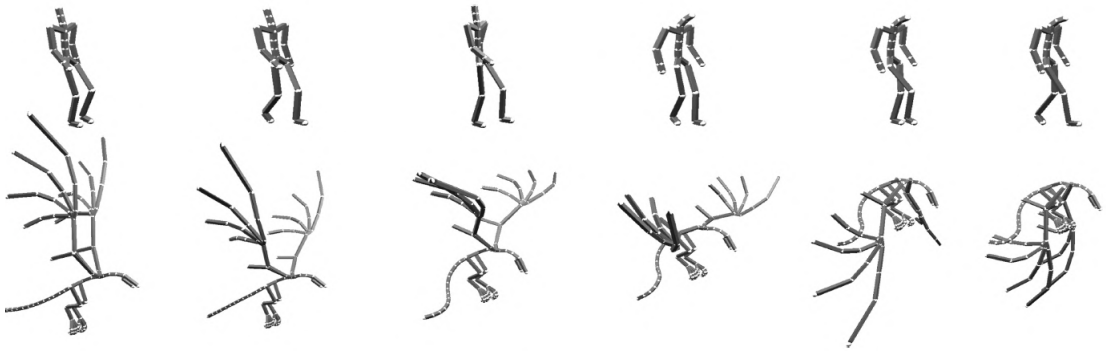

Figure 2.1: *An example of motion style transfer from a neutral to a sad emotion taken from human motion sequence (top) and retargeted to a dragon character's motion (bottom) [Abdul-Massih et al., 2017]*

.

When the characters have different topologies (i.e. different joints hier-

archies), IK based approaches follow a strategy to only consider the constraints on end-effector positions using an intermediate skeleton [Monzani et al., 2000] or simplified representation of motion [Kulpa et al., 2005] that is morphology-independent. The intermediate skeleton proposed by[Monzani et al., 2000] has fewer degrees of freedom (DOFs). This enables mapping important topology invariant features from source human motion to the target character, such as end effectors and the remaining DOFs analytically. [Kulpa et al., 2005] uses a character representation where limbs are represented as half planes and spine as spline.

Recently, [Abdul-Massih et al., 2017] focused on the transferring motion style between different morphologies using a concept called Groups of Body Parts (GBPs). GBPs are kinematic chains manually defined by a user, extracted from the source character, mapped to the target character and later transformed into constraints in a full-body optimization during retargeting. They demonstrate various style retargeting examples transferred from a human to a non-humanoid character like a dragon or a T-Rex (Fig. x). The key idea of this approach is mapping certain kinematic chains from the source character's skeleton to the corresponding kinematic chains to the target character.

For these aforementioned motion retargeting techniques, someone should decide for the cases when the target character has a tail or other extra limbs whose motion cannot be directly retargeted from human source motion data— either associating the tail to a human body part, such as head of the human performer, or not associating at all. The former will create unrealistic motions for the tail and the latter will leave it motionless.

2.1.2 Motion Puppetry

Motion Puppetry is similar to the traditional skeleton-based retargeting as it specifically tries to solve the problem of transforming the movements of a performer to an animated character. While in skeletal-based retargeting, human joint data are mapped to the character, in motion puppetry, users can control the motion of a virtual character not necessarily by defining a joints-to-joints mapping. When the virtual character has a topology which is very different from human skeleton, skeleton-based retargeting requires complex process. On the other hand, motion puppetry suggests solutions to simplify or even bypass the skeleton-based character retarget process and animate mostly non-humanoid characters, such as spiders or caterpillar.

Several researchers have investigated motion puppeteering. One of the early motion puppetry system is developed by [Shin et al., 2001]. The proposed

IK solver can realize the important aspects of the motion by employing importance sampling and keep these features preserved during mapping to the target character. For example, end effector positions that are in proximity to other object or ground considered important than others. However, this method supports only identical topology and connectivity between the performer and articulated figure. Later, [Dontcheva et al., 2003] developed a layer-based animation authoring system that allows animating an arbitrary character by synthesizing and editing motion in layers through a widget. Editing happens by assigning the widget motion to one or more character features, such as DOFs and relative handles (controlling points used in IK) on the character's skeleton, and mapping its reference frame to the reference frame of the camera view. [Hecker et al., 2008] developed a different animation authoring system for the same purpose. With the proposed animation authoring system, they brought the animators into the retargeting process itself and allowed them authorizing motions using semantic specifications in task space instead of joint space. The goal of their solution is not solving exactly the motion retargeting problem itself- mapping the motion created on one body to another, but rather representing the motion from the beginning in such a way that it can easily be transferred to a wide range of topologically different characters.

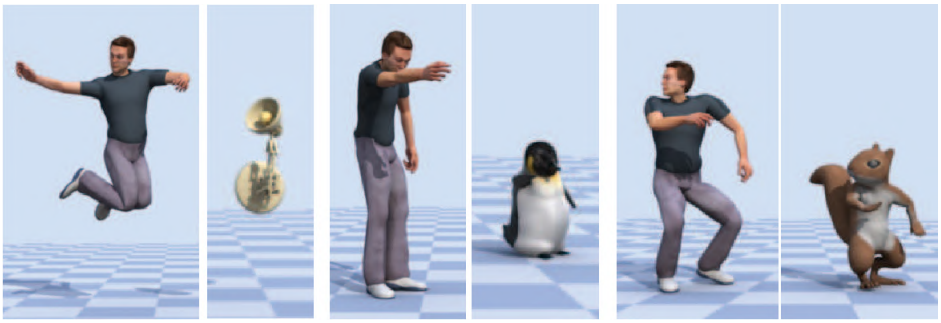

Figure 2.2: *A computer puppetry approach to animate non-humanoid characters, such as Pixars Luxo lamp [Yamane et al., 2010]. It requires the user to manually define 30 to 50 pose correspondences.*

Several approaches learn mapping a human pose to a character pose from a small set of sparse pose-to-pose correspondences. [Yamane et al., 2010] applied shared Gaussian process latent variable models (shared GPLVM) to learn a mapping function to animate non-humanoid characters from human motion capture data. To improve the physical realism, they apply an additional dynamics optimization based on the equations of motion. [Vögele et al., 2012] developed a method to animate deformable quadruped characters in real time by mapping from the skeleton motions of two humans. For example, one human skeleton is mapped to the front half of a horse

while the other skeleton to the back. [Rhodin et al., 2014] similarly establish a mapping between a human skeleton to a target character mesh with as few as 4 interactively-defined sparse pose correspondences. While Yamane et al., provided an offline approach, Rhodin et al., employ linear mapping to enable real-time control. Similarly, in the approach developed by [Seol et al., 2013], a correlation mapping between features of a human actions and a character's motions. However, these systems relies on some predefined animations, such that an actor needs to to carefully match the pose of the target character. Otherwise new character animations need to be deployed.

2.1.3 Physics-Based Characters

The physical implausibility is the main problem for performance tracking with virtual characters that have significantly different topologies (i.e. different joints hierarchies) and do not move in the same way with the human actor. For example, the legs of a dinosaur do not bend the same way as the legs of human, and the free limbs humans do not possess—such as a tail—remain static.

Tracking with physically simulated characters tackles these issues as it automatically yields natural and consistent motions. Motion capture driven simulated characters are first introduced by [Zordan and Van Der Horst, 2003]. The motion capture data is tracked through a mapping that attaches virtual springs between the mocap markers and the character's joints, and resistive torques applied to the character using a simple controller.

Even though, physically simulated characters has been well investigated across numerous specific tasks, such as balancing, standing, walking, etc., no doubt great amount of work has been released in walking and balance strategies [Yin et al., 2007, Tsai et al., 2010, Lee et al., 2010, Coros et al., 2010] and extending them to mimic the style in the motion [Kavafoglu et al., 2018]. Typically, a set of locomotion poses are tracked with Proportional Derivative (PD) control of target joint angles, and a balancing strategy based on the inverted pendulum (IP) model is used to adjust the target poses. While these controllers are very robust and don't need equations of motion, they can only track specific motions, such as locomotion, and do not extend easily to other types of motions.

Another line of work—so-called online optimization controllers—consider the control at a single time step and recompute the torques each time via inverse dynamics; assuming knowledge of the equations of motion ([Abe et al., 2007, Da Silva et al., 2008, Macchietto et al., 2009, Mordatch et al., 2010,

de Lasa et al., 2010, Levine and Popović, 2012]. When the equations of motion of an articulated rigid body system are expressed in generalized coordinates, a linear relation between joint torques and joint accelerations can be established for a single time step. As a consequence, it becomes possible to minimize a collection of quadratic objectives under the hard linear constraint that the equations of motion hold. [Abe et al., 2007] focused on motions that remained in balance (static contacts) and subsequent works focused on planning and engineering features for various motion skills ([Da Silva et al., 2008, Macchietto et al., 2009, Mordatch et al., 2010, de Lasa et al., 2010], or relaxing physical realism for robustness [Levine and Popović, 2012].

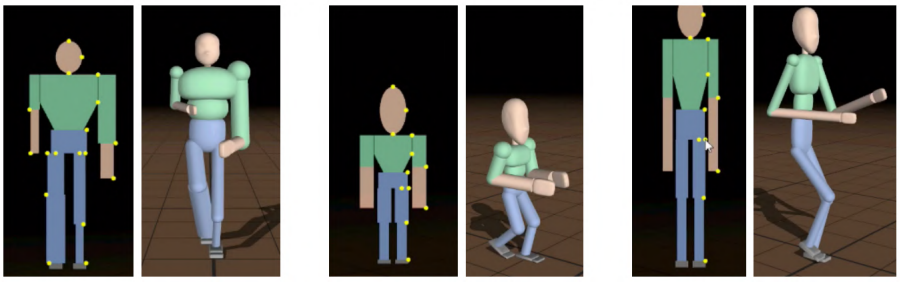

Figure 2.3: *The physics-based biped walking control system presented by [Coros et al., 2010] that allows Interactively editing of character proportions.*

Even though controlling a simulated character to track different motions has been well studied, it still remains a challenge to go from a kinematic –strictly positional– signal, to a controlled motion. In addition, it requires a scientist or engineer to design controller parameters for each new character or motion which makes physics-based motion retargeting complicated. Therefore, only a few prior works placed the control of simulated characters into the hands of casual users. [Coros et al., 2010] allowed the user to change the body proportions of the character, resulting in different motions. Unfortunately, their method is specific to human locomotion. [Levine and Popović, 2012] used the bone lengths to initialize the rigid bodies, but relax the physical accuracy by allowing root activation. With their method, the character can only track the same character, and there is no retargeting or free limbs activation. Notwithstanding, there is no technique that is capable of tracking human actors with different characters as well as different body parts associations.

2.2 Interacting with Virtual Characters in AR

This section begins with a brief history of development process of today's AR technology in mobile devices. Then, Section 2.2.2 lists the previous

works on creative and interactive AR applications, especially in the context of games that mixing realities from Milgram's Reality Virtuality (RV) continuum [Milgram and Kishino, 1994]. Lastly, in Section 2.2.3, we discuss the related research on interactive AR characters and agents.

2.2.1 A Brief History of Augmented Reality

> *"It consists of this pair of spectacles. While you wear them every one you meet will be marked upon the forehead with a letter indicating his or her character. The good will bear the letter 'G,' the evil the letter 'E.'"*
>
> – L. Frank Baum, *The Master Key, An Electrical Fairy Tale*

The earliest recorded reference to the concept of AR is made by L. Frank Baum in his short science-fiction novel 'Master Key' in 1901. In the story, a demon gives a gift to the main character, called *Character Marker*. The gift is a pair of electronic glasses that maps data on people— a lot like today's augmented reality headsets.

The concept of augmented reality can be traced back to the 1960's as the first head-mounted AR system is created by Ivan Sutherland in 1968 [Sutherland, 1968]. It was so heavy that it had to be hung from the ceiling and limited to displaying graphical wireframe models, though, it was the first step in making AR possible for users. The term of Augmented Reality, however, was coined later in 1992 by a Boeing researcher, Tom Caudell [Caudell and Mizell, 1992]. Then, Ronald Azuma clearly defined the term, Augmented Reality in 1997 [Azuma, 1997]. The usage of AR remained limited to scientific laboratories until 1999. Then, it gained momentum when Hirokazu Kato released ARToolkit [Kato and Billinghurst, 1999] to the open source community. ARToolkit was a software library that uses pose tracking from video capture to calculate the real camera position and orientation in real time relative to physical fiducial markers. 3D computer graphics model could be drawn to overlay the markers according to a virtual camera placed at the same exact position of the real camera. It allowed researchers and other enthusiastic people to build AR applications for any handheld device with a camera and an internet connection.

After 2000, the advances in mobile device capabilities introduce a strong drive towards mobile applications. In 2004, Möhring et al. developed the first video see-through augmented reality on a cell-phone tracking 3D markers [Mohring et al., 2004]. Later, the first truly usable natural feature tracking system for smartphones is introduced in 2008 [Wagner et al., 2008] which is

the precursor of the popular AR development toolkit— Vuforia [Vuforia, 2017]. Over the next ten years, the research and technology in AR is in rapid expansion. The most popular example of AR applications are Snapchat and the Pokémon go introduced in 2016. Throughout 2017, two more AR developer tools are released— ARKit for iOS mobile devices (iPad and iPhone) by Apple and ARCore for Android by Google.

According to [Chatzopoulos et al., 2017], a successful mobile AR system should enable users to focus on application rather than its implementation. Beyond the ability to seamlessly supplementing real world with digital enhancement, heightening interactivity is also crucial in enhancing the user experience in AR.

2.2.2 Interactive AR applications

The applications developed over the years are in the form of educational and creative games that user can experience the different parts of the spectrum of reality. Gradually, advances in the development of mixed reality have made games possible to take advantage of the entire spectrum of realities and converge them into a single application. It was the early 2000s when we began to see AR take its place media applications, particularly in education and entertainment. The world's first outdoor AR game, ARQuake— an AR adaptation of the popular Quake game— is launched [Thomas et al., 2000]. Besides a head-mounted display, players had to wear a backpack containing a computer. The player movements in the game are achieved with user's pose estimation via the combination of inertial sensor data (e.g., digital compasses, GPS) and visual tracking method using ARToolKit Library while the actions are performed by a simple two-button input device.

One of the early examples of AR technology in the field of education is the MagicBook [Billinghurst et al., 2001]. Large markers are integrated into a book's pages, which enable seeing 3D virtual content out of it through VR glasses. The user can also switch into an immersive VR experience and move into a scene and interact with characters. Later, [Grasset et al., 2008] developed a mixed reality book that incorporates various visual and auditory effects to the pages to enhance the reader's immersion. The Haunted Book [Scherrer et al., 2008] is a prime example of well-integrated AR content. The camera is mounted on a lamp on the table and the augmented book is viewed through a computer screen. Their focus lies on interaction between the virtual content and the physical book.

Following the AR magic books, interactive AR coloring books allowed users to create 3D virtual pop-up content by coloring the pages of a real book and

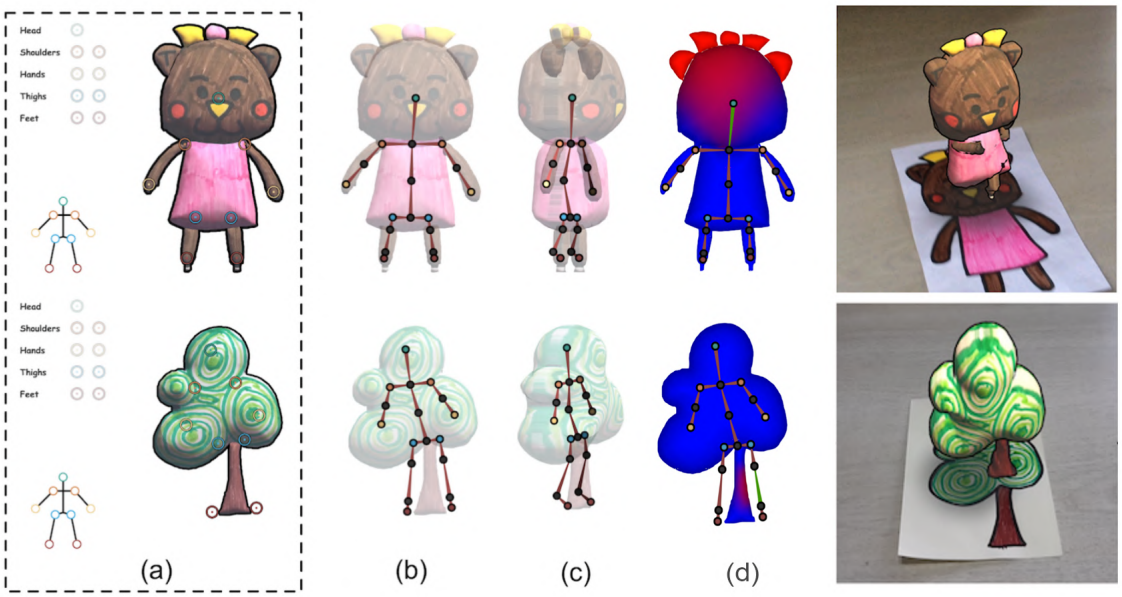

Figure 2.4: *The skeleton embedding and skinning in MagicToon [Feng et al., 2017], which allows creating 3D characters from 2D drawings. The user can define several joint positions (a). Then the system embed predefined human skeleton (18 joints) on the model (b)(c) and apply skinning using heat diffusion method (d).*

interact with it. [Clark et al., 2011] presented the first AR book that can map the colored pages of a book to virtual models. Later [Magnenat et al., 2015] extended the experience with alive AR characters with predefined animations making the coloring process in real time. Recently, MagicToon [Feng et al., 2017] added modeling pipeline into AR coloring concept allowing creating and coloring 3D models out of sketches, rigging as shown in Fig. 2.4 and animating them by interpolating between affine transformations defined by users. The system also support animating humanlike characters through a skeleton embedding and skinning process and finally attaching a set of predefined skeletal motions.

The research interest in mobile MR application that exploring various physical interactions within an entertaining game scenario was also not a new concept. One of the first examples of this exciting direction for AR gaming is The Invisible Train [Wagner et al., 2004], where the virtual trains move on physical wooden tracks. The interaction with the game environment was done by changing the path of tracks and the track switches. [Henrysson et al., 2006] developed AR Tennis game where players can use their mobile devices as an interaction tool by tilting to hit the ball around an AR-marked table. Recently, [Casas et al., 2018b] presented an adventure game that al-

low user to sequentially travel through the whole spectrum of reality while they can interact with both virtual and physical objects using only a mobile device.

2.2.3 Interactive AR Characters

Augmented reality makes virtual characters appear to coexist by superimposing them on top of the our physical world. However, the coexistence in the real environment is not enough for a strong sense of presence and life-like characters. The behavior of AR characters interacting with real world can modeled in two different ways: (1) based on a selection of predefined animations prepared for all possible situations; (2) based on observing changing changes in real environment including some objects in it and giving dynamic responses using physical interactions. Intelligent AR characters refer to the second scenario and is a relatively less explored research.

The ALIVE [Maes et al., 1995] is an early system that supports AR characters that behaves according to user interactions. In the system, a virtual dog character can behave according to users' hand gestures. In another MR system, a conversional humanoid agent, named Welbo [Anabuki et al., 2000], assists users wearing an HMD and designing a partially equipped physical living room with virtual objects. The virtual robot acts according to the user's instructions via speech recognition and interacts with virtual objects. Later [Wagner et al., 2006] developed and handheld AR application in which a virtual character teaches users about art history.

However, all of the above AR characters interact with user and merely observe the physical environment but can't be directly affected from the real objects and the changes in the physical environment. The Virtual Brownies [Aoki et al., 2005] is an early example for these kind of interactions. User can interact with AR characters, Kobitos, through the real objects. They can push around these physical objects such as a tea caddy. The real caddy moves by synchronizing its movement with a physically simulated virtual caddy according to the force applied from Kobitos. However, there is not further detail on the modeling of characters' animations and interactions.

An early example on autonomous AR characters developed by Barakonyi in a game called Monkey Bridge [Barakonyi et al., 2005] that change their behaviors based on the environmental changes. In the game, characters autonomously selects which path to walk on and which animations to play to reach a target location on the game platform. The players indirectly control the characters behaviors by changing position and distribution of the virtual

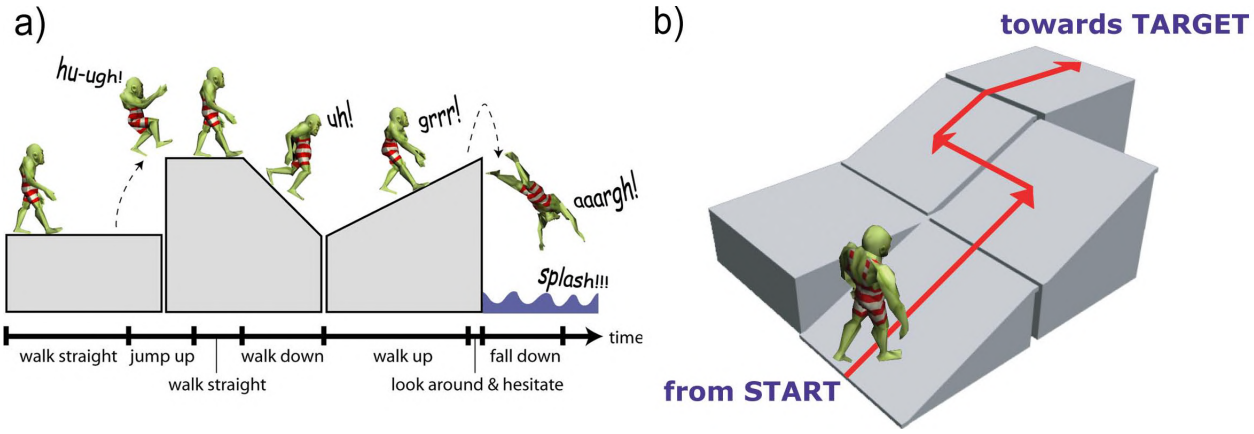

Figure 2.5: *The behavior of intelligent AR agent in Monkey Bridge [Barakonyi et al., 2005] is based on a motion planning that chooses the animation based on platform type and a path planning depending on the spatial distribution of platforms.*

and physical wooden blocks on the game board that have different shapes. Later, [Kang and Woo, 2011] explored a scenario that an AR character can pull or push a real object in his environment, a toy cart, besides giving realistic responses to manipulations of the toy in real time. The behaviors of the character is selected among a predefined character animations based on a state diagram that takes into account the movement of the toy cart and the relative position of the character to it. Recently, McIntosh et al. created a magic bench where a character appears and sit on the bench next to the user [McIntosh et al., 2017]. However, the communication of the user with the digital character is very limited as the movements of the user does not effect the pose or animation of the character.

2.3 Posing with AR Characters

In the first part of this section, we summarize the recent works on 2D and 3D pose estimation, which is the essential research topic in computer vision field and an enricher for AR applications. Followed by, Section 2.3.2 introduces the related work on gesture-based interaction techniques based on user tracking (e.g., hand and pose tracking) in interactive AR applications. Further in Section 2.3.3, we present an body of literature exists on augmentation concepts exploring around body.

2.3.1 Body Pose Estimation

For VR and AR applications, estimations of an user's 3D pose is a part of the virtual character's spatial reasoning and an important asset for the interactions between the user and the character. However, The reconstruction of a 3D human pose from an image is a not trivial task due to the ambiguities associated to the missing depth information, as well as the variations in human shapes.

With the introduction of commodity RGBD sensors in 2010, early efforts have been devoted to the pose estimation using a depth camera. [Shotton et al., 2013] introduced an approach that recognize a body part segmentation from single depth images using random forest classifiers, from which a skeleton is extracted. While their estimator used only depth-based features, a later approach combined color with depth-based segment labelling for a more robustness estimation [Buys et al., 2014]. Recently, [Zimmermann et al., 2018] presented an approach incorporating a 2D key point estimation from color images with information from depth maps using multiple calibrated Kinect devices. The downside of these methods, apart from their requirement of active depth sensing equipment, is that they need training on large number of synthetic of annotated skeleton poses and depth map pairs.

With the introduction of more powerful discriminative approaches, such as Convolutional Neural Networks (CNN), researchers experimented one-stage approaches that can recover full 3D pose from a single RGB image using CNN-based regression [Toshev and Szegedy, 2013, Pavlakos et al., 2016]. However, these data-hungry deep learning methods depends on large 3D annotated datasets and cannot benefit from large-scale 2D pose datasets which is feasible to obtain on in-the-wild data. Some introduced a new dataset [Mehta et al., 2017] combining real and synthetic data with the existing annotated 3D pose datasets [Ionescu et al., 2013], yet is hard to cover a wide range of poses.

The body pose estimation in 2D from monocular RGB has been widely researched yielding successful state-of-the-art estimators with high accuracy prediction with deep CNNs [Wei et al., 2016], even in real-time citeCao16. The key point detection in 2D have reached impressive performance since they usually generalize better on images in the wild with the availability of large scale datasets [Andriluka et al., 2014, Lin et al., 2014]. For this reason, we also integrate it into our 3D pose estimation approach.

We are not the first to consider breaking down the problem into a first 2D pose estimation step, followed by a 3D re-construction step. Some recent

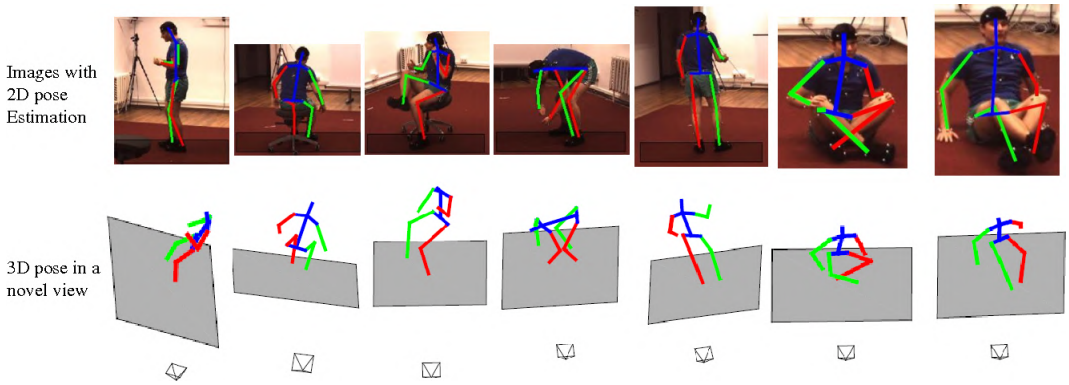

Figure 2.6: *3D pose estimation results by [Chen and Ramanan, 2016]: the estimation of a 2D pose from an input image (top), followed by 3D pose estimation as a result of matching to a library of 3D poses (bottom).*

techniques use a multilayer neural network architecture to regress 3D coordinates of the joints from their 2D locations [Tomè et al., 2017, Martinez et al., 2017]. However, the predicted poses from these approaches are in scale and translation normalized and not suitable for characters in real world units. Some methods reconstruct the 3D pose via optimization or search using a large database of poses. [Wang et al., 2014] represented 3D poses as a linear combination of a sparse set of bases learned from a large 3D pose dataset, and solve the 3D reconstruction as an optimization problem. [Yasin et al., 2015] gathered a data-set of 2D and 3D pose pairs and learned a mapping between the both in the context of 3D pose retrieval. [Chen and Ramanan, 2016] utilized a big 3D mocap library and their 2D projections from virtual camera views in order to estimate 3D pose via a data-driven matching using estimated 2D pose as query (Fig. 2.6).

Our approach is similar to the work in [Chen and Ramanan, 2016], which uses an example-based matching method with 200.000 exampler. We avoid having to gather a large dataset, and focused on a small set of poses geared towards entertaining AR selfies.

2.3.2 Human Body as Input for Interactions

With the advancements in hand-gesture tracking, pose tracking, face tracking, and eye tracking, natural interaction methods is becoming essential for the success of augmented reality. A new spectrum of interaction modalities is presented using a space in which digital characters can perceive humans' presence and react to their movements. According to [Schraffenberger and van der Heide, 2016], natural and multiple interaction modalities are needed

in an AR application as human interaction with the real world is inherently multimodal through body, face and speech. For example, human tracking input has been successfully applied to control and animate virtual character's facial expressions [Cao et al., 2014] and talking [Taylor et al., 2017]. Even though human tracking in the computer vision perspective is well established, exploiting these natural modalities in controlling virtual character's pose and motions in AR is relatively new.

Several works investigated new gestural interactions instead of using the conventional input devices (e.g., keyboard and mouse) to control the animations of virtual characters in AR. [Harviainen et al., 2009] presented an interaction technique that users can interact with a virtual character by gesturing with camera movements. It gives the illusion of the character being aware of the user (or camera) as different camera transformations (e.g., tilting) trigger different actions. [Chen et al., 2017] developed an AR application that allows a user to give orders to a virtual dog by using two interaction modalities— hand gestures and speech (Fig. 2.7). The virtual dog is designed to respond to the hand gestures from the user similar to how people interact with their pet in real world. However, it requires a depth sensing camera for hand tracking.

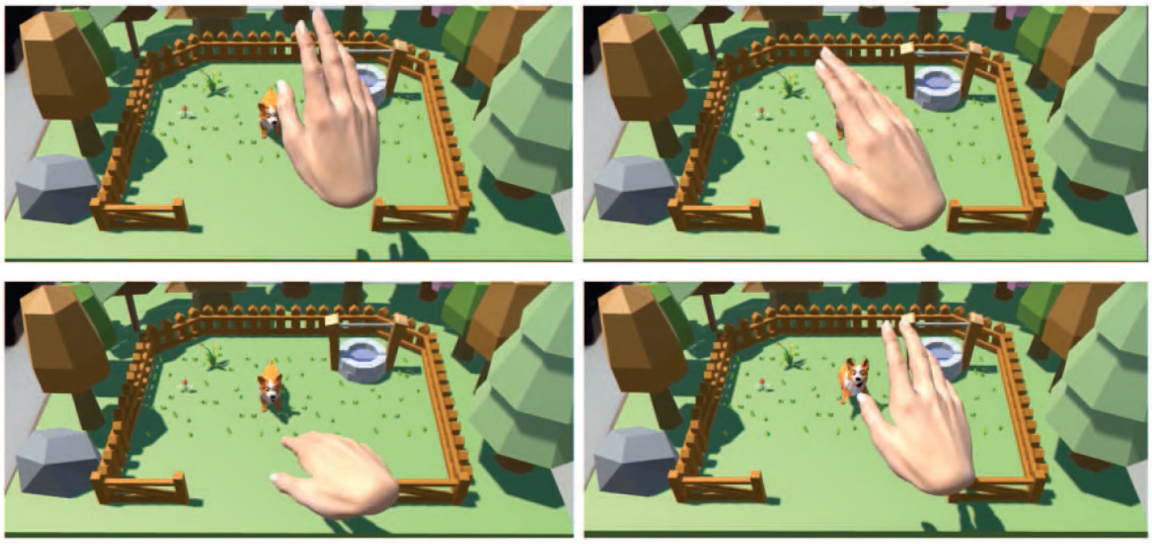

Figure 2.7: *A virtual dog character responds to hand gestures of the user by barking. The image is taken from Chen et al.'s work on multi-model interaction in AR [Chen et al., 2017].*

A few examples consider the case of adding a full digital character interacting with the person in the image [Zünd et al., 2014]. Zund et al. evaluated different aspects of reality mixing techniques. Recently, Apple's Animoji

[Apple, 2018] presents standard emojis as digital avatars that can recognize facial expressions from the front camera of the smart phone and mimic your facial movements. Life-like digital characters that can perceive an user's body poses and movements from common sources like camera photos or video and can react to them seamlessly in AR is a promising and new area of research. To our knowledge, there has not yet been a case of a person's pose automatically estimated and utilized in the context of digital character's mimicking user's behaviors in AR.

2.3.3 Augmenting Human Body

There has been several entertaining applications of people augmentation in the recent years— making user's body a part of the experience by mapping an augmented content onto it. With combining face tracking from RGB images with different 3D masks, Snapchat's lenses [SnapInc, 2018] are popular examples for face augmentation. With the progress in human tracking, AR has an unique potential for augmenting body while using it as a carrier of information.

Several augmentation concepts have been explored around body. [Javornik et al., 2017] presented an AR mirror overlays virtual make-up on people's faces making them look like someone else like a historical character. [Zhang et al., 2017] proposed an virtual try-on application that inserts virtual eyeglasses onto user's face. Their approach can even generate the refraction artifacts caused by lenses based on the user's eyeglasses prescription. Several systems extended the magic mirror concept for training of anatomy giving the illusion of that the user can look into his body utilizing a depth camera device (e.g. a Microsoft Kinect [Shotton et al., 2013]) [Blum et al., 2012, Bauer et al., 2017].

Among them, virtual clothes try-on systems received much attention as they allow user trying on different virtual cloth without the effort of changing them physically. There are two global challenges to tackle for a good cloth augmentation system— a robust pose estimation of the person for the alignment of the virtual clothes, and a scaling method to to fit it better onto the user's body. [Yuan et al., 2013] utilizes Kinect RGB-D imagery that provides body measurements and pose detection for their virtual try-on system. Their scaling method is based on a resizing 3D avatar wearing the virtual cloth according to the user's body size.

The recent progress in pose tracking using image processing techniques has led to new possibilities for virtual cloth augmentation from monocular RGB

Figure 2.8: *The mirror-like AR system by [Bauer et al., 2017] to display the internal anatomy of the current user using Microsoft V2.0 Kinect.*

imagery. The major problem is the lack of realism when actual clothes worn by the user remain visible from side, when the overlaid virtual clothes cannot completely cover them leaving them. [Rogge et al., 2011] presented a cloth augmentation based on a 3D pose estimation system from a single camera view. However, the system relies on a markered suit worn by the user with a specific marker layout to reconstruct 3D joint data.

CHAPTER 3

Physics-Based Character Control Interface for Performance Tracking

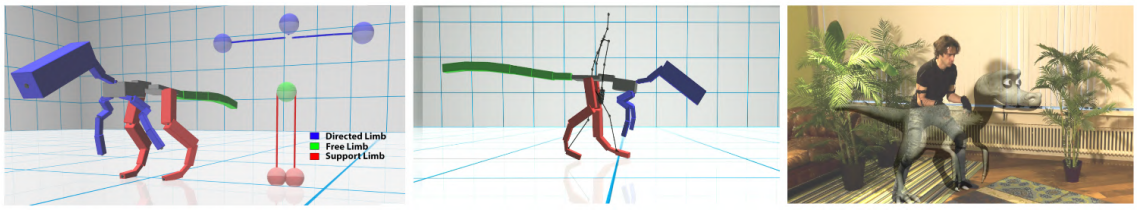

Figure 3.1: *Our method is designed for tracking a human actor with a virtual character in real-time. Traditional methods often leave free limbs such as tails without motion, or provide only repetitive motions. Our method based on optimal control, gets the free limbs involved in the tracking motion.*

While human actors can account for parts of an imaginary creature's motion, for example its feet and hands, human actors cannot simultaneously play parts of the body they do not have, such as a tail. Hence, during live performance capture and retargeting, free limbs such as the tail remain static, or move in a repetitive fashion. In this chapter, we investigate embedding virtual characters into a mechanical simulation, and activating free limbs through optimal control. One of the challenges with simulated control is the intricacy of the controller's design and parameters—leaving this technology out of the hands of casual users and digital artists. We introduce a user-friendly interface that allows casual users to quickly model the character's mechanical system, together with controller parameters required for tracking. We show various examples with a raptor dinosaur, as well as an alien character tracking a live actor in real-time.

3.1 Introduction

Motion capture is heavily used by the visual effects industry to allow a professional actor to craft the performance of a virtual creature. However, because the creature's shape and morphology often differs significantly from the actor's, a motion retargeting step is inevitable. A leading approach to retarget a human actor's motion onto a virtual character is to track shared features such as the feet and the hips using inverse kinematics. The problem with this approach is that it leaves free limbs, such as tails, static and without motion.

To address this issue, we embed the character into a physics-based simulation framework and track the actor's motion using optimal control. As a result, free limbs, such as a virtual creature's tail, are automatically animated to move in unique ways that are consistent with the character's overall performance. While this simulation control concept is a promising approach to synthesize unique movements, its biggest drawback is its complex nature. Physics-based simulation unfortunately requires intricate knowledge of mechanics and control mechanisms, leaving it out of reach for most visual effects artists that lack scientific training. This complexity comes from two main sources. First, the character must be provided an articulated mechanical system with corresponding rigid body and joint properties. And second, a controller general enough to track various human motions in real-time must be set up and tuned so that its parameters are appropriate for the new character. In our work we use model-based inverse dynamics [Abe et al. 2007].

In our research, we make physics-based retargeting more accessible with an intuitive user interface, designed around a general control framework, that allows both quickly designing the character's mechanical system as well as setting the controller parameters for optimal tracking. Our work relies on the core observation that, while limbs may be different, they often share a common function. For example, legs interact with the ground, and free limbs, such as tails, are used for balancing and controlling the character's overall orientation. Hence, we devised a bipedal limb-based abstraction where the user simply drags-and-drops from the abstract limbs to the simulated character's limbs to automatically fill the controller's parameters for tracking.

Our method comes with additional benefits. First of all, we can automatically clean-up contacts from the often noisy captured motion. This is possible thanks to the notion that certain limbs are used for support and interact with the ground. Secondly, our controller penalizes deviations from a nat-

ural pose, which can be used to control the style of the motion, simply by specifying a new default pose. And finally, the tracking can be performed in real-time. We show results of a raptor and alien tracking different motions.

3.2 Technical Overview

Our goal is to have a bipedal creature follow the motion of a human actor in real-time. While characters and the actor may be different morphologically, we observe that they often share a set of common features. In particular, most characters share a global position, global orientation, as well as a set of limbs whose end effector (EE) trajectories match, but at a different scale (see Fig. 3). While tracking only the re-targeted positions conveys the essence of the actor's motion, it leaves the free limbs of the character, such as the tail, without motion. To active the free limbs and provide the character with additional realism, we model the full body dynamics and regulate angular momentum with an additional control objective.

In order to mix different objectives while satisfying the equations of motion, we formulate our tracking as model-based multi-objective control [Abe et al. 2007]:

$$\begin{aligned} \min_{\ddot{q},\tau,f} \quad & \sum w_i \, E_i \\ \text{s.t.} \quad & M(q)\ddot{q} - C(q,\dot{q}) = J_x^T f + [0 \;\; \tau]^T, \end{aligned} \tag{3.1}$$

which exploits the linear relation between joint accelerations $\ddot{q}$ and torques τ when the equations of motion are expressed in generalized coordinates, where the vector q is the root position, the root orientation and the set of joint relative orientations. The matrix $M(q)$ is the generalized mass matrix, $C(q,\dot{q})$ is the vector that combines gravitational forces, coriolis and centrifugal terms. The jacobian transpose J_x^T measures the change in position w.r.t. the generalized degrees of freedom and maps the cartesian ground reaction forces f into generalized forces, with the construction $[0 \;\; \tau]^T$ avoiding root activation.

Setting the weights w_i of the objectives is a cumbersome and time consuming task. To avoid setting all the weights manually, we introduce a *limb* abstraction—summarized in Table 3.1—that automatically sets the weights for each objective. Here we define a *limb* as a connected linear chain of bones attached to the *body frame* which is comprised of the pelvis and upper body (shown in grey in Fig. 3.2).

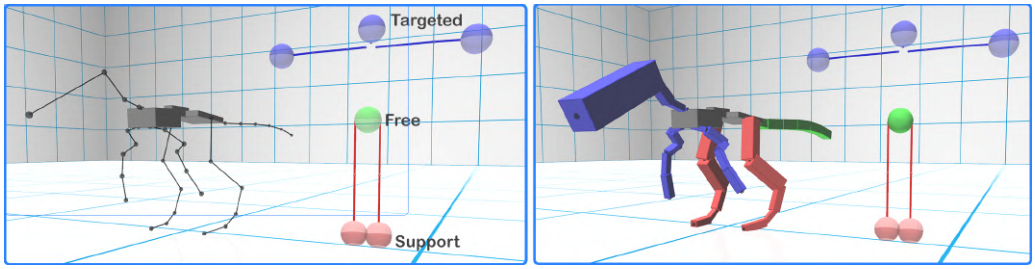

Figure 3.2: *The user interface we use to create an articulated rigid body system and to quickly set control objectives based on a limb abstraction of the bipedal character. The user simply drags and drops a limb type from the abstract humanoid onto the limbs of the character to get it ready for simulated tracking.*

Type	Role
Support	Carry the body through ground reaction forces and ensure contact constraints
Free	Does not track a human part, such as a tail, but participate in angular momentum control.
Targeted	Tracks a human part, but without contacts (e.g. head or hands).

Table 3.1: *Our limb abstraction is comprised of three types.* ***Support*** *limbs, which drive the body to a desired location through ground reaction forces, and alternate between swing and stance states.* ***Free*** *limbs, which do not track a human body part, but help balance and control the character by regulating angular momentum. And finally,* ***Targeted*** *limbs which track a part of the human body such as the hands or head.*

The first step to realize our simulated tracker concept is to model the character's articulated rigid body system and to specify the type of limb, as well as target location on the human actor's skeleton. Hence, in the next section we describe our intuitive user interface to get the character simulation- and tracking-ready.

3.3 Simulated Tracking Interface

The input to our modeler is a bipedal character skeleton, and we provide the necessary widgets to track a *human* skeleton through our limb-based optimal control. We begin by modelling the character's mechanical system, i.e. the linked rigid bodies that approximate the mechanical properties of the character. The second step consists in attributing each of the character's limbs a type from one of our limbs (support, targeted or free).

The user creates rigid bodies by clicking on the bones of the skeleton. We first initialize the rigid body shape with the orientation and size of the skeleton bone, and set the mass and friction to default values. In most cases, the user can use a facilitating function that fills a linear chain of bones with a linear chain of connected rigid bodies, starting from the root, as shown in our accompanying video.

Our limb-based abstraction implicitly encodes which human body part to track. Hence when setting the type of limb by drag-and-dropping a limb type from the limb-based abstraction (shown in Fig. 3.2), the controller is automatically set ready for tracking the human actor. The ordering of the process does not matter, the user can set the type of limb during or after having modeled the articulated rigid body system.

3.4 Online Feature Retargeting

Generally, the features we track are human positions $x_r(t)$ and orientations $\theta_r(t)$, where r denotes *reference* motion. In particular, we track each of the human's end effector positions, together with the root position and orientation.

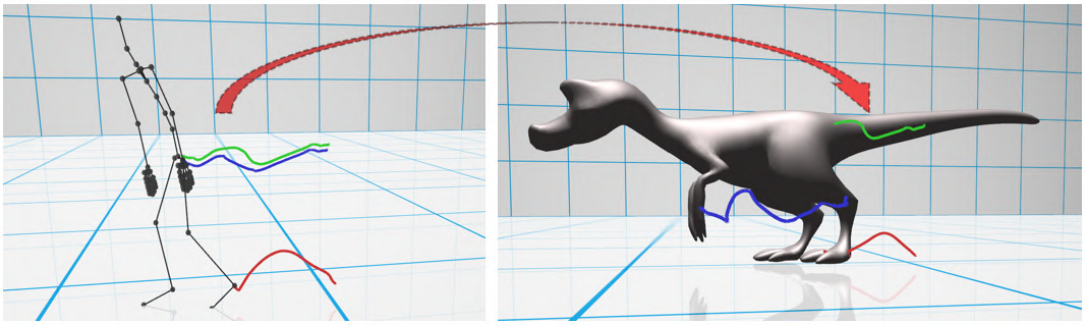

Figure 3.3: *Some of the features in both the character and the human actor's match but at a different scale (e.g. the hand, feet and head, and pelvis positions). We retarget these trajectories to the position and scale of the simulated character for tracking.*

While the orientations can be tracked directly (i.e. $\theta_{des} = \theta_r$ where θ_{des} is the desired orientation), the positions need to be translated and scaled as to be reachable by a character with differently sized limbs. Secondly, because the captured motion can have noisy contact trajectories, we perform online cleaning-up of the end effectors at the extremity of support limbs.

Retargeting Human Position Trajectories. To ensure feasibility of tracked end effector positions, we first compute the differential coordinates of the actor's positions $\Delta x_r(t) = x_r(t + \Delta t) - x_r(t)$, and scale it down based on the proportions $\alpha = l_c \setminus l_r$, where l_* is the distance between a limb's end effector and root position for the rest pose, with c denoting character. This results in:

$$x_{des}(t) = \alpha \, \Delta x_r(t) + x(t), \tag{3.2}$$

where $x(t)$ is the end effector's position. This rescaling is illustrated in Fig. 3.4.

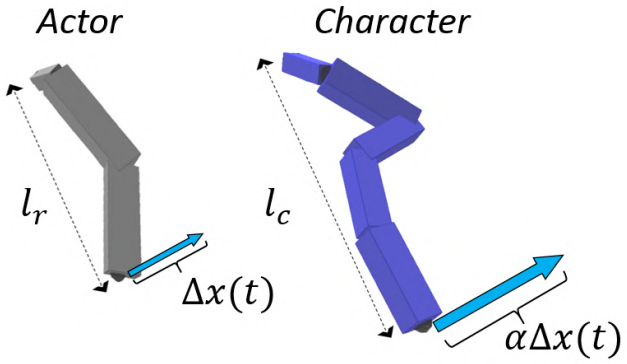

Figure 3.4: *When retargeting the end effector positions to the character, we scale the relative displacements proportionally to each limb length, that we define as the distance between the end effector and the root of the limb.*

Online Contacts Clean-up. We process the end effectors (EEs) at the extremity of *support* limbs, as to clean the contacts in real-time. This is particularly challenging when the capture is being streamed in *real-time*, and we do not have the full trajectory to determine whether a position is supposed to be in contact and remain fixed, or it should be moving.

Our solution to this problem is to keep the actor's end effector fixed when close to the ground (below a contact threshold), and to perform a smooth-in and a smooth-out to transition between the fixed contact position, and the moving position beyond the contact threshold. When the EE position gets below a threshold at time t^0, we project the position onto the ground using its velocity, and define this position as the contact position $x_{contact}$. We then perform a smooth transition between the EE position at the threshold position $x(t^0)$ and the contact position $x_{contact}$ using linear interpolation over a small time window. When the actor's EE position leaves beyond the contact threshold at time t^1, we perform a smooth out transition between the contact position $x_{contact}$ and $x(t^1)$ (see Fig.3.5).

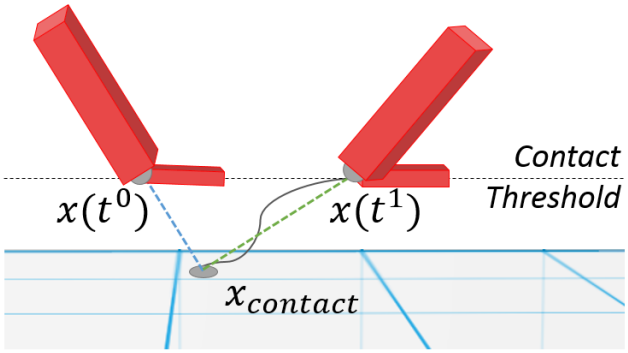

Figure 3.5: *To clean contacts in real-time, we perform smooth-in and smooth-out transitions between the fixed (below the contact threshold) and the moving (above the contact threshold) end effector positions.*

3.5 Control Objectives

We describe our objectives that include both tracking re-targeted positions to be reached by the character, as well as objectives for regulating angular momentum, and controlling style. Each type of limbs contributes to the overall sum of weighted objectives, and we describe at the end of the section how we automatically prioritize the weights based on the type of limb.

Our limb-based controller tracks the global root position, root orientation, the collection of end-effectors at the extremity of the limbs, as well as influences the tracking with additional full body angular momentum and pose regularization. We assemble this sum of weighted objectives (detailed below), and solve problem (3.1) for the optimal torques.

Target Position and Orientation. These objectives are used to track the re-targeted positions x_{des} and orientations θ_{des}, by the character. We compute the desired acceleration for the concerned rigid body based on proportional-derivative (PD) control:

$$\begin{aligned} \ddot{x}_{des} &= k_p(x_{des} - x) + k_d(\dot{x}_r - J_x\dot{q}), \\ \ddot{\theta}_{des} &= k_p(\theta_{des} - \theta) + k_d(\dot{\theta}_r - J_\theta\dot{q}), \end{aligned} \tag{3.3}$$

where x and θ are the rigid position and orientation, k_p the proportional stiffness and k_d, the derivative value, which both remain constant for all motions. Here J_θ denotes the change in orientation for the rigid body, w.r.t. the all the joint orientations. From these desired accelerations, we measure the error to the character's current accelerations:

$$E_p = \|\ddot{x}_{des} - (J_x\ddot{q} + \dot{J}_x\dot{q})\|^2, \tag{3.4}$$

$$E_o = \|\ddot{\theta}_{des} - (J_\theta\ddot{q} + \dot{J}_\theta\dot{q})\|^2. \tag{3.5}$$

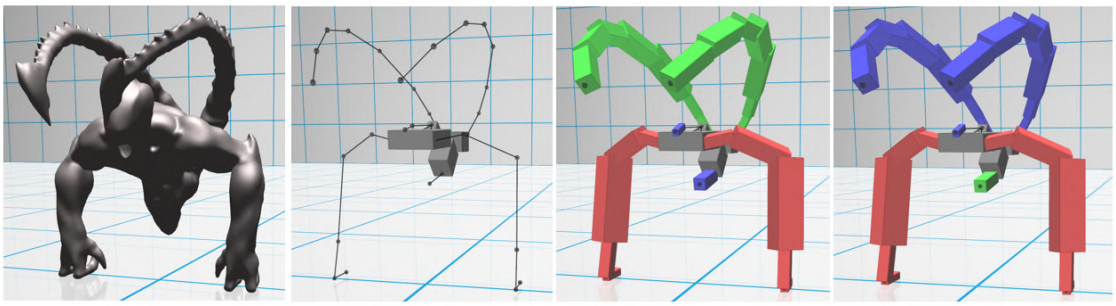

Figure 3.6: *In this figure we see the result of setting two different types of limbs for the alien character. The limb association on the left sets the tails to free limbs, while the association on the right sets the tails to the targeted limbs, which allows the stingers at the tip to perform attacking motions.*

Angular Momentum (AM). The total angular momentum about a point (often the character's center-of-mass) gives a measure of the system's internal rotations. For example, when the actor's upper body bends forward, the total angular momentum changes, and we can minimize this change in AM w.r.t. to all the links (including the tail) to activate the tail's motion.

Hence we minimize the change in total angular momentum w.r.t. to the center of mass:

$$E_{AM} = \|\dot{L}_{AM}\|^2, \tag{3.6}$$

where

$$\dot{L}_{AM} = R(q)M(q)(J_x^T\ddot{q} + \dot{J}_x^T\dot{q}),$$

$$R(q) = [\ [\ r_1(q)\ ]_\times\ \ \ldots\ \ [\ r_m(q)\ ]_\times\ ],$$

where $r_i(q)$ are the position vectors of the i^{th} body links relative to the center-of-mass, $[\ r_i(q)\ ]_\times$ are the skew symmetric coefficients from the cross product, $J_x^T = [\ J_{x_1}^T\ \ \ldots\ \ J_{x_m}^T\ ]$ is the jacobian transpose mapping generalized coordinates to cartesian positions, and $M(q)$ is the mass matrix of the entire articulated rigid body system.

Pose Regularization. To control the style of the animation, as well as to prevent the character from entering unrealistic configurations, we penalize deviations from a *rest* pose. This regularization objective is the sum of errors between joint angle accelerations computed through PD control (similarly to equation (3.5)) with the desired pose defined as the rest pose in equation (3.3), resulting: E_j^{reg} for each joint j.

Total Sum of Objectives. Using equation (3.4) and (3.5), we define a root position and orientation objective E^{root}, as well as the set of end effector objectives E_i^{EE}, $i = 1, ..., m$ with m limbs. With the angular momentum regulation and the pose regularization, the total sum of objectives we minimize is:

$$w^{root}E^{root} + \sum_i w_i^{EE}E_i^{EE} + w^{AM}E^{AM} + \sum_i w_j^{reg}E_j^{reg}, \tag{3.7}$$

which we set as the objectives in problem (3.1). But before solving the problem, we first need to set the weights of each objective.

Limb-based Prioritization. Solving the torques that minimize this sum of weighted objectives (3.7) is challenging in practice as the objectives may be conflicting and need to be prioritized. We observed that certain limbs play a more important role when it comes to performance tracking. For example, the support limbs need to provide clean contact positions and should be weighted higher. Hence, to accommodate the user in setting the weights, we use our limb-based abstraction to automatically set values.

To express the relative priorities of the objectives, we first define a maximum weight value w_{max}, and define each weight based on this maximum value, and on the type of limb. Hence, the global orientation and position being visually important are weighted with the maximum value $w_{root} = w_{max}$, the angular momentum playing a lesser role $w_{AM} = 0.5w_{max}$, and the objectives that depend on the type of limb (the end effectors, as well as the pose regularization) are summarized below:

Limb	w_i^{EE}	w_j^{reg}
Support	w_{max}	0.01 w_{max}
Targeted	0.5 w_{max}	0.02 w_{max}
Free	-	0.03 w_{max}

where $w_{max} = 4000$, and depends on the mass of the system.

3.6 Results

Character setup. We created two characters using our interface described in Section 3.3. The first is a raptor dinosaur and the second an alien that walks on his hands and has two tails. In both cases, an artist created a mesh and a skeletal rig in *Maya* ([Autodesk, 2017]) without constraints regarding its skeleton. Our interface for the character set up allows quickly creating

rigid bodies based off the skeleton, as well as setting different types of limbs for the tracking (see Fig. 3.6 for different types of limb associations). All of the character set ups including the full articulated rigid body systems were created under three minutes each.

Capturing human motion. We captured all the actor's motions using an *Axis Neuron* ([Noitom, 2017]) full body motion capturing system. The system as 13 captors and samples the motion at 120 frames per second. The character's motion is not always accurate and may include body interpenetrations, shakiness in the feet or hands, as well as unstable foot contacts.

Solver. We solve problem (3.1) at each time step $\Delta t = 0.01$, with linearly-constrained quadratic programming. We then integrate the generalized accelerations using the generalized equations of motion. Note that we do not send the torques to a cartesian rigid body simulator, but always perform the simulation in generalized coordinates. Our single threaded implementation runs in real-time on a 4.00 GHz Intel Core i7 machine. We used the same objectives and weights provided by our limb-based abstraction, to track human locomotion (forward and backward), as well as various expressions gestured with hands and upper body such as roaring, being scary and biting (shown in our accompanying video).

Using our limb-based controller across different characters. It is often the case that for each new character, the control objective weights must be adjusted to this new character's proportions. We tested using a similar limb attribution on both the raptor—which has a long and heavy tail with a long upper body—and an alien character—which has two long tails and no legs, but stands on his hands. We found our limb-based control framework to be quite robust in that regard, adapting quite well to changes in character morphology and producing motions that are characteristic to the character's intrinsic shape (shown in our accompanying video).

Controlling style through different rest poses. We experimented with our pose regularization to provide the motion with a different feel. For example, we changed the raptor's pose to have its head slightly tilting forward, which resulted in a sadder look for the motion (shown in our accompanying video).

3.7 Summary and Outlook

Setting the control parameters for a new simulated character is traditionally complex and requires engineering skills. We introduced an intuitive interface for casual users to track human actors with a simulated character in real-time. Our limb-based abstraction simplifies the initial set up to clicking

and dragging on a few nodes. Through the use of optimal control for the tracking, our method automatically activates free limbs such as tails, and provides the character with unique motions that are coherent with the overall action.

One of the main intricacies associated with multi-objective inverse dynamics ([Abe et al., 2007]) are the conflicts between objectives and constraints, which may become unfeasible—causing the simulator to diverge and blow up. One of our remedies was to relax the hard position constraints and replace them with objectives (with a large weight value provided by the support-type of limb). While we greatly simplified the process of tracking human actors with bipedal characters, we left out tracking with quadrupeds and multi-legged creatures, which could addressed by coordinating the stance limbs of the character.

In the next chapter, we focus on the interaction scenarios between users and virtual characters in an AR application via real-world objects, while still ensuring the physical feasibility in the motion model of the characters.

CHAPTER 4

Interacting with Intelligent Characters in AR

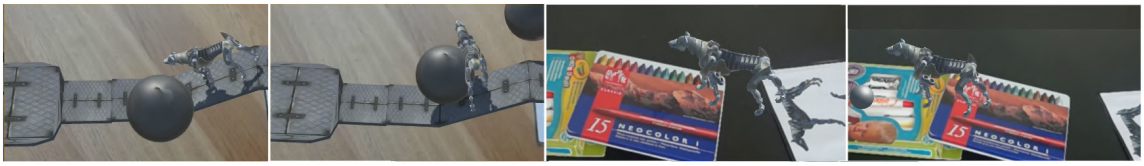

Figure 4.1: *Our intelligent virtual characters can navigate real world environments (right) and react to objects that collide with them (left).*

In this chapter, we explore interacting with virtual characters in AR along real-world environments. Our vision is that virtual characters will be able to understand the real-world environment and interact in an intelligent and realistic manner with it. For example, a character can walk around un-even stairs and slopes, or be pushed away by collisions with real-world objects like a ball. We describe how to automatically animate a new character, and imbue it's motion with adaption to environments and reactions to perturbations from the real world.

4.1 Introduction

Augmenting our environment with intelligent virtual characters that can walk around and interact with our environment is an exciting and promising vision. However, achieving this idea represents several technical challenges.

It remains a challenge to model the motion of a character, have it understand its environment and navigate the world in a natural way.

In this work, we take a first step in the direction of making a character intelligent, and able to interact in AR. We separate the problem into three main components. First is the modelling of the character's motion and its ability to move around. Given a character's skeleton, how should the joints move in order to go from point A to point B, including on un-even terrain. We describe a parametric model of quadruped locomotion, which we use to fill a blend-tree that outputs motions conforming to control directions. The second problem is how to adapt the characters motion to un-even terrains, as well as collisions with objects (such as a ball). We full-fill this by layering on top of the blend-tree, an inverse kinematics solver for terrain adaptation, and a physically simulated rag-doll for character-object collisions. The last problem is the character's ability to understand the environment and navigate it. Online consumer-level scanning of 3D worlds remains inaccurate, and we describe our solution which cleverly combines pre-defined objects with off-the-shelf scanning solutions to provide high-resolution 3D reconstructions of the environment.

Given our intelligent character that can understand the real world and move around, we describe the types of AR interactions we support with real world objects, as well as the experiments we conducted using these interactions.

4.2 Technical Overview

Our animation model provides a virtual quadruped character the ability to navigate real world environments, and react to objects in it in real-time—given only the character's skeleton as input. The motion model is composed of motion clips (walk straight, left, right, backward, etc) that are blended in real-time. Then an inverse kinematics and short-lived ragdoll retargeting method are layered on top to adapt the motion to terrains and perturbations. We start by describing how we generate motion clips from a skeleton.

4.3 Parametric Locomotion Model

We use mechanical simulation, together with characterizations of quadruped motion, to generate locomotion for characters of different shapes and sizes. Internally, our parameterized motion generation system is based on constrained multi-objective optimization. The parameters are what

we call the motion plan: a gait pattern (which foot falls at which time), foot height, center of mass velocity and rotational velocity. We optimize to match these values together with various regularizers ensuring smooth transitions between clips. Some constraints are implicit, or by construction. To support a wide range of characters, we constrain the skeleton to a known simplified template (see Fig. 2) that has only hinge joints and pre-defined masses. The final stage consists in upscaling the motion from the simplified template to the higher-resolution template (see Figure 4.2)

Parameterization

We use a parametric model of quadruped that are composed of articulated chain like structures, in particular, of serially connected and actuated links. The design parameters $\mathbf{s}$ is used to specify the quadruped morphology, which is given by

$$\mathbf{s} = \left(l_1, \ldots, l_g, \mathbf{a}_1, \ldots, \mathbf{a}_n, b_w, b_l\right) , \tag{4.1}$$

where g is the number of links, $l_i \in \mathbb{R}$ is the length of each link, n is the number of actuators, and $\mathbf{a}_i \in \mathbb{R}^3$ is the actuator parameters. For linear actuators, $\mathbf{a}_i$ defines the 3D attachment points, while for rotary actuators, it corresponds to orientation of axis of rotation. Apart from these parameters that represent the kinematic tree morphology of the quadruped, we use two additional parameters b_w and b_l to represent the physical dimensions of the quadruped body (width and length respectively).

Likewise, the motion parameters $\mathbf{m} = (\mathbf{P}_1, \ldots, \mathbf{P}_T)$ are defined by a time-indexed sequence of vectors $\mathbf{P}_i$, where T denotes the time for each motion cycle. $\mathbf{P}_i$ is defined as:

$$\mathbf{P}_i = \left(\mathbf{q}_i, \mathbf{x}_i, \mathbf{e}_i^1, \ldots, \mathbf{e}_i^k, \mathbf{f}_i^1, \ldots, \mathbf{f}_i^k, c_i^1, \ldots, c_i^k, \right) , \tag{4.2}$$

where $\mathbf{q}_i$ defines the pose of the quadruped, i.e., the position, and orientation of the root as well as joint information such as angle values, $\mathbf{x}_i \in \mathbb{R}^3$ is the position of the quadruped's center of mass (COM), and k is the number of end-effectors. For each end-effector j, we use $\mathbf{e}_i^j \in \mathbb{R}^3$ to represent its position and $\mathbf{f}_i^j \in \mathbb{R}^3$ to denote the ground reaction force acting on it. We also use a contact flag c_i^j to indicate whether it should be grounded ($c_i^j = 1$) or not ($c_i^j = 0$).

Motion Optimization

The purpose of motion optimization is to take a quadruped design $\mathbf{s}$ and optmize its motion for user specified task while satisfying certain constraints. We used a cost function $F(\mathbf{s}, \mathbf{m})$ to encode the task specifications. We now describe how $F(\mathbf{s}, \mathbf{m})$ is constructed. To this end, we use a set of objectives that capture users' requirements, and constraints that ensure task feasibility.

Objectives We allow the users to define various high-level goals to be achieved by their quadruped designs such as moving in desired direction with specific speeds, different motion styles, etc. To capture the desired direction and speed of motion, we define the following objectives:

$$\begin{aligned} E_{\text{Travel}} &= \frac{1}{2}||\mathbf{x}_T - \mathbf{x}_1 - \mathbf{d}^D||^2, \\ E_{\text{Turn}} &= \frac{1}{2}||\tau(\mathbf{q}_T) - \tau(\mathbf{q}_1) - \tau^D||^2, \end{aligned} \tag{4.3}$$

where $\mathbf{x_i}$ is the quadruped's COM as defined in eq. 4.2, $\tau(\mathbf{q_i})$ is the turning angle computed from pose $\mathbf{q_i}$, while $\mathbf{d}^D$ and τ^D are desired travel distance and turning angles respectively. E_{Travel} ensures that the quadruped travels a specific distance in desired time, while E_{Turn} can be used to make a quadruped move on arbitrary shaped paths.

Motion style is highly effected by gait or foot-fall patterns that define the order and timings of individual limbs of a quadruped. We internally define various foot-fall patterns for different motion styles such as trotting, pacing, and galloping. When users select a specific motion style, our system automatically loads the necessary foot-fall patterns, thereby allowing novice users to create many expressive quadruped motions. Motion style is also effected by quadruped poses. For expert users, we allow the capability to specify and achieve desired poses, if needed, using the following objectives:

$$\begin{aligned} E_{\text{StyleCOM}} &= \frac{1}{2}\sum_{i}^{T}||\mathbf{x}_i - \mathbf{x}_i^D||^2, \\ E_{\text{StyleEE}} &= \frac{1}{2}\sum_{i}^{T}\sum_{j}^{k}||\mathbf{e}_i^j - \mathbf{e}_i^{jD}||^2, \end{aligned} \tag{4.4}$$

where k is the number of end-effectors, $\mathbf{x}_i^D$ and $\mathbf{e}_i^D$ represent desired quadruped COM, and end-effector positions respectively. Apart from these,

motion smoothness is often desired by the users, which is encoded by the following objective:

$$E_{\text{Smooth}} = \frac{1}{2} \sum_{i=2}^{T-1} ||\mathbf{q}_{i-1} - 2\mathbf{q}_i + \mathbf{q}_{i+1}||^2 . \tag{4.5}$$

Constraints We next define various constraints to ensure that the generated motion is stable.

Kinematic constraints: The first set of constraints ask the position of COM, and end-effectors to match with the pose of the quadruped. For every time step i, and end-effector j:

$$\begin{aligned} \varphi_{COM}(\mathbf{q}_i) - \mathbf{x}_i &= 0 , \\ \varphi_{EE}(\mathbf{q}_i)^j - \mathbf{e}_i^j &= 0 , \quad \forall j, \end{aligned} \tag{4.6}$$

where φ_{COM} and φ_{EE} are forward kinematics functions outputting the position of COM and end-effectors respectively.

We also have a set of constraints that relate the net force and torque to the acceleration and angular acceleration of the quadruped's COM:

$$\begin{aligned} \sum_{j=1}^{k} c_i^j \mathbf{f}_i^j &= M\ddot{\mathbf{x}}_i , \\ \sum_{j=1}^{k} c_i^j (\mathbf{e}_i^j - \mathbf{x}_i^j) \times \mathbf{f}_i^j &= \mathbf{I}\ddot{\mathbf{o}}_i , \end{aligned} \tag{4.7}$$

where M is the total mass of the quadruped, and $\mathbf{I}$ is the moment of inertia tensor. The acceleration $\ddot{\mathbf{x}}_i$ can be evaluated using finite differences: $\ddot{\mathbf{x}}_i = (\mathbf{x}_{i-1} - 2\mathbf{x}_i + \mathbf{x}_{i+1})/h^2$, where h is the time step. Similarly, the angular acceleration $\ddot{\mathbf{o}}_i$ can be expressed as $\ddot{\mathbf{o}}_i = (\mathbf{o}_{i-1} - 2\mathbf{o}_i + \mathbf{o}_{i+1})/h^2$. We note that the orientation of the root $\mathbf{o}_i$ is part of the pose $\mathbf{q_i}$, and it uses axis-angle representation.

Friction constraints: To avoid foot-slipping, we also have the following constraints for each end-effector j:

$$c_i^j(\mathbf{e}_{i-1}^j - \mathbf{e}_i^j) = 0,\ c_i^j(\mathbf{e}_i^j - \mathbf{e}_{i+1}^j) = 0 , \tag{4.8}$$

for all $2 \leq i \leq T-1$, which implies that the end-effectors are only allowed to move when they are not in contact with the ground. Further, to account for different ground surfaces, we enforce the friction cone constraints:

$$f_{i\,\parallel}^j \leq \mu f_{i\,\perp}^j , \tag{4.9}$$

where $f_{i\,\parallel}^{j}$ and $f_{i\,\perp}^{j}$ denote the tangential and normal component of $\mathbf{f}_i^j$ respectively, and μ is the coefficient of friction of the ground surface.

Limb collisions: For physical feasibility, we propose a collision constraint that ensures a safe minimum distance between the limb segments of quadruped over the entire duration of the motion.

$$d(\mathbf{V}_i^{k_1}, \mathbf{V}_i^{k_2}) \leq \delta\,, \tag{4.10}$$

where $\mathbf{V}_i^k$ represents a 3D segment representing the position and orientation of k^{th} limb, $d(\cdot)$ computes the distance between k_1 and k_2 limbs, and δ is the threshold distance beyond which collisions may happen.

Motion periodicity: If the users prefer a periodic motion, we can add an additional constraint that relates the start pose $\mathbf{q}_1$ and the end pose $\mathbf{q}_T$ of the quadruped:

$$\mathbf{J}(\mathbf{q}_T) - \mathbf{J}(\mathbf{q}_1) = 0\,, \tag{4.11}$$

where $\mathbf{J}(\mathbf{q_i})$ extract the orientation of the root and joint parameters from pose $\mathbf{q_i}$.

4.4 High-Resolution Motion

The motion planning algorithm described above mainly cares about the root and the end-effectors of the skeleton, and it does not optimize for the motion style of the limbs. Thus, for motion planning, we choose to use a reduced version of the modeled skeleton which only has two joints for each limb (Figure 4.2(a)). After the motion is generated, we do IK post-processing to match all joints (except intermediate limb joints) and end-effectors between the original high-resolution skeleton and the reduced one (Figure 4.2(c)).

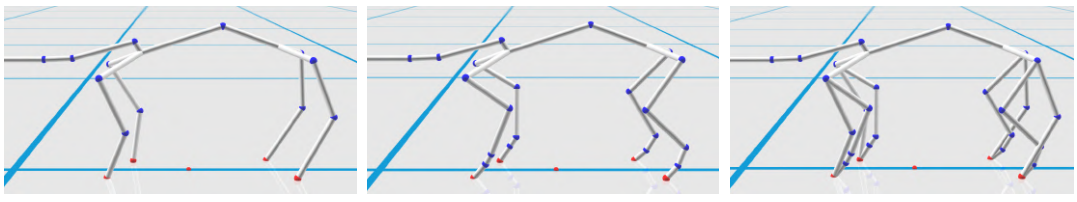

Figure 4.2: *(a) Low-resolution skeleton. (b) High-resolution skeleton. (c) Joint correspondence between low-resolution and high-resolution skeletons.*

IK post-processing We will just use front limb for the discussion. Since motion planning only produces the positions of the shoulder and the end-effector for each limb and we have two additional joints, i.e., wrist and finger, we need to add two parameters to constrain the limb and provide stylizing interface for the user. As illustrated in Figure 4.3(a), L is the distance from the elbow to the end-effector, which can help determine the elbow's position. θ is the angle between the finger and the upright direction, which infers the positions of the finger and the wrist. Additionally, Figure 4.3(b-c) tells us that there are two solutions for the elbow, and similarly for the wrist. Thus, we need two binary parameters choose which way we want the elbow and the wrist to bend. If we inspect the motion of real animals, we will find that their joint angles keep changing during a motion cycle. To mimic such behavior, we use a different set of L and θ when the limb is in full swing, and linearly interpolate these two sets of parameters for other motion phases.

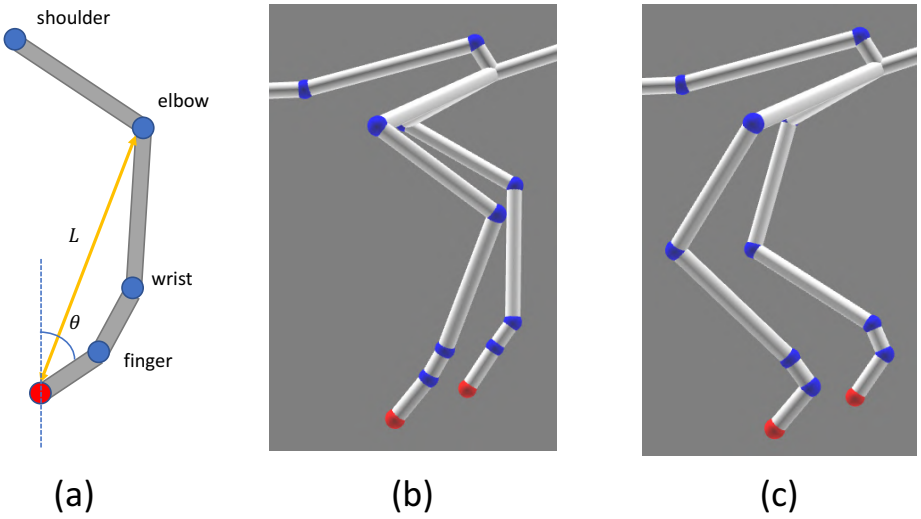

Figure 4.3: *(a) Illustration of the front limb. Two parameters L and θ are used to constrain the joints and stylize the limb motion. (b-c) Since their are two analytical solutions for the elbow position, a binary parameter is used to select one of them.*

4.5 Kinematic Controller and Adaptation

The real-time motions is produced with an animation controller which transitions and blends between motion clips based on two input parameters: the *speed* and *direction* of the character's root. The controller is a state machine holding idle, walk forward, and walk backward states. The walking states

(forward and backward) each blend between 3 motion clips: left, straight and right. The parameters for blending between clips or transitioning between states are detailed in Figure 4.5, and are automatically created from the generated motion clips, which is described in the previous section.

This motion controller performs only motions over flat terrain, and cannot react naturally to pushes and perturbations. To walk over different terrains, we adapt the current frame of the animation using inverse kinematics (IK), based on the terrain height. The ground height is computed by raycasting from the ground foot position, as shown in Figure 4.4.

Finally, to have the character react realistically to physical perturbations, such as being pushed or hit, we added a simulated character (ragdoll) layer on top. For this, we used PuppetMaster ([PupperMaster, 2017]) which is a character's physics tool for automatically generating ragdolls for bipeds. It enables creating active ragdolls that can follow kinematic reference motions. We extended its ragdoll layer for quadruped characters, and used it for simulating reactions.

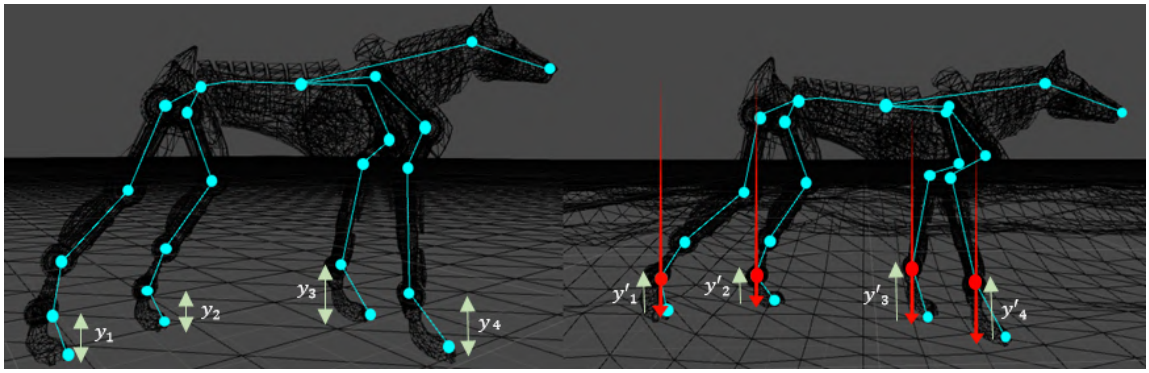

Figure 4.4: *Terrain adaptation is maintained by the estimation of the ground height at the position of the each feet by casting a ray and adding the feet offsets at the current animation frame.*

4.6 3D reconstruction

We describe our approach to understanding the environment for AR purposes. Because current consumer level hardware devices such as the Hololens only offer coarse reconstructions online, we cannot use them for having characters walk over as they appear to be floating in air.

Hence, we developed pre-defined objects that are recognized and localized in space using feature-based technology (Vuforia Engine [Vuforia,

Speed								
	[-1.0, -0.1]	Backward Locomotion	Speed	[-1.0, -0.8]	Walk	Direction	[-1.0, -0.1]	Back. Walk Left
							[-0.1, 0.1]	Back. Walk Straight
							[0.1, 1.0]	Back. Walk Right
				[-0.8, -0.1]	Trot		[-1.0, -0.1]	Back. Trot Left
							[-0.1, 0.1]	Back. Trot Straight
							[0.1, 1.0]	Back. Trot Right
	[-0.1, 0.1]	Standing	Direction	[-1.0, -0.1]	Turn Left			
				[-0.1, 0.1]	Idle			
				[0.1, 1.0]	Turn Right			
	[0.1, 1.0]	Forward Locomotion	Speed	[0.1, 0.8]	Walk	Direction	[-1.0, -0.1]	Forw. Walk Left
							[-0.1, 0.1]	Forw. Walk Straight
							[0.1, 1.0]	Forw. Walk Right
				[0.8, 1.0]	Trot		[-1.0, -0.1]	Forw. Trot Left
							[-0.1, 0.1]	Forw. Trot Straight
							[0.1, 1.0]	Forw. Trot Right

Figure 4.5: *A blending diagram is automatically created from the generated motion clips that controls the motion transition using parametric inputs- speed and direction (of root).*

2017]). For each real world object, we define a corresponding 3D digital geomtric counter-part that matches in shape and size. Then, we scan the real world object from all directions using an RGB camera to obtain a database of image-based features and transformation pairs. At runtime, Vuforia searches for matching features and returns the id of the object, together with its transformation that we apply to the 3D object in the scene.

We encountered a few issues recognizing objects with Vuforia. One problem is when the objects are transparent, or have plain textures. In this case, the lack of features causes the recognition to fail. Similarly to object recognition, objects with shiny and reflective properties do not give successful image recognition and tracking. Hence for some objects, we add a rich texture on top to make them distinguishable, as shown in our figures bellow.

4.7 Interactions

We take our animated character together with its ability to navigate the real world environment, and design AR interactions in 3D. In particular, we provide ways for the user to specify where the character should go, ways to have 3D virtual and real objects collide with the character, as well as ways to configure different terrains.

Specifying trajectories via touch on the screen. The quadruped character can be directed through any arbitrary paths in the physical environment,

over different terrains. The paths are generated by *projecting* onto the environment, the user's fingertip when drawing on the touchscreen, as shown in Figure 4.6.

Walking over different real-world slopes. The virtual character's behavior depends on the purpose of the interaction. Therefore we label objects either as terrain or non-terrain in order for the motion model to behave in the correct manner. We label the terrain automatically by defining objects as terrain if their height is bellow a certain threshold, that corresponds to the maximum height the character can climb.

Different slopes and platforms can be formed with different arrangements of the objects as shown in Figure 4.7. While the character's motion model will only employ the inverse kinematics for adapting to objects labeled as terrain, it should react differently for the other objects, as described next.

Pushing characters with real-world objects. Non-terrain objects can be used for interactions like colliding with or pushing the quadruped, as shown in Figure 4.8. For animating the reactions, a ragdoll simulation (uncontrolled passive dynamics) is activated for a short period of time, letting the character react to the perturbation. After the short period of time, the state of the simulated (ragdoll) character is blended back into the animation state over another small window of time. Completely switching to a ragdoll simulation causes to the character to fall. Hence, above a certain force threshold, we do not blend back to the animation and simply let the character flow.

Interacting with virtual objects. We also experimented the interactions between the character and virtual objects. We designed a simple platformer game (which is shown in our accompanying video), where the user can use various *props* to carry the virtual character from the beginning to the end of the platform puzzle, while trying to prevent him from falling. The character only moves forward, and its moving direction can only be changed if it hits a *wall prop*. When the character meets an *elevator platform* which goes up and down, the user needs to use a *fan prop* (shown in Figure 4.10) to stop the character such that it can wait. For an increased challenge, we added an *enemy cannon* which shoots 3D balls at the character, possibly causing it to fall from the platform, as shown in Figure 4.9.

4.8 Summary and Outlook

We proposed a first step in the direction of bringing intelligent characters to life in augmented reality. The system automatically models the charac-

Figure 4.6: *Path drawing with touch is used to direct the character in the physical environment.*

Figure 4.7: *Different arrangements of predefined physical objects creates different slopes for the character to walk on. For the details of the 3D object reconstruction, we kindly refer to Section 4.6.*

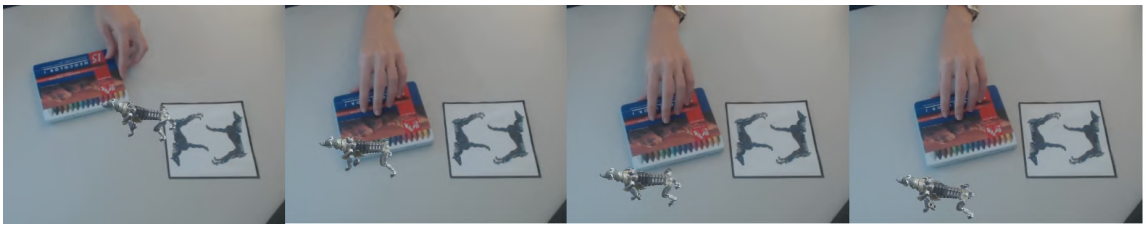

Figure 4.8: *The character reacts to the pushes by real objects.*

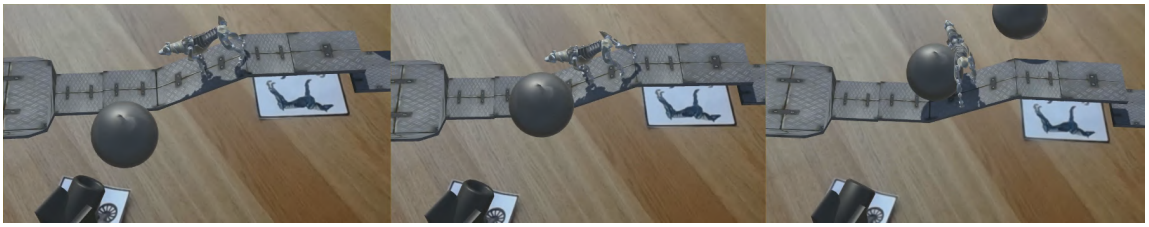

Figure 4.9: *During the platformer game, the character can be hit by a virtual cannon ball and fall down.*

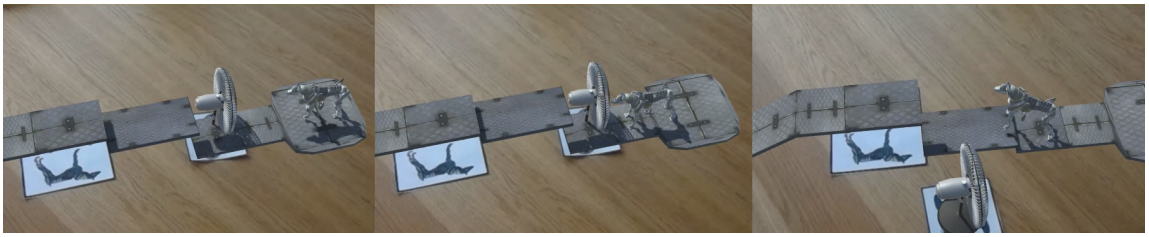

Figure 4.10: *A virtual fan can be used to stop the character which creates virtual forces.*

ter's motions respecting its skeletal structure and joint movements, as well as giving an ability to adapt it to changes in terrain and interact with both physical and virtual objects in real time. The AR characters can understand understand their environments with a set of predefined real-world objects and navigate them.

Even if the character's responses are in real-time, we can only interact slowly with objects, as the tracking is remains at low frequency. In addition, we used pre-defined 3D objects instead of scanning the world. We believe that both of these issues will improve with the evolution of hardware.

In the next chapter, we further investigate AR characters that can capture user's poses to mimic them in AR selfies. Furthermore, the chapter continues by introducing an AR costume concept that allows augmenting watertight clothes onto the estimated user's poses.

CHAPTER 5

AR Selfies

This chapter presents two frameworks that incorporate digital characters and costumes into selfie settings in AR. First, we introduce *AR Poser*, which allows virtual characters intelligently to mimic the user's pose in an AR selfie. The core of our solution in this section relies on identifying the person in the selfie scene utilising a computer vision technique and then estimating the closest 3D pose with a projection on a 3D pose subspace. Second, we present another framework, *AR Costumes*, that overlays a "watertight" costume on a digital image of a person. The method proposed in Section 5.2 is a combination of techniques: estimating the proportions of a body, approximating a 3D pose as in *AR Poser* for the costume with an additional refinement considering the estimated proportions, and using masking and inpainting to remove the visible skin and clothes behind the costume.

5.1 AR Poser

Figure 5.1: *Examples of poses automatically recovered and augmented with a digital character using our method.*

We introduce *AR Poser*: a framework for posing *with, or as* a digital character. In this chapter, we describe our first contribution to *AR Poser*: a technique for digital characters to recognize and automatically reproduce the same pose as a person in a picture (using only RGB information from a mobile device). 3D human pose estimation from RGB is an under-constrained and ambiguous problem that remains today an active field of study. Instead of addressing the general case of human pose estimation, we propose a solution that can be tailored to a specific scenario—such as entertainment poses for AR selfies. At the heart of our solution is a set of predefined poses (selfie poses) utilized to reduce ambiguities. In a nutshell, our method consists of two reliable steps: we first perform 2D pose estimation, and then perform a projection onto the 3D subspace to find the closest matching 3D pose. With our method, we are able to automatically create augmented reality selfies for a variety of different poses.

5.1.1 Introduction

Digital augmentation of the real world opens new dimensions for ideation, communication and entertainment. For example, facial tracking combined with different mask overlays recently resulted in highly entertaining and popular mobile applications. In the future, we can imagine combining human shape estimation with digital character augmentation to unlock various entertaining selfie scenarios. Hence, we introduce *AR Poser*: a framework for posing with or as a digital character. We describe our first contribution to *AR Poser*: a technique for digital characters to automatically reproduce the same pose as a person in a picture.

To automatically imitate the person's pose with a 3D digital character, we need to estimate the 3D pose of the person from a single monocular image (RGB). 3D human pose estimation from RGB is an under-constrained and ambiguous problem that remains an active field of study. Instead of addressing the general case of human pose estimation, we propose a solution that can be tailored to a specific scenario—such as poses AR selfies. At the heart of our solution is a set of predefined poses (selfie poses) utilized to reduce ambiguities associated with depth when estimating 3D poses. In a nutshell, our approach consists of breaking down the problem into two more reliable steps: first a 2D pose estimation, and then a projection onto our 3D subspace to find the closest matching 3D pose. With our method, we were able to automatically create augmented reality selfies for a variety of different poses.

5.1.2 2D Pose Estimation

We have recently seen rapid progress in 2D pose estimation from monocular images using deep learning "in the wild". There are now packaged solutions that offer robust solutions for multiple subjects and occluded parts. In this work, we use the pre-trained network *OpenPose* [Cao et al., 2017], which was trained on the COCO [Lin et al., 2014] and MPII [Andriluka et al., 2014] datasets.

The pre-trained network takes as input the RGB image, and returns a list of joint positions y_i together with a confidence value c_i. For example, a partially visible body will result in a low confidence value for the joints outside the image. The neural network was trained over a large data-set of hand-annotated images—each with the skeleton of the people in the image.

The 2D skeleton has a set of joint names, that we associate to the 3D joints of our character, as shown in Fig.5.2. This map is defined manually in our case, but could be done automatically given corresponding T poses for example. With the 2D joint positions associated to 3D joints, we can proceed to the step of computing the best matching 3D pose.

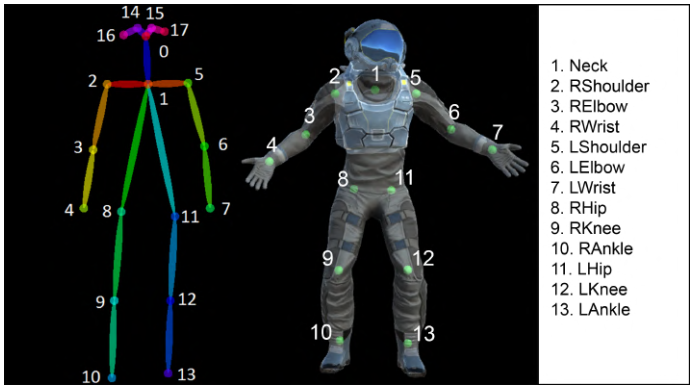

Figure 5.2: *The 2D skeleton on the left is obtained from OpenPose. It has 18 joints. On the right is the 3D character that we used in our experiment. A common subset of joints need to be mapped for the 3D pose matching process.*

5.1.3 3D Pose Projection

The way we project the 2D skeleton onto the 3D pose space is via local optimization. We assume a small set of 3D poses, in our case entertaining selfies, as shown in the results section. The 3D poses constrain the solution space to only plausible articulations of the character's body. Since possible selfie

poses are symmetrical, we handle symmetries by mirroring the 3D poses in the dataset along the y-axis.

For each pose in the data base, we optimize for the rigid transformation that will bring the 3D pose, closest to the 2D projected skeleton, in terms for joint positions. The global transformation of a 3D pose is parameterized with 3 degrees of freedom, as we constrain the translation of the character along the y-axis, as shown in Fig. 2.

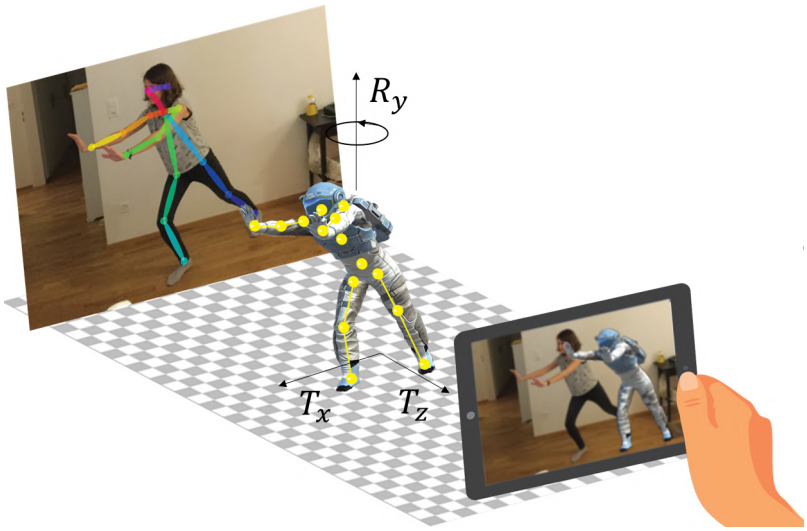

Figure 5.3: *From 2D pose estimation to 3D pose subspace and finding optimal character pose.*

Formally, for each pose $X^k = \{x_i\}^k$ defined as a set of joint positions x_i, we optimize for a reduced rigid transformation M composed of a rotation around the y axis R_y, and translations along the x and z axises T_x, T_z—resulting in $M = T_y\, T_x\, R_y$ and shown in Fig.5.3—that minimizes the similarity cost between the 3D projected joint positions $P\ M\, x_i$ and the 2D joint positions y_i, where P is the view and projection transformation of the camera (see next section for how we estimate the mobile camera's parameters). Finally, we go through all the optimal transformations and poses pairs $< X^k, M >$, and pick the one that has the smallest cost value, resulting in the following optimization problem:

$$X^*, M^* = \underset{<X^k, M>}{\operatorname{argmin}} \min_{M} \sum_i ||y_i - P\ M\, x_i||^2. \tag{5.1}$$

We solve the internal optimization for the transformation M using gradient-based optimization along numerical derivatives. This requires initializing

the 3D pose front facing the camera as to ensure convergence towards a sensible solution.

We described how to match the 3D pose to the 2D skeleton, but this depends on 3D camera parameters for the projection. Next we describe how we estimate these for the mobile device given a know *a priori* marker in the scene.

5.1.4 Augmentation and Mobile Setup

To incorporate a 3D character into the real world picture using a mobile device, we need to estimate the camera parameters: a view and perspective matrix. The perspective matrix is given by the device, while we use marker-base technology (*Vuforia*) [Vuforia, 2017] to recognize and track the camera's transformations. We print a real world marker that is about the size of a person, and process the texture for visual features. When the mobile device takes a picture, it contains the marker, which is then used to estimate the orientation and position of the camera.

The 3D character pose used in the optimization (section 5.1.3), is initialized to roughly fit inside the bounding box of the marker. The optimization adjusts the character's depth translation to match the same size as the person's 2D skeleton. If the character is to be smaller, (e.g. a dwarf) we wait until the end of the optimization, to scale the final 3D pose back to its original size.

Finally, the neural network we use in section 5.1.2 (*OpenPose*) to estimate the 2D pose of a person is sizable and runs optimally on a graphics card. Deploying such a system on a mobile device represents a significant integration effort, and will suffer from a loss of performance due to the difference in hardware. Our solution is to place the 2D pose estimation "in the cloud" and send messages between the mobile device taking pictures, and the 2D pose estimation running on a server.

5.1.5 Results and Discussion

We designed a creative concept around Space Exploration that resulted in 12 relevant poses. We started with a set of 10 poses, and invited people to experience the application. The subjects performed poses we did not have, which we then crafted and included in the dataset, removing the ones that were not relevant. Two such iterations ended up thee 12 relevant poses shown below.

The pictures were taken from a mobile device, sent to a server for the 2D skeleton estimation (running OpenPose), and then the 3D pose matching

was performed on the mobile device. The whole process took about 2 seconds.

The sum of joint positions that we minimize is successful at matching the shape of the character, but does not always succeed at finding a perceptually similar size for the character. It can be seen in our results that sometimes the character is larger than others. We could fix this with a final pass that adjusts the size based on the shoulder and feet proportions, which seem to be visually important.

Naturally, poses not present in the database fail to be discovered. This is a limitation by design. Also, at the moment we only tackled and demonstrated pose similarity for body joints—excluding the face and the hands. In consequence, similar body poses that have different hand gestures will fail to be discriminated. We think this could be tackled with a 2 step matching where first the full body is matched, then the different hand poses are considered.

5.1.6 Summary and Outlook

We proposed a practical approach to produce augmented reality selfies with digital characters. It relies on a set of predefined poses that are automatically selected and adjusted based on a 2D pose estimate of the character. The process works with mobile devices like smartphones and tablets.

While a few minor improvements are required to match people of different sizes, it unlocks possibilities to investigate new interactions. Inspired by this, in the next section of this chapter, we have the digital character's costume be worn by the person in the picture— AR costumes. While pose estimation is sufficient for many applications, it falls short when fitting cloth onto a person, which requires a good estimate of shape. Therefore, we perform an additional optimization on the full degrees of freedom of the 3D skeleton to match the 2D skeleton better with local refinements. These refinements are relatively small such that we do not lose the notion of what constitutes a viable and common human pose that is obtained with the first constrained optimization step.

5.2 AR Costumes

Figure 5.4: *Naively matching a 3D costume pose to person's pose (middle column), results in several parts of the person visible. In this work, we solve these problems with shape estimation of the costume, together with inpainting of the person's body.*

We describe a method to automatically augment a watertight digital costume onto a person's body from a monocular RGB image. When overlaying a digital costume onto a body using pose matching, several parts of the person's cloth or skin remain visible due to differences in shape and proportions. In this thesis, we present a practical solution to these artifacts which requires minimal costume parameterization work, and a straightforward inpainting approach. To our knowledge, our approach is the first to deliver plausible watertight costumes from RGB imagery only, and is compatible with mobile devices. We believe this can serve as a useful baseline for future improvements and comparisons.

5.2.1 Introduction

Imagine taking a selfie and magically wearing your favorite character or hero's suit. While we did see digital cloth added onto people in the past, it was often with a depth camera such as a *Kinect*, which is not always reliable in outdoor conditions, and is not as widespread as monocular cameras on mobile devices. In this thesis, we carry out this concept from a single RGB image, in a manner compatible with mobile devices.

People come in different shapes and sizes, and estimating the best costume to fit their given pose and proportions is a challenge. While recent work supports estimating shapes, it might not be the desired solution to fully cover the body: artistic direction might require the shape to remain slender or muscular, for example. Hence, we approach this problem with a costume parametrization based on different skeleton *proportions* (variations in limb lengths such as legs and spine), and combine this with *inpainting* to remove the remaining visible parts, such as cloth or skin from the person behind, as shown in Fig. 5.4.

Our solution is practical and requires minimal parametrization work. Given a 3D costume, we manually create different versions associated to a 3D skeleton of different proportions. Together with a data set of poses, we optimize for the best matching 3D costume to the person's 2D skeleton (estimated from the RGB image using a 2D pose tracker). Once the best matching shape (pose and proportions) is found, we need to remove the remaining visible regions of the person. To solve this, we estimate the person's body mask we want to remove (e.g. the body, but without the head or hands), and proceed with inpainting the masked region. To inpaint, we capture the background image without the person, and then compute a projection transform—or *homography*—from four feature points in the source image to the target image, followed by Poisson image editing to match the surrounding color and lighting.

The entire process runs in about ten seconds on a first generation *Surface Book* convertible, without an optimized solution. We show successful results on various poses and proportions—including complex poses where limbs are crossing. Together with successful results, we show failure cases in supplementary material which will be useful for comparisons and evaluating future improvements.

5.2.2 3D Costume Shape

To summarize, our approach breaks down the problem of costume fitting from a single RGB image into two main parts: a shape estimation described in this Section, followed by *mask* estimation and *inpainting* for the *remaining* visible parts, described in Section 5.2.3.

Hence given an RGB image containing a single person, our goal is to find a costume shape which best fits the pose *and* proportions of the person. Our 3D shape estimation follows a 2D inference plus 3D matching type of approach as in *AR Poser* (Section 5.1.3), but extended with estimating proportions followed by refinement.

We first estimate the 2D skeleton with joint positions y_i of the person, as described in Section 5.1.2. We then parameterize the 3D costume mesh with different 3D skeleton poses k *and* proportions c (variations in limb lengths), resulting in $p = c \times k$ shapes in our data set.

From the 2D skeleton, we optimize for the closest 3D pose k^*, then search for the optimal proportions c^* using a heuristic that favors shoulders and hips for closer perceptual similarity. The final pose is close to the 2D skeleton, but could still be refined. Hence we perform a final full space refinement optimization to match more exactly the limb directions and joint positions of the 2D skeleton.

Proportions Estimate

Given our closest pose k^*, we seek to choose the closest matching proportions c^* to better fit the 2D skeleton. In our experiments, we found that comparing the sum of all joint positions, such as in the previous section, did not lead to perceptually similar proportions, or resulted in confusing the optimization (5.1) into the wrong pose. We found that focusing on the shoulders and hips, which are visually more prominent, yielded better results perceptually, and more robust pose and proportions pairs.

Figure 5.5: *Our three shapes with variations in limb lenghts. The arms and legs are longer on the left, and shorter on the right. A better estimate of the proportions helps the refinement of the pose converge to a better solution.*

Our selection criteria is based on two features $f = [f_{s/w}, f_{h/w}]$ measuring the shoulder-to-waist ratio $f_{s/w}$, and the shoulder width versus average upper

body height ratio $f_{s/h}$, defined as:

$$f_{s/w} = \frac{\mid S_L - S_R \mid}{\mid H_L - H_R \mid}$$
$$f_{s/h} = \frac{2 \cdot \mid S_L - S_R \mid}{\mid S_L - H_L \mid + \mid S_R - H_R \mid},$$

where S_L and S_R are the left and right shoulders, and H_L, H_R the hips of the skeleton in 3D.

We select the 3D shape c which has the closest feature vector to the *target* 2D skeleton features f_t when inverse projected onto a plane centered on the 3D costume. Specifically, we pick the shape c that minimizes the weighted sum at the L2 norm:

$$c^* = \underset{c}{\operatorname{argmin}} \| w \ [f_t - f_c]^T \|^2.$$

where $w = [w_0 \ w_1]$ are both equal to 1 in our implementation.

While there is a variety of different proportions in people, we found that three main modes ($\mid c \mid = 3$) was sufficient to span most of our subjects, and represented a satisfactory compromise between speed, set-up complexity and quality. Additionally, the refinement step discussed next can contribute to fixing slight proportion mismatches in 2D, as we optimize in 3D allowing the limbs to visual shorten when projected onto the screen.

Global-local Refinement

At this point, we have a 3D shape (pose k^* and proportions c^*) which is close to the person's shape, but is still different in the exact bone orientation and joint position, as shown on the left in Fig. 5.6. To remove these differences, we perform an additional refinement step with respect to the full degrees of freedom of the 3D character: the joint orientations $Q = q_i$ and the root position x_0 of the character. Because bone positions may not match exactly, we weight down this objective in optimization (Eq. 5.1), and add an additional objective function which seeks to match the bone *directions*, resulting in the following optimization:

$$Q^*, x_0^* = \min_{Q, x_0} w_p \ E_p + w_{dir} \ E_{dir},$$

$$E_{dir} = \sum_i \| (y_i - y_{p(i)}) - (PM^* x_i - PM^* x_{p(i)}) \|^2,$$

where $p(i)$ is the parent of i. We solve this problem in a global / local fashion where we optimize for the global position while keeping the orientation

Figure 5.6: *We optimize globally for the root position to adjust the scale, and alternate with local optimization of the joint angles in a back-and-forth manner, to finally converge to a well matching pose.*

fixed, and solve for the individual joint orientations while keeping the position fixed. Both of these steps are performed using local gradient descent along numerical derivatives.

We now have a costume that matches closely in pose and proportions, but when overlayed over the person, leaves cloth and skin from the person visible, as shown on the right in Fig. 5.6. We remove these in the next section by estimating a 2D mask and inpainting.

5.2.3 Inpainting and Composition

The costume shape overlayed on the person at this point still has cloth or skin visible, as shown in Fig. 5.7. To remove these artifacts, we estimate the 2D mask of the person's body and head, and then inpaint the body area using background information. When rendering the 3D costume, we can obtain an odd look when the lighting and shadows differ from the real world, and when the costume appears plastic or unnatural. Hence we estimate the lighting direction by sampling the picture, and filter the final render to produce a more natural look for the costume.

Masking

To compute the 2D mask, we use *Grabcut* [Rother et al., 2004], which requires an initial labelling of the *foreground, probably foreground* and *background* pix-

Figure 5.7: *We first estimate the person's mask using the estimated 2D skeleton and Grabcut. Then we define a Homography transformation from target image coordinates to source (background) coordinates in order to color the masked pixels. Finally we apply Poisson image editing to fix the remaining color discrepencies.*

els. We use the estimated 2D skeleton, and set *foreground* pixels that are within a distance r of a few pixels of the joint positions, and within $2r$ of the skeleton bones—defined as lines between joints. For the head specifically, we set a slightly larger ellipse to indicate the facial pixels to obtain a more precise boundary. Pixels within a larger radius are marked as *probably foreground*, while the rest remains assumed *background*. We run the algorithm for 5 iterations which yields reasonable results in most cases. With complex backgrounds, it sometimes misclassifies pixels. To circumvent this problem we simply inflate the mask to be inpainted. The final result can be seen in Fig.5.7.

Inpainting

Our goal is to color the masked pixels with plausible underlying scene values. Hence we capture the environment (with a video) and seek to find the pixel colors that best match the structure of the captured background, while resembling the colorization of the target picture.

Our solution consists in computing a *projective* transformation (a.k.a Homography) from the closest matching background with respect to camera parameters, to the new target image, using 4 corresponding points in the images: in our case, the 4 corners of the *AR Poser* poster. When capturing the back-

ground, we record the camera position x and orientation q. Given a new camera position and orientation $x^{'}$ and $q^{'}$ (at runtime), we search our dataset for the nearest background image. Note that for speed we used a KD-tree.

Given the nearest background image, we want a warping function that maps coordinates x, y in the target image, to coordinates x'', y'' in the source (background) image. Hence we track the four positions of the corners in the source image $S_{1,2,3,4}$ and target image $T_{1,2,3,4}$, and define a projection transformation by assembling:

$$W_S = S_{(1-3)}^{-1} \cdot S_4, \tag{5.2}$$

where $S_{(1-3)}$ is the 3×3 matrix concatenating the first 3 vectors in the source image as homogenous coordinates $x, y, 1$. The matrix resulting from multiplying $S_{(1-3)}$ by the vector W_S, is the transform that maps the source square to canonical coordinates. Hence we can transform from the *target* square to the canonical space and then to the *source* via:

$$M = W_T \cdot T_{(1-3)} \left(W_S \cdot S_{(1-3)} \right)^{-1}, \tag{5.3}$$

which for a given pixel coordinate x, y, we obtain the intermediate coordinates: $\begin{bmatrix} x' & y' & z' \end{bmatrix}^T = M \cdot \begin{bmatrix} x & y & 1 \end{bmatrix}^T$, which require a final dehomogenization:

$$x^{primeprime} = \frac{x'}{z'} \qquad y'' = \frac{y'}{z'}.$$

Sampling pixels from this function yields similar color and structure, but does not ensure boundary smoothness and color consistency, as can be seen in Fig.5.7. Hence we further optimize the pixel values to blend with the target image by minimizing the target color gradient while preserving the source color gradient—a method known as Poisson image editing [Pérez et al., 2003]. We solve this using an existing packaged solution in OpenCV [Bradski, 2000].

Composition

We render the 3D costume using rasterization rendering in *Unity*. Simply overlaying the inpainted picture with the rendered costume might hide parts of the head of the target person. To avoid this, we attach a simple, transparent 3D object approximating a generic human head to the neck-bone of the character's rig, which acts as a depth mask during the render pass and occludes the relevant parts of the costume.

Figure 5.8: *Our optimzation may result in large deformations when misclassifying the person's proportions (left). Another issue is we do not track the 2D feet orientation at the moment, and cannot reproduce this pharao pose at the moment (middle). Similarly limgs crossing are not prevented for the moment in our optimization. Estimating the mask area of the face in the legs crossing pose, without the hands, is challenging. Finally, poses that expose the inner area of the mesh are not taken into account at the moment, and methods to adress this are discussed in our results section.*

	Neutral	Hero	Wave / greet	Arms behind	Victory - 2 hands in the air	Arms crossed in front	legs crossed - arms behind head crossed	Wow - lean forward	Pharao
%	100	100	100	85.7	71.4	57.1	42.9	14.3	14.3

Figure 5.9: *Average likeability score for the 9 poses, performed by 7 subjects. Some of the poses are well handled accross people, others yield mitigated likebility, while others are not well handled by our current method.*

As for the rendering, we use a single directional light to approximate the scene lighting. When the light direction is different from the one in the picture, the rendering looks odd. Hence we need to find an appropriate lighting direction, which we do by sampling the face of the person in the image. Additionally, Phong shading tends to yield plastic-looking materials, which differs from the overall feel of the picture. Our quick fix is to add noise to the costume's rendering.

To estimate the lighting direction, we use the 2D face landmarks from the 2D pose estimation to sample different points in the source picture. We then sample their HSV values by averaging the neighboring pixels. In particular, we use points around the cheeks and forehead since they tend to have less unwanted noise in comparison to glasses or hair. Thanks to the face joints we can also align a 3D mesh of a face to match the joint positions.

By sampling the same set of points over the 3D mesh, we can read the normal direction of that vertex, and by a weighted average of the normals multiplied by the value of the pixels, we can infer a rough approximation of the direc-

tion of the light source. We use the resulting vector to set the new rotation of a directional light that illuminates the virtual costume and creates shadows in the ground. A more accurate approach is described in citeface2light, but an implementation in this context is left for future work.

5.2.4 Results and Discussion

The pictures are taken from a surface book, sent to a server for the 2D skeleton estimation (running a pose tracker on the GPU). The skeleton is sent back to the device which processes the skeleton and image to match the shape and perform inpainting. The whole process takes about 10 seconds, from which two thirds is used by computing the segmentation with *Grabcut* [Rother et al., 2004] and the inpainting using Poisson image editing [Pérez et al., 2003]. Our code was not optimized for speed.

Qualitative Study

Our data set holds 12 poses and we performed a qualitative user study of 9 poses, similar to the most recurrent ones people do. We had 7 different person perform the 9 poses. We then showed the results different people and asked to rank the likability of the results as binary value: 1 for like, and 0 for do not like. The average of the evaluations shown in table 5.9 resulted in 4 of the poses with a success rate above 80%, with 3 having 100%, 3 having mitigated likability, and 3 being systematically unconvincing (bellow 20%).

The mitigated likeability we believe are due to two main artifacts. We sometimes obtain large deformations when our proportions classification fails, which causes the subsequent refinement stage to over-compensate resulting in la large deformations, as shown in on the left in Fig.5.8. The second artifact is the collar, which sometimes overlap with the mouth, which changes the nature of the costume. We think this could be addressed by fitting a 3D head model to the person's face, and avoiding interpenetration of the costume with the head.

The systematically unconvincing results we believe are due to poses are method cannot handle properly at the moment. Our optimization does not hold 2D feet markers, and so we fixe the orientation the feet—preventing from matching the sideways pose of the *pharao*, as shown in Fig.5.8. Similarly, we do not avoid intersections between limbs, which can cause the legs crossing pose to fail in most cases. This could be improved with a subspace optimization of the costume shape, or similarly by restricting the bones to

anatomically plausible angles. Finally, the "wow" pose which leans forward exposes the inner area of the mesh, which our method does not handle automatically. We would need a 3D model of the person's head to cull the back side of the mesh from being visible after rendering.

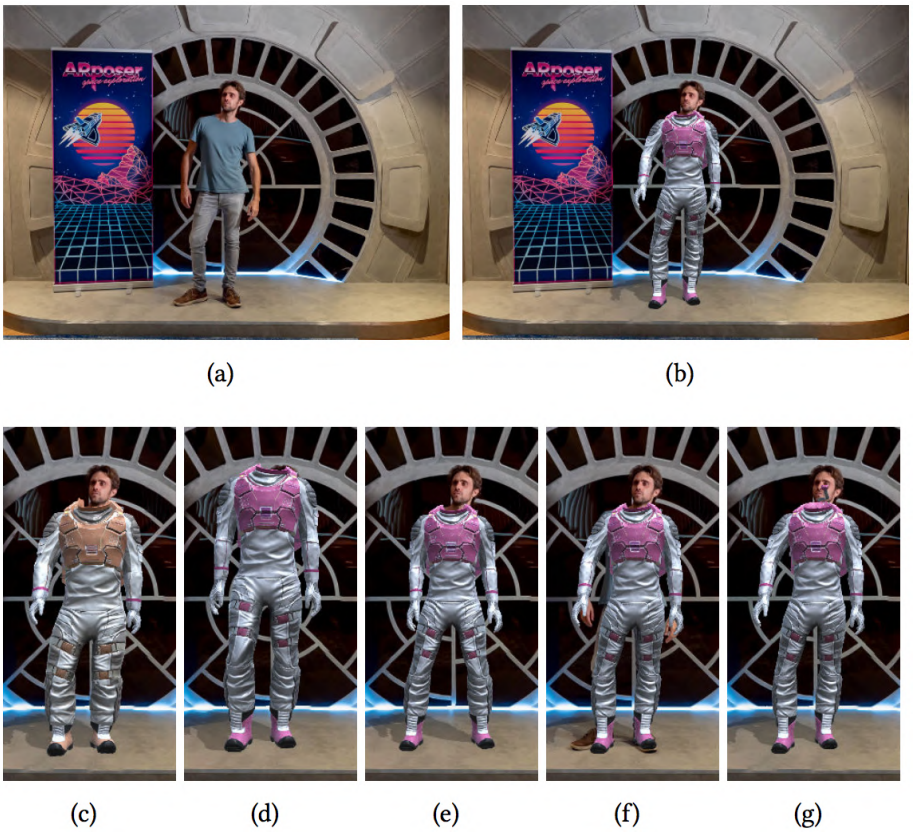

Figure 5.10: *To judge the importance of each step in our method, we performed an ablation study by computing the results with the full pipeline, each time leaving out on step. Image (b) shows the result with all steps applied to source image (a). The bottom row shows the partial results with: (c) no proportion estimate, (d) no size refinement, (e) no bone direction refinement. (f) no pinpointing and (g) no approximate head masking.*

Ablation Study

To evaluate the effect of the different steps as well as their necessity, we generated the results by iteratively leaving one out. Fig. 5.10 shows the results. Refinement and inpainting have the most dramatic effect and leaving them out results in unconvincing compositions. A lesser impactful step is our proportions estimation which selects amongst a few discrete costumes (3 shapes). We observed that an ill-matched character can be adjusted by the

refinement process. However, it can be observed that the visual quality of the results is generally better when a costume with a similar body type is selected. The same holds for the method used to color correct the inpainted image material. In many cases, histogram matching is enough to get a convincing result, but the Poisson energy minimization compensates for much more differences in color and can make the difference in more extreme cases.

5.2.5 Summary and Outlook

We created a system to accurately overlay a person in a monocular RGB image with a watertight 3D costume matching in proportions and pose. It furthermore improves the quality of the result by removing visible artifacts of the source picture by inpainting the relevant areas, but keeping specific body parts of the target person, resulting in a realistic image composition.

Trying with new characters, different than the astronauts, requires fine tuning parameters in our optimization (sections 5.1.3 and 5.2.2). We also observed that we could obtain better results for certain people and poses by tweaking the parameters. Note that we kept them fixed for our evaluation, but it could be interesting to classify the optimization parameters based on the person's picture.

The next chapter concludes the work presented in this thesis and discusses future directions and limitations of the each different frameworks described in previous chapters.


ARposer
ARposer
ARposer
ARposer
ARposer
ARposer
ARposer
ARposer


ARnaser
ARnaser
ARnaser
ARnaser
ARnaser
ARnaser
ARnaser
ARnaser


ARposer
pure exploration


ARnoser


ARposer

CHAPTER 6

Conclusion

The AR characters' biggest draw is their interactive and engaging nature. However, not only overlaying virtual characters into the the real world, but also fully exploiting the physical world with natural and direct input modalities in the context of human-character interaction is yet an unexplored concept. Our vision is that AR characters will increasingly become more interactive in such a way that they can respond to voice, gestures and even touch, and develop a better understanding of the space they are in with the objects in their physical surrounding such that they immediately react to changes in the environment in a natural way.

This thesis has presented steps toward creating motion models for AR characters possessing some of the aforementioned characteristics through several example application scenarios. Our focus in these scenarios was the directness of interaction, the consistency of the characters' behaviors with their physical surroundings, thus making human-character interaction more natural. To conclude, in this chapter, we summarize the main contributions, discuss the limitations and providing future directions.

6.1 Summary

Physics-Based Character Control Interface. Physics-based approaches are better suited for AR and VR applications as they can generate motions online that are responsive to the environment changes and the user input, and physically accurate since optimization directly relies on the equations of mo-

tion. However, these methods require knowledge of multi-body dynamics, numerical integration and control theory.

In Chapter 3, we presented an intuitive interface for physics-based character control where a user can drive a virtual characters motions in real-time. The approach generalizes the process of setting up the control parameters for tracking, which is a tedious task and requires re-tuning each time when the character is changed, by providing a limb-based abstraction. Our limb-based controller operates on the principle that different limbs play certain role in a specific scenario like performance tracking which results in different distributions of effort. Some is responsible for carrying the body to the target trajectories provided by the human performer, other only contributes to the regulation for more stable and balanced motions. The advantage of this proposed automatic weighting scheme is that it combines the parameter tuning with character mapping, and makes it available to casual users by integrating this information into a limb-based abstraction.

Thanks to the proposed abstraction, our system can also supports a variety of bipedal characters with different morphologies, as well as different limb settings for the same character. We tested the generalization of our system with two different biped creature: Raptor and alien, which have very different morphologies. Finally, another contribution of our proposed tracking system is its activation of the free limbs of the character (e.g., tails). Normally, the tail of a character in simulation would automatically have passive motions as it being dragged by the body. On the other hand, animals in nature voluntarily use their tails increase balance and stability in their movements. For example, an interesting study showed that losing horizontal sways when geckos drop their tails limits the rotation of their pelvis, which ultimately decrease step length in their locomotion [Jagnandan and Higham, 2017]. In our system, the tail is actuated to regulate angular momentum and eventually balance in the overall motion of the character.

Interacting with Intelligent AR Characters. With computer vision's capability of identifying real objects and mapping virtual overlays atop them, AR has the power to turn our physical environments into digital gaming platforms. Unfortunately, having virtual characters understanding changes of real objects manipulated by users and naturally responding them is currently underexplored. In Chapter 4, we took the first steps to support virtual AR characters to understand and react to different physical and virtual objects, as well as coping with different terrains arranged with real objects, and steering to target locations (through target paths), which are interactively defined by user.

We use a parametric model of quadruped motion based on trajectory opti-

mization with multiple objectives. The model enable automatically applying locomotion to a variety of characters based on several motion parameters defined by user. These parameters of the model are the gait pattern, the speed of the motion, the turning rate. In the optimization, these parameters turn into objectives: the end effector trajectories, together with different regularizations (e.g., smoothness). The advantage of our system is that the user does not have to deal with setting such as character's mechanical system for the optimization because adapting the predefined, simplified character template to a different character is straight-forward. In addition, it still allows users to change the style to the generate motions with the aforementioned parameters.

AR Selfies. In Chapter 5, we described two frameworks that explore new social interaction scenarios between users and digital characters— *AR Selfies*. First, we introduced *AR Poser*, which allows taking a selfie with a human-like avatar, or a virtual 3D recreation of a popular comic book character imitating your pose. It is based on a 3D pose estimation technique from single RGB camera image taken with a mobile device. The advantage of our technique is that it is compatible with mobile devices and reliable in outdoor conditions by combining robust 2D pose estimation with 3D lifting in a small 3D poses space.

Further, we described *AR Costumes*, which allows projecting a watertight costume onto a person by approximating the dimension and matching the costume to the body joints of the person. Finally, to remove cloth or skin visible behind the costume, we use masking the body while still leaving hands visible, and in-painting to fill in with the background image. After in-painting, additionally, we apply Poisson image editing technique to match the light and color difference between current and background image. To our knowledge, our solution is the first that can overlay watertight costumes from RGB imagery only, and is practical with minimum parametrization.

6.2 Limitations and Future Directions

Physics-Based Character Control Interface.

The downside of combining multiple objectives is that there may be interference between different objectives. We handle this by implicitly prioritizing objectives with our limb-based abstraction that sets higher weights to the limbs that meets high-priority objectives. However, we noticed that, as a design mistake, our angular momentum objective cancels rotations about the COM, which causing a conflict with the root orientation objective that

is responsible to achieve desired direction. Consequently, we are unable to generate motions including sharp turns. One potential solution is to allow the angular momentum changes producing the target body rotation.

While our limb-based abstraction supports walking on hands and having multiple tails, it is designed specifically for bipedal characters. In the future, we would like to extend our system to the case of quadrupeds by introducing a coordinated stance-limb planner which coordinates end-effector trajectories between the support limbs of the character.

Our system can alter the character's motion style by simply changing a single pose regularization. It doesn't depend on large collections of data. However, the single pose regularization could easily be replaced with a data-driven pose function learned from a collection of example poses (using for example [Grochow et al., 2004]).

Interacting with Intelligent AR Characters.

The pose estimation of virtual objects with marker tracking is in general not stable in AR. This error in position tracking has an effect on the animated responses of the virtual characters. For example, it can be seen that the character's feet drift little on each step while walking. This is the biggest issue in marker based AR issues and will be overcome with the evolution to more stable tracking technology.

Detecting physical toys and embedding their virtual counterparts would boost the immersiveness of this work. Therefore, a future direction would be investigating chameleon technology [Chameleon, 2018] with in-painting, allowing real-world characters to "come-to-life", by replacing their background. We can further increase the connection of the characters to the real world by adding shadows.

A future direction for enhancing our character's intelligence would involve other sense as well, such as responding to the user's voice and specific hand gestures, similar to what [Chen et al., 2017] proposed. Additionally, with hand tracking, it would be possible to steer characters by drawing a virtual path at the user's fingertip on the real table. Furthermore, a robust 3D hand reconstruction of the user's hand on top of it, can allow user to change pose of the characters by animating their virtual body-parts. Moreover, with a haptic feedback suit, which applies forces and vibrations to the user's hand, can enable the interaction using real touch.

AR Selfies.

Both frameworks presented in Chapter 5 is tied to the physical location marker, *AR Poser* poster, to estimate the orientation and position of the cam-

era. Not only we use it to import the character at a location, but also in the optimization to pose avatars at a location. A future option would be to utilize markerless AR technology that allows augmenting without the use of image targets, like the Wikitude SDK [SDK, 2019] and their instant tracking, or simultaneous localization and mapping (SLAM) technology. It tracks the user's surrounding to localize the device and to detect plane surfaces (e.g. floor, walls).

Another limitation is the number of poses and shapes that are stored in dataset. Our methods remain to be tested with children, who have more variation in limb lengths, compared to adults. We plan to increase the number of poses to make them more stable and robust. We think that accommodating characters that have significantly different or exaggerated limb proportions, such as a cartoon character with tiny legs, would require changing the head position, and thus revisiting our design.

Our in-painting requires scanning the area before hand, which requires starting over when the environment changes. Also, we in-paint using a projection transformation derived from a known marker in the scene (the AR Poser poster), and in the event it changes location, we must rescan the environment once again.

Finally, when masking the target person using *Grabcut* [Rother et al., 2004], we don't always get a segmentation that is precise enough. This results in body and background parts that are still visible after the in-painting, or it may hide parts of the head. Additionally, the approximated head model used to hide parts of the 3D costume may not be accurate enough (see figure 5.8). This could be improved by using a more detailed, dynamically adjustable model, to estimate the shape of the person.

References

Abdul-Massih, M., Yoo, I., and Benes, B. (2017). Motion style retargeting to characters with different morphologies. *Computer Graphics Forum*, 36(6):86–99.

Abe, Y., da Silva, M., and Popović, J. (2007). Multiobjective control with frictional contacts. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 249–258.

Anabuki, M., Kakuta, H., Yamamoto, H., and Tamura, H. (2000). Welbo: An embodied conversational agent living in mixed reality space. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, pages 10–11.

Andriluka, M., Pishchulin, L., Gehler, P. V., and Schiele, B. (2014). 2d human pose estimation: New benchmark and state of the art analysis. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3686–3693.

Aoki, T., Matsushita, T., Iio, Y., Mitake, H., Toyama, T., Hasegawa, S., Ayukawa, R., Ichikawa, H., Sato, M., Kuriyama, T., Asano, K., Kawase, T., and Matumura, I. (2005). Kobito: virtual brownies. In *SIGGRAPH Emerging Technologies*.

Apple (2018). Animoji. `https://support.apple.com/en-us/HT208190/`.

Autodesk (2017). Maya. computer animation and modeling software. `http://www.autodesk.com/products/maya/overview/`.

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355–385.

Barakonyi, I., Weilguny, M., Psik, T., and Schmalstieg, D. (2005). Monkeybridge: Autonomous agents in augmented reality games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '05, pages 172–175.

Bauer, A., Neog, D. R., Dicko, A.-H., Pai, D. K., Faure, F., Palombi, O., and Troccaz, J. (2017). Anatomical augmented reality with 3d commodity tracking and image-space alignment. *Computers and Graphics*, 69:140–153.

Billinghurst, M., Kato, H., and Poupyrev, I. (2001). Magicbook: Transitioning between reality and virtuality. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 25–26.

Blum, T., Kleeberger, V., Bichlmeier, C., and Navab, N. (2012). Mirracle: An augmented reality magic mirror system for anatomy education. In *Proceedings of the 2012 IEEE Virtual Reality*, VR '12, pages 115–116.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Buys, K., Cagniart, C., Baksheev, A., Laet, T. D., Schutter, J. D., and Pantofaru, C. (2014). An adaptable system for rgb-d based human body detection and pose estimation. *Journal of Visual Communication and Image Representation*, 25(1):39 – 52.

Cao, C., Hou, Q., and Zhou, K. (2014). Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph.*, 33(4):43:1–43:10.

Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310.

Casas, L., Ciccone, L., Çimen, G., Wiedemann, P., Fauconneau, M., Sumner, R. W., and Mitchell, K. (2018a). Multi-reality games: An experience across the entire reality-virtuality continuum. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '18, pages 18:1–18:4.

Casas, L., Ciccone, L., Cimen, G., Wiedemann, P., Fauconneau, M., Sumner, R. W., and Mitchell, K. (2018b). Multi-reality games: an experience across the entire reality-virtuality continuum. In *ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, VRCAI '18.

Caudell, T. P. and Mizell, D. W. (1992). Augmented reality: an application of heads-up display technology to manual manufacturing processes. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, volume 2, pages 659–669.

Chameleon (2018). Chameleon power. `http://www.chameleonpower.com/`.

Chatzopoulos, D., Bermejo, C., Huang, Z., and Hui, P. (2017). Mobile augmented reality survey: From where we are to where we go. *IEEE Access*, 5:6917–6950.

Chen, C. and Ramanan, D. (2016). 3d human pose estimation = 2d pose estimation + matching. *CoRR*, abs/1612.06524.

Chen, Z., Li, J., Hua, Y., Shen, R., and Basu, A. (2017). Multimodal interaction in augmented reality. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 206–209.

Choi, K.-J. and Ko, H.-S. (1999). On-line motion retargetting. In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, PG '99, pages 32–42.

Cimen, G., Maurhofer, C., Sumner, B., and Guay, M. (2018a). AR Poser: Automatically Augmenting Mobile Pictures with Digital Avatars Imitating Poses. In *12th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing 2018*.

Cimen, G., Yuan, Y., Sumner, R., Coros, S., and Guay, M. (2018b). Interacting with intelligent characters in ar. *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)*, (2017/2).

Clark, A., Dünser, A., and Grasset, R. (2011). An interactive augmented reality coloring book. In *SIGGRAPH Asia 2011 Emerging Technologies*, SA '11, pages 25:1–25:1.

Coros, S., Beaudoin, P., and van de Panne, M. (2010). Generalized biped walking control. *ACM Trans. Graph.*, 29(4):130:1–130:9.

Da Silva, M., Abe, Y., and Popović, J. (2008). Simulation of human motion data using short-horizon model-predictive control. *Computer Graphics Forum*, 27(2):371–380.

de Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 131:1–131:10.

Dontcheva, M., Yngve, G., and Popovi'c, Z. (2003). Layered acting for character animation. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 409–416.

Feng, L., Yang, X., and Xiao, S. (2017). Magictoon: A 2d-to-3d creative cartoon modeling system with mobile ar. *2017 IEEE Virtual Reality (VR)*, pages 195–204.

Gleicher, M. (1998). Retargetting motion to new characters. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 33–42.

Grasset, R., Dünser, A., and Billinghurst, M. (2008). Edutainment with a mixed reality book: A visually augmented illustrative childrens' book. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 292–295.

Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531.

Guillemot, C. and Le Meur, O. (2014). Image inpainting : Overview and recent advances. *Signal Processing Magazine, IEEE*, 31:127–144.

Harviainen, T., Korkalo, O., and Woodward, C. (2009). Camera-based interactions for augmented reality. In *Proceedings of the International Conference on Advances in Computer Enterntainment Technology*, ACE '09, pages 307–310.

Hecker, C., Raabe, B., Enslow, R. W., DeWeese, J., Maynard, J., and van Prooijen, K. (2008). Real-time motion retargeting to highly varied user-created morphologies. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 27:1–27:11.

Henrysson, A., Billinghurst, M., and Ollila, M. (2006). Ar tennis. In *ACM SIGGRAPH 2006 Emerging Technologies*, SIGGRAPH '06.

Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 71–78.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2013). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36.

Jagnandan, K. and Higham, T. (2017). Lateral movements of a massive tail influence gecko locomotion: An integrative study comparing tail restriction and autotomy. *Scientific Reports*, 7:10865.

Javornik, A., Rogers, Y., Gander, D., and Moutinho, A. M. (2017). Magicface: Stepping into character through an augmented reality mirror. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 4838–4849.

Kang, C. and Woo, W. (2011). Armate: An interactive ar character responding to real objects. In *Proceedings of the 6th International Conference on E-learning and Games, Edutainment Technologies*, Edutainment'11, pages 12–19.

Kato, H. and Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmentedreality conferencing system. pages 85–94.

Kavafoglu, Z., Kavafoglu, E., Cimen, G., Capin, T., and Gurcay, H. (2018). Style-based biped walking control. *Vis. Comput.*, 34(3):359–375.

Kenwright, B. (2010). Character inverted pendulum pogo-sticks, pole-vaulting, and dynamic stepping.

Kulpa, R., Multon, F., and Arnaldi, B. (2005). Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum*, 24(3):343–351.

Lee, Y., Kim, S., and Lee, J. (2010). Data-driven biped control. *ACM Trans. Graph.*, 29(4):129:1–129:8.

Levine, S. and Popović, J. (2012). Physically plausible simulation for character animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '12, pages 221–230.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. In *ECCV*.

Macchietto, A., Zordan, V., and Shelton, C. R. (2009). Momentum control for balance. *ACM Trans. Graph.*, 28(3):80:1–80:8.

Maes, P., Darrell, T., Blumberg, B., and Pentland, A. (1995). The alive system: full-body interaction with autonomous agents. In *Proceedings Computer Animation'95*, pages 11–18.

Magnenat, S., Ngo, D. T., Zünd, F., Ryffel, M., Noris, G., Roethlin, G., Marra, A., Nitti, M., Fua, P., Gross, M. H., and Sumner, R. W. (2015). Live texturing of augmented reality characters from colored drawings. *IEEE Transactions on Visualization and Computer Graphics*, 21:1201–1210.

Martinez, J., Hossain, R., Romero, J., and Little, J. J. (2017). A simple yet effective baseline for 3d human pose estimation. *CoRR*, abs/1705.03098.

Maurhofer, C., Cimen, G., Ryffel, M., Sumner, R. W., and Guay, M. (2018). Ar costumes: Automatically augmenting watertight costumes from a single rgb image. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '18, pages 4:1–4:8.

McIntosh, K., Mars, J., Krahe, J., McCann, J., Rivera, A., Marsico, J., Israr, A., Lawson, S., and Mahler, M. (2017). Magic bench: A multi-user & multi-sensory ar/mr platform. In *ACM SIGGRAPH 2017 VR Village*, SIGGRAPH '17, pages 11:1–11:2.

Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., and Theobalt, C. (2017). Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 Fifth International Conference on 3D Vision (3DV)*. IEEE.

Milgram, P. and Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12:1321–1329.

Mohring, M., Lessig, C., and Bimber, O. (2004). Video see-through ar on consumer cell-phones. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '04, pages 252–253.

Monzani, J.-S., Baerlocher, P., Boulic, R., and Thalmann, D. (2000). Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum*, 19:11–19.

Mordatch, I., de Lasa, M., and Hertzmann, A. (2010). Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics*, 29(4):1.

Noitom (2017). Perception neuron. `https://neuronmocap.com/content/axis-neuron-software/`.

Pavlakos, G., Zhou, X., Derpanis, K. G., and Daniilidis, K. (2016). Coarse-to-fine volumetric prediction for single-image 3d human pose. *CoRR*, abs/1611.07828.

Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318.

PupperMaster (2017). Rootmotion. `http://root-motion.com/`.

Rhodin, H., Tompkin, J., In Kim, K., Varanasi, K., Seidel, H.-P., and Theobalt, C. (2014). Interactive motion mapping for real-time character control. *Comput. Graph. Forum*, 33(2):273–282.

Rogge, L., Neumann, T., Wacker, M., and Magnor, M. (2011). Monocular pose reconstruction for an augmented reality clothing system. In *Proc. Vision, Modeling and Visualization (VMV)*, pages 339–346. Eurographics.

Rother, C., Kolmogorov, V., and Blake, A. (2004). "grabcut": Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 309–314.

Ruiperez, J. and Ruiperez, G. (2018). Vivo lifelike reactive characters for vr. In *ACM SIGGRAPH 2018 Virtual, Augmented, and Mixed Reality*, SIGGRAPH '18, pages 29:1–29:1.

Scherrer, C., Pilet, J., Fua, P., and Lepetit, V. (2008). The haunted book. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 163–164.

Schraffenberger, H. and van der Heide, E. (2016). Multimodal augmented reality: The norm rather than the exception. In *Proceedings of the 2016 Workshop on Multimodal Virtual and Augmented Reality*, MVAR '16, pages 1:1–1:6.

SDK, W. (2019). Wikitude gmbhr. `https://www.wikitude.com/`.

Seol, Y., O'Sullivan, C., and Lee, J. (2013). Creature features: Online motion puppetry for non-human characters. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '13, pages 213–221.

Shin, H. J., Lee, J., Shin, S. Y., and Gleicher, M. (2001). Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2013). *Real-Time Human Pose Recognition in Parts from Single Depth Images*, pages 119–135.

SnapInc (2018). Snapchat face lenses. `https://support.snapchat.com/en-US/a/face-world-lenses/`.

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 757–764.

Tak, S. and Ko, H.-S. (2005). A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24:98–117.

Taylor, S., Kim, T., Yue, Y., Mahler, M., Krahe, J., Rodriguez, A. G., Hodgins, J., and Matthews, I. (2017). A deep learning approach for generalized speech animation. *ACM Trans. Graph.*, 36(4):93:1–93:11.

Thomas, B., Close, B., Donoghue, J., Squires, J., Bondi, P. D., Morris, M., and Piekarski, W. (2000). Arquake: an outdoor/indoor augmented reality first person application. In *Digest of Papers. Fourth International Symposium on Wearable Computers*, pages 139–146.

Tomè, D., Russell, C., and Agapito, L. (2017). Lifting from the deep: Convolutional 3d pose estimation from a single image. *CoRR*, abs/1701.00295.

Toshev, A. and Szegedy, C. (2013). Deeppose: Human pose estimation via deep neural networks. *CoRR*, abs/1312.4659.

Tsai, Y.-Y., Lin, W.-C., Cheng, k. b., Lee, J., and Lee, T.-Y. (2010). Real-time physics-based 3d biped character animation using an inverted pendulum model. *Visualization and Computer Graphics, IEEE Transactions on*, 16:325–337.

Vögele, A., Hermann, M., Krüger, B., and Klein, R. (2012). Interactive steering of mesh animations. In *Proceedings of the 11th ACM SIGGRAPH / Eurographics Conference on Computer Animation*, EUROSCA'12, pages 53–58.

Vuforia (2017). Qualcomm, http://www.qualcomm.com/vuforia. `http://www.qualcomm.com/Vuforia/`.

Wagner, D., Billinghurst, M., and Schmalstieg, D. (2006). How real should virtual characters be? In *Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '06.

Wagner, D., Pintaric, T., and Schmalstieg, D. (2004). The invisible train: A collaborative handheld augmented reality demonstrator. In *ACM SIGGRAPH 2004 Emerging Technologies*, SIGGRAPH '04, pages 12–.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., and Schmalstieg, D. (2008). Pose tracking from natural features on mobile phones. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134.

Wang, C., Wang, Y., Lin, Z., Yuille, A. L., and Gao, W. (2014). Robust estimation of 3d human poses from a single image. *CoRR*, abs/1406.2282.

Wei, S., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. *CoRR*, abs/1602.00134.

Yamane, K., Ariki, Y., and Hodgins, J. (2010). Animating non-humanoid characters with human motion data. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 169–178.

Yasin, H., Iqbal, U., Krüger, B., Weber, A., and Gall, J. (2015). 3d pose estimation from a single monocular image. *CoRR*, abs/1509.06720.

Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3).

Yuan, M., Khan, I. R., Farbiz, F., Yao, S., Niswar, A., and Foo, M.-H. (2013). A mixed reality virtual clothes try-on system. *IEEE Transactions on Multimedia*, 15:1958–1968.

Zhang, Q., Guo, Y., Laffont, P.-Y., Martin, T., and Gross, M. (2017). A virtual try-on system for prescription eyeglasses. *IEEE Computer Graphics and Applications*, 37:84–93.

Zimmermann, C., Welschehold, T., Dornhege, C., Burgard, W., and Brox, T. (2018). 3d human pose estimation in RGBD images for robotic task learning. *CoRR*.

Zordan, V. B. and Van Der Horst, N. C. (2003). Mapping optical motion capture data to skeletal motion using a physical model. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 245–250.

Zünd, F., Lancelle, M., Ryffel, M., Sumner, R. W., Mitchell, K., and Gross, M. (2014). Influence of animated reality mixing techniques on user experience. In *Proceedings of the Seventh International Conference on Motion in Games*, MIG '14, pages 125–132.

Çimen, G., Guay, M., Coros, S., and W Sumner, R. (2017). An intuitive interface for human performance tracking with simulated characters. In *11th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing 2017*.