Bachelor Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Retargeting Infant Movements to Baby Humanoid Robots

Ondřej Fiala

Supervisor: doc. Mgr. Matěj Hoffmann, Ph.D.
Supervisor–specialist: Valentin Marcel, Ph.D.
Study program: Cybernetics and Robotics
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Fiala Ondřej**	Personal ID number:	**499272**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Retargeting Infant Movements to Baby Humanoid Robots

Bachelor’s thesis title in Czech:

Převedení pohybů dítěte na humanoidní roboty

Guidelines:

Motion retargeting deals with the transformation of movements from one body to another. In this case, the inputs are 3D coordinates of keypoints on the bodies of infants and the outputs are movements of a simulated baby robot. This will make it possible to replay the movements of real babies in a humanoid robot and complement the kinematic data with sensory feedback - proprioception (joint angles), vision (from the eyes of the robot), and eventually touch (using artificial skin on the robot body). Such a tool will be key to understanding the mechanisms of early infant development.
Instructions:
1) Familiarize yourself with the outputs of 3D human keypoint extraction methods (e.g., BODY_25, etc.).
2) Study the literature on motion retargeting.
3) With 3D coordinates of keypoints from videos of moving children on the input, develop a method that reconstructs joint angle movements. Using keypoints on the face, reconstruct the head direction as well.
4) Filter and smooth the trajectories in joint angle space and animate an infant dummy/kinematic figure.
5) Consider two different target platforms - iCub simulator and MIMo [MAT2022] - and transform the movements to both of them, respecting their specific constraints. Evaluate the accuracy of the retargeting.
6) “Open the eyes” in the robot simulator and visualize what the baby robot sees from its perspective, synchronized with the joint movements. Add also a visualization of the touch sensor activations.

Bibliography / sources:

[1] [GLE1998] M. Gleicher, “Retargeting motion to new characters,” in Proc. of the 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998, pp. 33–42
[2] [HES2018] Hesse, N., Bodensteiner, C., Arens, M., Hofmann, U. G., Weinberger, R., & Sebastian Schroeder, A. (2018). Computer vision for medical infant motion analysis: State of the art and rgb-d data set. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops
[3] [KAN2023] Kanazawa, Yamada, Kuniyoshi et al. (2023): Open-ended movements structure sensorimotor information in early human development, PNAS.
[4] [MAT2022] Mattern, D., López, F. M., Ernst, M. R., Aubret, A., & Triesch, J. (2022, September). MIMo: A Multi-Modal Infant Model for Studying Cognitive Development in Humans and AIs. In 2022 IEEE International Conference on Development and Learning (ICDL) (pp. 23-29). IEEE.
[5] [POL2002] N. S. Pollard, J. K. Hodgins, M. J. Riley and C. G. Atkeson, "Adapting human motion for the control of a humanoid robot," Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), 2002, pp. 1390-1397 vol.2

Name and workplace of bachelor's thesis supervisor:

doc. Mgr. Matěj Hoffmann, Ph.D. Vision for Robotics and Autonomous Systems FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Valentin Marcel, Ph.D. Vision for Robotics and Autonomous Systems FEE

Date of bachelor's thesis assignment: **01.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

doc. Mgr. Matěj Hoffmann, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I want to thank my supervisor Matěj Hoffmann for his professional guidance, and valuable insights, and for always keeping a positive attitude. Also, thanks to Valentin Marcel, Sergiu Popescu, and Filipe Gama for answering all my questions and helping with the correction and writing part. Special thanks to Dongmin Kim for providing me with the Fetus simulator and for his patience with me during learning about it. A big thanks also goes to the people from the lab for creating such a nice and fun work environment. Last but not least, a great thanks goes to my classmates, who endured these difficult times with me, and to my family and especially my mother for providing me with the most needed emotional support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací.

V Praze dne 26. května 2023

....
Ondřej Fiala

Abstract

Motion retargeting describes the process of transferring a source character's motion to a target character, aiming to replicate body poses of the source character or perform specific tasks in the same way as the source character. In this work, we focus on motion retargeting involving infants as source characters and multiple robotic simulators as target characters. For this purpose, we use the iCub, MIMo, and Fetus simulators as target platforms. We present a method that extracts the movement data in the form of joint angles from 3D coordinates of keypoints on infants' bodies by using a set of projections, trigonometric functions, and inverse kinematics. We demonstrate retargeting of the infant's motion to target platforms, while simultaneously activating sensory outputs, to provide visual and tactile feedback. Our results enhance the field of studying infants through the possibility of replaying the movements and exploring the infants' perspective through cameras placed in robots' eyes. Furthermore, the visualization of tactile sensor outputs provides a better understanding of infants' interaction with their environment and provides an observation of the self-touch events. This tool may contribute to an overall understanding of the mechanisms of early infant motor and cognitive development.

Keywords: motion retargeting, infant kinematics, robotic simulators, inverse kinematics, visual feedback, tactile feedback

Supervisor: doc. Mgr. Matěj Hoffmann, Ph.D.

Abstrakt

V této práci se věnujeme procesu převedení pohybu zdrojové postavy na postavu cílovou, cílem tohoto procesu je replikovat pohyb zdrojové postavy, nebo provést konkrétní úlohu stejným způsobem, jakým ji vykonala zdrojová postava. Zaměřujeme se konkrétně na převedení pohybu z kojenců v roli zdrojových postav na modely v robotických simulátorech představující postavy cílové. Jako cílové simulátory pro tento účel používáme konkrétně iCub simulátor, MIMo a Fetus simulátor. Představujeme metodu, která z 3D souřadnic klíčových bodů na těle kojenců extrahuje data pro pohyb ve formě úhlů v kloubech pomocí sekvence projekcí, trigonometrických funkcí a inverzní kinematiky. Následně demontrujeme převedení pohybu na vybrané cílové modely v simulátorech spolu s aktivací výstupů ze senzorů imitujících smysly, abychom byli schopni získat vizuální a hmatovou zpětnou vazbu. Naše výsledky rozšiřují oblast studia kojenců prostřednictvím možnosti opakovaného přehrávání pohybů kojenců. Současně umožňujeme získat informaci o kojencově perspektivě pomocí kamer umístěných v očích robotických modelů. Poskytujeme také vizualiaci z hmatových senzorů umístěných na těle robotických modelů, prostřednictvím které je možno pozorovat, kdy se části těla kojence vzájemně dotýkají. Naše výsledky mohou přispět k celkovému pochopení mechanismů raného motorického a kognitivního vývoje kojenců.

Klíčová slova: převedení pohybu, kinematika kojence, robotické simulátory, inverzní kinematika, vizuální zpětná vazba, hmatová zpětná vazba

Překlad názvu: Převedení pohybů dítěte na humanoidní roboty

Contents

Chapter 1

Introduction

1.1 Motivation

Recent advances in technology have opened up new ways for scientific exploration, offering fresh perspectives on a wide range of topics. One such area of interest focuses on the development of infants and children, consisting not only of the psychological aspects but also of factors like motor skills, self-awareness, or self-exploration. Typically, simulators or robots that replicate the characteristics of real babies, including their proportions, range of motion, or musculoskeletal system, serve as platforms for studying these topics in the field of Developmental Robotics [1]. These platforms often incorporate sensory systems, allowing researchers to enhance their observations through sensory outputs. Furthermore, the integration of Artificial Intelligence techniques such as Reinforcement Learning (RL) algorithms or neural networks offers a unique way to study learning processes and other related aspects. In general, researchers can repeat their experiments or modify various parameters, such as the environment, on these platforms. The overview of various possible contributions of humanoid robots to the study of human cognition and human intelligence was provided by Hoffmann & Pfeifer in [2].

Recently, there has been a growing interest among scientists in studying spontaneous bodily movements and self-body touches [3], which are exhibited by newborns, and even fetuses in the womb. These movements, although not goal-oriented, are believed to play a vital role in the development of a sensorimotor system of infants [4]. By analyzing these movements, researchers may gain a better understanding of learning processes in newborns, as well as a chance to identify developmental disorders at early stages. For example, Kanazawa *et al.* in [5] analyzed the spatial and temporal characteristics of spontaneous bodily movements of infants, revealing their role in structuring sensorimotor interactions.

Our work aims to contribute to this field by presenting a method to transfer captured movement of infants onto various modeling platforms. The transfer process ultimately enables the replication of real babies' movements and the collection of kinematic data of their joints along with the corresponding sensory feedback. By doing so, we gain insight into an infant's perspective, allowing us to observe instances where the infant's body parts make contact and providing a deeper understanding of their proprioceptive skills.

1.2 Goals

The goal of this bachelor thesis is to retarget the infant movements from 3D keypoints extracted from videos onto baby humanoid robots. Movement should involve the entire body, including body parts such as arms, legs, torso, and head. Specifically, the joint angles for each body part should be calculated in each frame, generating a joint angle trajectory. This trajectory should then be filtered to eliminate possible noise and used to animate an infant dummy figure. After the animation of the dummy figure, three target platforms—the iCub simulator, MIMo simulator, and Fetus simulator—should be utilized to transfer the motion onto them. Finally, the robot's perspective should be visualized by extracting the images from the cameras situated in the robot's eyes. Furthermore, the touch sensor activations should also be visualized by extracting the output from tactile sensors placed on the robot's body.

1.3 Problem formulation

In this section, we will provide the formulation of the problem addressed in this work, together with the most important definitions that correspond to the problem. As we said in Section 1.2, our goal is to retarget infant movements onto baby humanoid robots. The input for our work is created by capturing the infant movements by phone cameras and transforming them into 3D keypoints, as described in Section 3.1. A **keypoint** refers to a specific point or landmark on the human body whose position is monitored and recorded throughout the duration of the video. Keypoints are strategically placed on the body to capture its movements accurately; they are usually attached to joints or prominent body features such as eyes or ears. In this case, the **joint** refers to the connections, *e.g.*, knee joints or elbow joints of various body segments, *e.g.*, bones. Joints in the human body can generally be divided into two groups based on their number of Degrees of Freedom (DoF): revolute joints (also called hinge joints) with 1 DoF and spherical joints with 3 DoF.

From the 3D coordinates of the input keypoints, we calculate the joint angles as described in Section 3.4. A **joint angle** is defined as the angle formed between two body segments at a joint and is typically measured in degrees or radians. Then we concatenate these angles for each frame of the video to form a joint angle trajectory. We filter such trajectories to reduce noise and use them to animate an infant dummy figure defined in Section 3.5. The trajectories are later used to control the target platforms: the iCub robot (Section 4.1), the MIMo simulator (Section 4.2) and the Fetus simulator (Section 4.3). From these platforms, we extract the sensory output from tactile sensors and cameras. The result of the retargeting to the target platforms is then evaluated using Median Angle Error (MAE), which describes the error between the direction of the body segment between two input keypoints and the direction of the body segment of a target platform, as explained in Section 5.1.2. In Figure 1.1 there is shown the entire process.

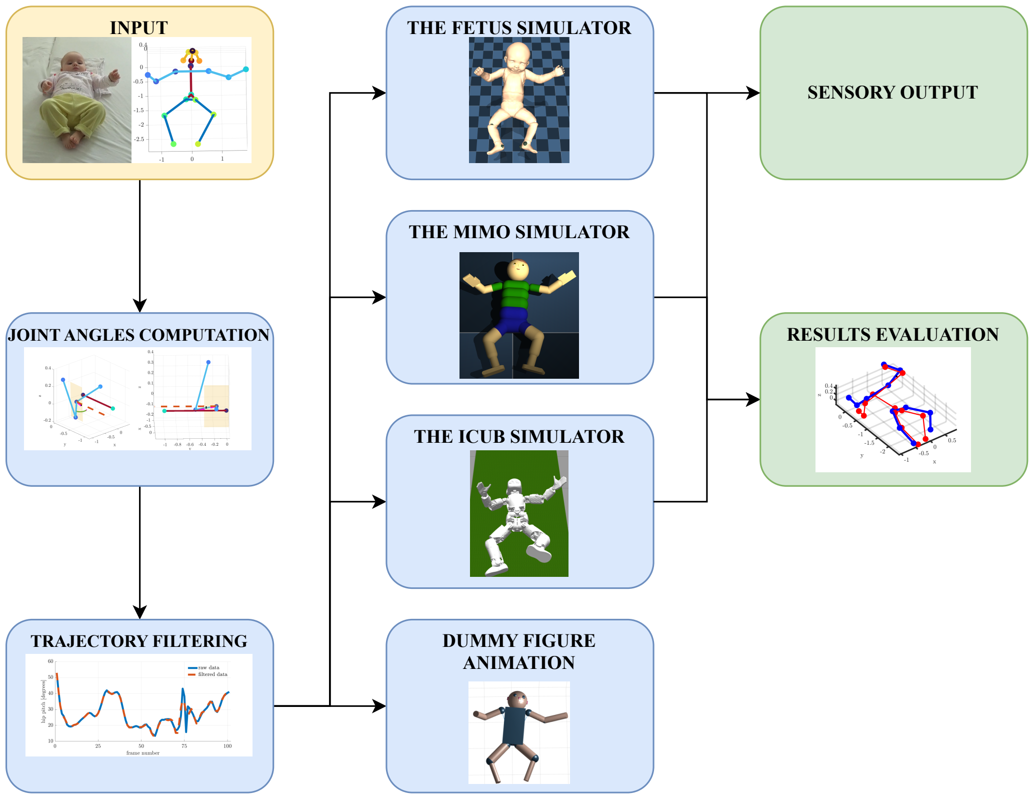


Figure 1.1: Schematics for the problem formulation. Inputs are yellow, processing is blue, and outputs are green.

Chapter 2

Related Work

Motion retargeting is a process of transferring captured movement from a source subject to a target subject and adapting it for the best fit. Although the goal is straightforward, the process itself is complex and has proven to be challenging. This is mainly due to difficulties that arise from differences in the proportions of the source and target subjects. Gleicher [6] presented a technique for adapting the motion of one figure to another figure with an identical structure but different segment lengths, by which he set the grounds for the field of motion retargeting. He utilized Inverse Kinematics (IK) for reconstructing the body pose, as we do in this work, and added a low-pass filter to eliminate the influence of higher frequencies generated by the IK solver. The motion retargeting methods can be further divided by their main focus into two possible categories: upper body motion retargeting and whole body motion retargeting.

Upper body motion retargeting

Gonen *et al.* [7] and Pollard *et al.* [8] have both focused on retargeting human motion to physical robots, such as Nao and Sarcos, with a primary focus on retargeting upper body motion while neglecting the rest of the body. Gonen *et al.* used Microsoft Kinect Markerless Motion Capture System providing them with a set of 3D positions of keypoints, from which they extracted the joint angle values by using trigonometric functions. However, they often used the *arccos* and *arcsin* functions, which we try to avoid, due to their known inaccuracy for small angles. Pollard *et al.* explored limiting conditions for motion retargeting such as constraining the motion to the robot joints, gimbal lock issue, and joint angle and velocity limits, providing us with valuable insight into the problems we could encounter in our work.

Whole body motion retargeting

Retargeting of whole body motions has also been addressed in several works. Riley *et al.* [9] used 3D vision and a human teacher to estimate body postures with inverse kinematics while incorporating the kinematic model of the teacher. Koenemann *et al.* [10] provided a system for online human task imitation without constraining the configuration of legs to be in double support, but rather finding stable configurations using inverse kinematics. Penco *et al.* [11] presented real-time retargeting of the whole-body motion to the iCub robot. However, all of these works focused on maintaining balance while successfully imitating human motion or completing an upper-body task. In this work, we do not consider maintaining balance on two feet because the provided data contains only infants lying in a supine position, supporting themselves with their torsos and not their feet.

Darvish *et al.* [12] used a geometric approach for retargeting the whole body motion to various humanoid robot platforms. They used the IK and nonlinear optimization to compute the human joint angles and velocities from the rotation matrix and then transformed them into the robot joint angles and velocities. Their approach allows transferring human motion to multiple platforms by processing the Unified Robot Description Format (URDF) models of each platform.

Head pose estimation

In addition to whole body retargeting, head pose estimation has also been explored in the context of humanoid robots. It is important when exploring gaze direction or exploring different perspectives. Khan *et al.* [13] introduced a survey of the last ten years of estimation of head pose, where they reviewed the literature of the last decade listing the advantages and disadvantages of existing algorithms and provided a performance comparison of existing frameworks. They have also created a taxonomy of methods based on their approach. For our purpose, the most relevant method is called geometric and requires the existence of facial keypoints such as eyes, nose, ears, etc.

Barra *et al.* [14] presented a rapid method for online head pose estimation using 68 well-known facial landmarks. However, the use of a large number of facial landmarks decreases computational efficiency, which is why some other works limit the number of facial features used. Nikolaidis & Pitas [15] considered the eyebrows, eyes, nostrils, mouth, cheeks, and chin to determine head pose and gaze direction but limited their pose orientation to left or right. Vatahska *et al.* [16] presented a neural network that determines all three head pose angles (pitch, roll, and yaw). They first roughly classify the pose as frontal, left, or right profile, then extract facial features including the nose tip and the eyes. On the basis of these facial features, the neural network estimates rotation angles. Due to the low number of facial features used, this method is computationally highly efficient. However, none of these methods can be used in our work, since our input provides us with only five keypoints at the head (eyes, ears, and nose vertex) and has also not been annotated to use the neural network proposed by Vatashka *et al.*. We note these works here as the possible way to improve our head pose estimation once we are provided with a larger number of facial keypoints.

Infant kinematics

Throughout the motion retargeting process, the kinematics of the source object and the models in target platforms play a significant role. In our case, the kinematics of an infant is important. Karch *et al.* [17] presented a method to capture the movements of infants by the electromagnetic tracking system, the data from this system were later used in calculating the joint centers and joints axes of a kinematic chain model. Kanazawa *et al.* in [5] used a whole body motion capture and obtained the movements of infants body joints via inverse kinematics using OpenSim [18] that allows developing models of musculoskeletal structures and is used to create dynamic simulations of a variety of movements.

In conclusion, the topic of motion retargeting was heavily studied from different points of view, using various techniques including the geometric way with trigonometric calculations of joint angles [7], [12], as well as estimating joint angles using neural networks [16]. The subject of motion retargeting is usually an adult performing a specific task, which defines the straightforward goal of the retargeting process: to successfully perform the task on a robotic platform. In contrast, our work focuses on infant motion retargeting, which has not been extensively explored before. Infants' movements are characterized by spontaneous actions rather than specific tasks; therefore, our goal is to accurately retarget the orientation of infants' limbs into various platforms.

Chapter 3

Modeling Infant Movements with Joint Angles

The code used to achieve what is described in this thesis, together with the visual materials, is available in a dedicated Gitlab repository [19].

The joint angles defined in Section 1.3 play a crucial role in describing the angular motion of the joint, providing valuable insight into the movement of the human body. Furthermore, joint angles are essential in controlling robotic simulators, as the motion of the robot or motor positions are often controlled by them. The entire process of calculating the joint angles involves representing the human body as a set of interconnected segments and determining the angles between those segments at specific points. In this chapter, the infants' datasets used for the whole process of motion retargeting will be defined and the necessary mathematical apparatus will be provided. We will define all the body joint angles and show a typical movement of the human limbs corresponding to them. Finally, we will show a way of calculating such angles and present an infant dummy figure, used for visualization of the calculated angles.

3.1 Input datasets to the motion retargeting

Before we proceed to the description of how we calculate each of the body joint angles, it is essential to provide an overview of the input data format. We utilize two distinct datasets for our analysis. The first dataset is the Moving INfants In RGB–D (MINI–RGBD) dataset of synthetic infants formed by Hesse *et al.* [20]. This dataset consists of 12 videos that capture the movements of synthetic infants during their early developmental stages. The dataset is further divided into three categories based on the difficulty of the observed movements. These categories span from simple actions such as lying on the back and moving the limbs, to more complex actions involving limbs touching each other and the infant turning to the sides.

The second dataset, Babies to Baby Humanoids Motion Retargeting (BBHMR) dataset, consists of a subset of 12 video recordings following one real infant, name code TH, primarily in a supine position, between 8 and 23 weeks old, and recorded at home by parents with phone cameras at 25 Frames Per Second (FPS), at different camera distances and angles between sessions. These video recordings were processed through the 2D pose estimation and keypoint extraction method OpenPose [21], then through 2D-to-3D lifting with SMPLify-x [22], using the SMIL infant model [23]. It is a subset of the data from [24], used to study infants' spontaneous behaviors. Videos of infants from BBHMR dataset are not public, but the extracted keypoints are available at the dedicated Gitlab repository [19].

In both the MINI-RGBD and BBHMR datasets, supplementary files are provided that contain three-dimensional coordinates of keypoints for each frame of the video. The BBHMR dataset is processed through OpenPose and therefore there are 25 different keypoints as in the standardized OpenPose body format BODY_25 [21]. The MINI-RGBD dataset also contains 25 keypoints, but different ones than those defined by the BODY_25 format. In Table 3.1 the overview of the keypoints utilized in our work is shown, together with their corresponding names in the MINI-RGBD and BBHMR datasets. If the keypoint is not present in the dataset, it is marked as Not Available (N/A).

keypoint	**abbreviation**	**MINI-RGBD**	**BBHMR**
left hip	*lh*	left thigh	left hip
right hip	*rh*	right thigh	right hip
mid hip	*mh*	spine	mid hip
left knee	*lk*	left calf	left knee
right knee	*rk*	right calf	right knee
left ankle	*la*	left foot	left ankle
right ankle	*ra*	right foot	right ankle
neck	*nck*	neck	neck
left shoulder	*lsh*	left shoulder	left shoulder
right shoulder	*rsh*	right shoulder	right shoulder
left elbow	*lel*	left fore arm	left elbow
right elbow	*rel*	right fore arm	right elbow
left wrist	*lwr*	left hand	left wrist
right wrist	*rwr*	right hand	right wrist
nose	*ns*	nose	nose
right eye	*re*	N/A	right eye
left eye	*le*	N/A	left eye
right ear	*rer*	N/A	right ear
left ear	*ler*	N/A	left ear

Table 3.1: General keypoints used in our work put together with provided datasets.

As shown in Table 3.1, the MINI-RGBD dataset contains only one keypoint located on the infant's head, which is insufficient to accurately reconstruct the head pose. Furthermore, this dataset is limited to 12,000 frames, whereas the BBHMR dataset includes 129,914 frames. Due to these limitations, the MINI-RGBD dataset was used only for visual accuracy verification purposes. The results, errors, and visual and touch outputs were obtained exclusively using the BBHMR dataset. In Figure 3.1 there are shown the keypoints from BBHMR dataset. We added a middle-ear keypoint for better plotting of the body skeleton.

When working with the three-dimensional coordinates of a keypoint from Table 3.1, we will use the following notation:

$$k_{\mathrm{ab}} = \begin{bmatrix} x_{\mathrm{ab}} & y_{\mathrm{ab}} & z_{\mathrm{ab}}, \end{bmatrix} \tag{3.1}$$

where k represents a keypoint with its abbreviation **ab**, and x, y and z represent its corresponding coordinates.

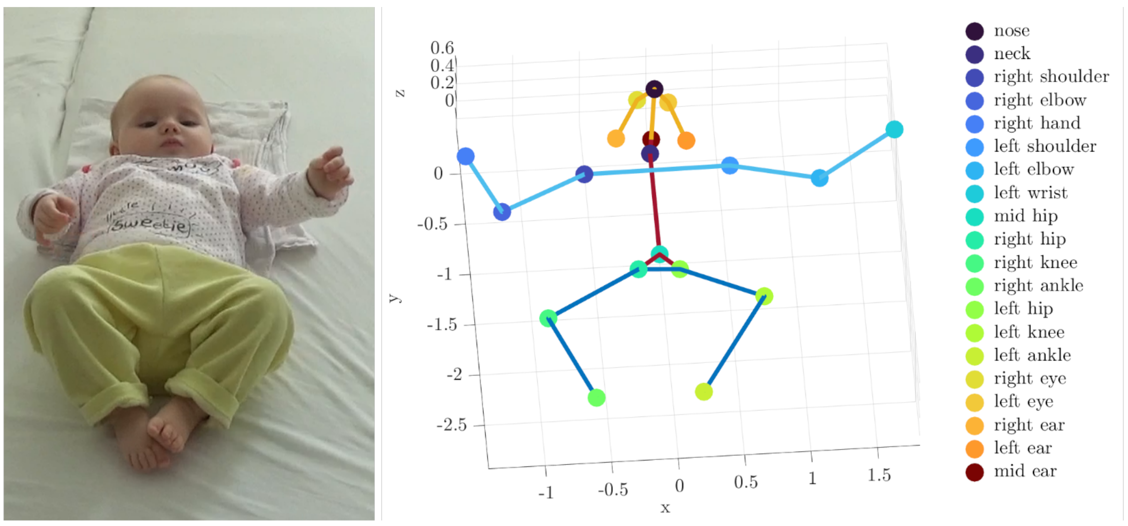


Figure 3.1: Infant with its extracted keypoints as the input to the motion retargeting process.

3.2 Math apparatus

Rigid body transformations

One of the key concepts in our work is rigid body transformations, which refers to a sequence of rotations and translations that allow us to map a set of points to another set of points while preserving the distance between the points. In three-dimensional space, this is usually achieved with rotations about specified axes and translations along these axes. First, when applying rotations and translations in 3D, homogeneous coordinates must be used. A point p and general transformational matrix $\mathbf{T}$ in homogeneous coordinates are represented by:

$$p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{14} \\ r_{21} & r_{22} & r_{23} & t_{24} \\ r_{31} & r_{32} & r_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

The top left 3×3 matrix in $\mathbf{T}$ is the rotation matrix $\mathbf{R}$, while t_{14}, t_{24}, t_{34} represent x, y and z translation coordinates. The rotation matrix $\mathbf{R}$ must be orthogonal, and the following rules must be satisfied:

$$\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}, \qquad \det \mathbf{R} = 1, \tag{3.3}$$

where $\mathbf{I}$ denotes the identity matrix. The most common rotations are rotations about the base coordinate axes x, y and z and they are represented as follows, where ϕ denotes the angle of rotation:

$$\mathbf{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},$$
$$\mathbf{R_y}(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix},$$
$$\mathbf{R_z}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.4}$$

While above, there are typical rotation matrices about the base axes, sometimes the rotation about a different set axis is needed. For this purpose, we use the **Axis-Angle representation** of rotation. However, this representation does not give us an easy way to apply the rotation to the vector; therefore, conversion from Axis-Angle representation to rotation matrix is important. Let $\mathbf{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}$ be an axis of rotation and ϕ an angle of rotation, the conversion to rotation matrix $\mathbf{R}$ is done followingly:

$$\mathbf{R} = \mathbf{I} + [\mathbf{s}]_x \cdot \sin\phi + [\mathbf{s}]_x^2 \cdot (1 - \cos\phi), \tag{3.5}$$

where $[\mathbf{s}]_x$ denotes skew-symetric matrix defined as:

$$[\mathbf{s}]_x = \begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}. \tag{3.6}$$

Another representation of rotation in three-dimensional space is representation with the use of **Roll, Pitch, Yaw**, which is a standard representation used in humanoid kinematics for a description of spherical joints. Angles represent three sequential right-handed rotations, where the roll signifies rotation about the z axis by an angle ϕ, pitch corresponds to rotation about the new y axis by an angle θ, and finally, yaw corresponds to rotation about the new x axis by an angle ψ [25]. Fig. 3.2 provides an example of these rotations, we want to rotate the base coordinate system denoted with bright colors to the final coordinate system denoted with dull colors. First, the roll is applied, then the pitch, and finally the yaw, resulting in the successful rotation.

Signed angle between vectors

Usually, when finding the rigid body transformation from one vector to another, the first step is to determine the signed angle between the two vectors. In 2D, the signed angle between vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ can be easily determined using the following equation:

$$\phi = \operatorname{arctan2}(v_{1y} - v_{2y}, v_{1x} - v_{2x}), \tag{3.7}$$

because **arctan2** function is available in all four quadrants.

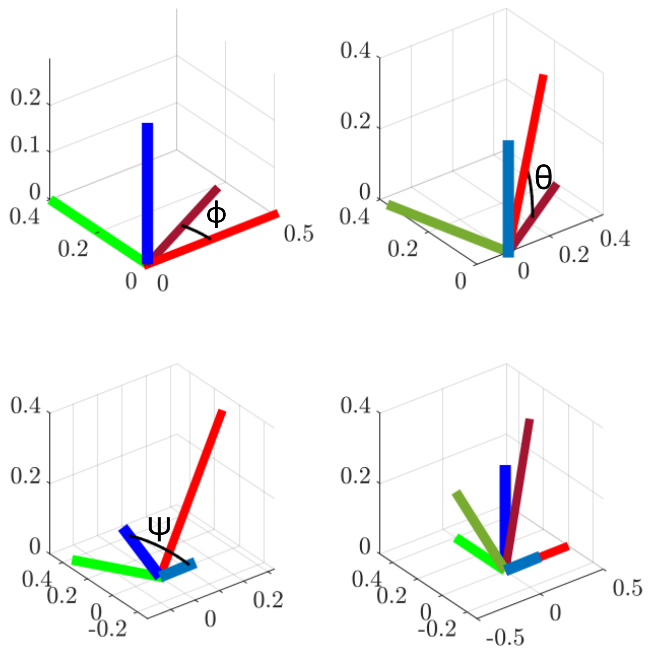


Figure 3.2: Example of using roll(ϕ)—pitch(θ)—yaw(ψ) representation for 3D rotations from one coordinate frame (bright colors) to another (dull colors).

However, finding the rigid body transformation in three-dimensional space is more complicated than in 2D because the orientation of the angle needs to be taken into account. To do this, the normal vector of the plane of rotation, denoted as $\mathbf{v_n}$, is required. The plane of rotation is defined by the two vectors, usually, by the one that needs to be rotated and by the one to which it needs to be rotated. The signed angle can then be determined by the following equations:

$$\phi = \arccos \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{||\mathbf{v_1}||||\mathbf{v_2}||}, \text{for } \mathbf{n} \cdot (\mathbf{v_1} \times \mathbf{v_2}) > 0, \tag{3.8}$$

$$\phi = -\arccos \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{||\mathbf{v_1}||||\mathbf{v_2}||}, \text{for } \mathbf{n} \cdot (\mathbf{v_1} \times \mathbf{v_2}) < 0. \tag{3.9}$$

The **arccos** function is numerically unstable, and it is recommended to use the **arctan2** function instead, in the following paragraphs, the equation using the **arctan2** function will be presented. From the definitions, we can write

$$\mathbf{v_1} \times \mathbf{v_2} = \|\mathbf{v_1}\|\|\mathbf{v_2}\| \sin(\phi)\ n, \tag{3.10}$$

$$\mathbf{v_1} \cdot \mathbf{v_2} = ||\mathbf{v_1}||||\mathbf{v_2}|| \cos \phi \tag{3.11}$$

where ϕ is the angle between $\mathbf{v_1}$ and $\mathbf{v_2}$, respectively between 0° and 180° and n is a unit vector perpendicular to the plane containing the two vectors and has the orientation respecting the right-hand rule. Let us define the signed angle for which we are looking as ψ, which is determined either as $\psi = \phi$ or $\psi = 360° - \phi$, depending on the value of ϕ. Then we can write

$$\mathbf{v_1} \cdot \mathbf{v_2} = \|\mathbf{v_1}\|\|\mathbf{v_2}\| \cos\phi = \|\mathbf{v_1}\|\|\mathbf{v_2}\| \cos(-\phi) = \|\mathbf{v_1}\|\|\mathbf{v_2}\| \cos(360° - \phi) = \|\mathbf{v_1}\|\|\mathbf{v_2}\| \cos\psi.$$

The cross product of two vectors produces a vector that is perpendicular to them, so our normalized final vector $\mathbf{v_n}$ has either the same direction as n or exactly the opposite, mathematically expressed:

$$\mathbf{n} \cdot \mathbf{v_n} = 1 \;\text{ when } \psi < 180°, \qquad \text{or} \qquad \mathbf{n} \cdot \mathbf{v_n} = -1 \text{ when } \psi > 180°,$$

and therefore

$$\begin{aligned} (\mathbf{v_1} \times \mathbf{v_2}) \cdot \mathbf{v_n} &= \|\mathbf{v_1}\|\|\mathbf{v_2}\| \sin(\phi)\; \mathbf{n} \cdot \mathbf{v_n}, \\ (\mathbf{v_1} \times \mathbf{v_2}) \cdot \mathbf{v_n} &= \|\mathbf{v_1}\|\|\mathbf{v_2}\| \sin\psi, \end{aligned} \tag{3.12}$$

because $\mathbf{n} \cdot \mathbf{v}_n$ denotes the sign of **sine** function. If we take Equations 3.12 and 3.11 and divide them, we get:

$$\frac{\sin\psi}{\cos\psi} = \tan\psi = \frac{(\mathbf{v_1} \times \mathbf{v_2}) \cdot \mathbf{v_n}}{\mathbf{v_1} \cdot \mathbf{v_2}}, \tag{3.13}$$

allowing us to compute the signed angle ψ as [26]:

$$\psi = \operatorname{arctan2}\left(\frac{(\mathbf{v_1} \times \mathbf{v_2}) \cdot \mathbf{v_n}}{\mathbf{v_1} \cdot \mathbf{v_2}}\right). \tag{3.14}$$

Vector projection onto a plane

The signed angle between two vectors is generally calculated in a plane. This plane can be either formed by the two given vectors or defined by its normal vector, which can be obtained from the cross-product of the two vectors that are in that plane. In the second case, if we want to calculate the angle in the given plane, we need to project the vectors onto that plane. The following text will provide a mathematical description of the process of projecting a vector onto a plane, given its normal vector.

The vector projection of the vector $\mathbf{v}$ onto another vector $\mathbf{n}$ can be mathematically described by the following equation:

$$\mathbf{v}_{\text{projected\_on\_n}} = \frac{\mathbf{v_1} \cdot \mathbf{n}}{\|\mathbf{n}\|^2} \cdot \mathbf{n}. \tag{3.15}$$

Every vector $\mathbf{v}$ can be expressed as the sum of two other vectors that are perpendicular to each other. These two perpendicular vectors are called the orthogonal components of $\mathbf{v}$. When calculating the projection of a vector $\mathbf{v}$ on a plane P defined by its normal vector $\mathbf{n}$, the projection to the normal vector can be calculated first. The resulting value is then subtracted from the vector $\mathbf{v}$, leaving us with the projection of vector $\mathbf{v_1}$ onto the plane P:

$$\mathbf{v}_{\text{projected\_on\_}P} = \mathbf{v} - \frac{\mathbf{v} \cdot \mathbf{n}}{\|\mathbf{n}\|^2} \cdot \mathbf{n}. \tag{3.16}$$

3.3 Definition of the joint angles

Generally, when we want to describe the orientation of an object in three-dimensional space, its rotation about three perpendicular axes needs to be specified. The three angles that describe this rotation are commonly known as Roll, Pitch, and Yaw and were defined in Section 3.2. In robotics and humanoid kinematics, these angles are widely used to specify the orientation of the end-effector or joints of a robot arm. They can also be combined to form a rotation matrix, which can be used to transform a vector or a point from one coordinate frame to another.

In addition, joints can be divided into groups according to their number of DoF. Two important joint groups in the human body are revolute joints and spherical joints, which both represent rotations. The revolute joints have one DoF, such as the elbow flexion and extension joint. The spherical joints have three DoFs, such as the shoulder joint. This classification provides insight into the structure of the joints, which is crucial in designing the limbs of humanoid robots. When defining body angles, both the spherical and revolute joints will be considered.

In this section, the body angles will be further defined, and their direction and correspondence to the important movements of the body parts will also be discussed. The following paragraphs are divided into four parts: arms, legs, torso, and head.

Arm joint angles

For retargeting arm movements, we consider three separate joints: shoulder joint, elbow joint, and wrist. We treat the shoulder joint as a spherical joint with 3 DoF and the elbow joint as a revolute joint with 1 DoF, while the wrist is in our case an immovable joint considered as an end effector because the input datasets defined in Section 3.1 do not provide us with enough keypoints to successfully calculate the motion of the wrist. With that in mind, we define four arm joint angles for arm movement: shoulder roll, pitch and yaw angles, and elbow angle. Fig. 3.3 illustrates all significant movements of the human arm that are further discussed below.

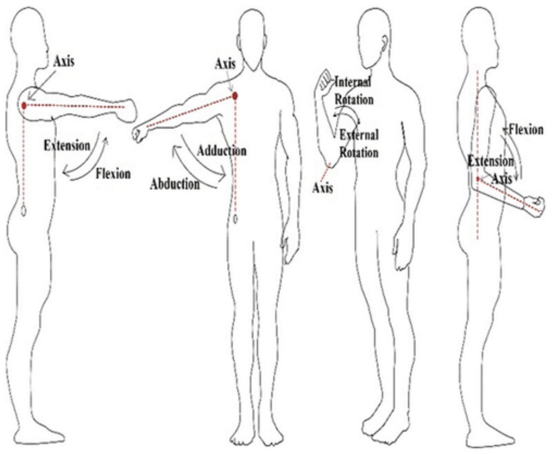


Figure 3.3: Different movements for human upper-arm. Image obtained from [27].

The shoulder roll corresponds to the abduction-adduction movement, with a positive value of an angle indicating abduction and a negative value indicating adduction. Shoulder pitch illustrates extension and flexion movement and exhibits a positive value of an angle during flexion. Shoulder yaw corresponds to the internal/external rotational movement, with a positive value of an angle indicating internal rotation. The elbow angle describes the extension or flexion of the lower arm, with a positive value representing flexion.

Leg joint angles

Similar to the definition of arm joint angles, we can divide the human leg into two movable joints and one end effector, in this case, the hip joint, knee joint, and ankle end effector. The motion of an ankle is not considered due to the consistency of DoF between the arms and legs. The hip joint is treated as spherical and the knee joint as a revolute joint. Accordingly, we characterize leg movements with four leg joint angles: hip roll, pitch, and yaw angles, and knee angle. Various possible movements of the human hip and lower leg are depicted in Fig. 3.4.

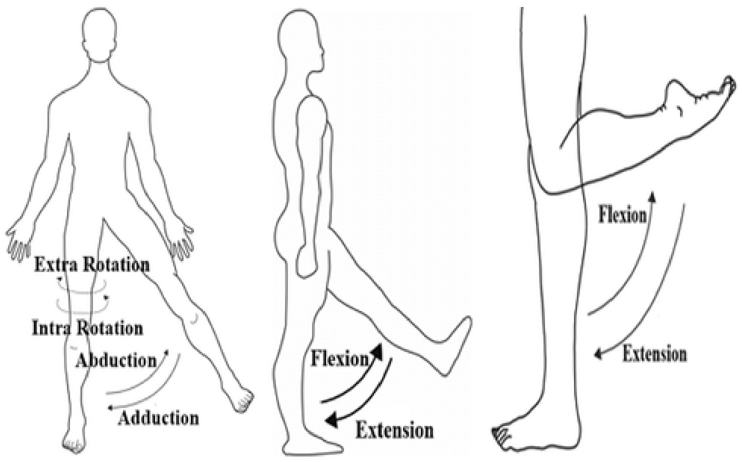


Figure 3.4: Different movements of human hip and lower leg. Image obtained from [28].

The hip roll corresponds to abduction-adduction movement of the leg, with a positive value of an angle indicating abduction. Hip pitch, on the other hand, represents flexion and extension movement, with a positive value of an angle indicating flexion. The hip yaw is associated with internal/external rotation, with a positive value of an angle indicating internal rotation. The knee angle describes the extension or flexion of the lower leg, with a positive value representing the flexion.

Torso joint angles

Similar to the limbs, the torso joint can also be classified into roll, pitch, and yaw movement. Similar to the neck, the torso pitch corresponds to tilting and bringing shoulders towards both hips, and a positive value of the hip pitch angle indicates the direction towards the hips. The torso roll then represents tilting one shoulder towards a corresponding hip and away from the other hip, with a positive value of an angle indicating tilting towards the right side.

Torso yaw is the rotation of the upper body about its vertical axis, with a positive value indicating a clockwise rotation viewed from above. This movement is similar to the motion of twisting the torso, such as when turning to look over the shoulder. All these movements are shown in Fig. 3.5.

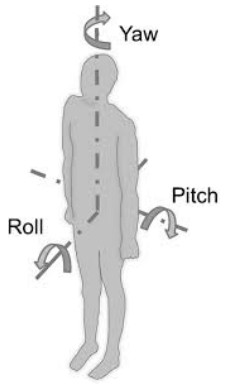


Figure 3.5: Definition of torso movements. Image obtained from [29].

Neck joint angles

As mentioned above, most joints in the human body joints can be separated into roll, pitch, and yaw movements, and the head or neck joint is no exception. In this case, the neck pitch corresponds to the tilting motion of the head towards both shoulders, and a positive value of the neck pitch angle indicates the direction away from the shoulders. The neck roll describes tilting the head toward one shoulder and away from the other, with a positive value of an angle indicating tilting towards the right shoulder. The neck yaw corresponds to head rotation from the right to the left side, and vice versa. One can imagine rotating their head as saying “no“, with a positive value of the yaw angle signifying rotating to the left. For a better understanding, refer to Fig. 3.6.

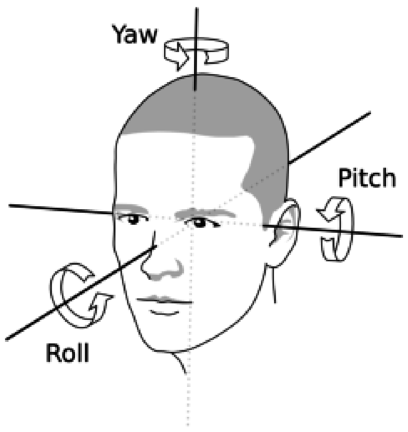


Figure 3.6: Definition of head movements. Image obtained from [30].

3.4 Calculation of the joint angles

Joint angles are defined as the angles formed between two body segments at a joint. However, the input datasets provide us only with 3D coordinates of keypoints, the body segments have yet to be extracted. Since the extraction is done by subtracting the coordinates of two keypoints, which is a mathematical process to create a vector, we will call the body segments "body vectors“. All important body vectors that will be used later are listed in Table 3.2.

body vector	calculation	direction
spine vector	$\mathbf{v_{sp}} = k_{\text{nck}} - k_{\text{mh}}$	midhip to neck
lower spine vector	$\mathbf{v_{lsp}} = k_{\text{mh}} - \frac{k_{\text{lh}}+k_{\text{rh}}}{2}$	middle of the left and right hip to midhip
shoulder vector	$\mathbf{v_{sh}} = k_{\text{lsh}} - k_{\text{rsh}}$	right shoulder to left shoulder
left upper arm vector	$\mathbf{v_{lua}} = k_{\text{lel}} - k_{\text{lsh}}$	left shoulder to left elbow
right upper arm vector	$\mathbf{v_{rua}} = k_{\text{rel}} - k_{\text{rsh}}$	right shoulder to right elbow
left forearm vector	$\mathbf{v_{lfa}} = k_{\text{lwr}} - k_{\text{lel}}$	left elbow to left wrist
right forearm vector	$\mathbf{v_{rfa}} = k_{\text{rwr}} - k_{\text{rel}}$	right elbow to right wrist
hip vector	$\mathbf{v_{hp}} = k_{\text{lh}} - k_{\text{rh}}$	right hip to left hip
left thigh vector	$\mathbf{v_{lth}} = k_{\text{lh}} - k_{\text{lk}}$	left knee to left hip
right thigh vector	$\mathbf{v_{rth}} = k_{\text{rh}} - k_{\text{rk}}$	right knee to right hip
left calf vector	$\mathbf{v_{clf}} = k_{\text{lk}} - k_{\text{la}}$	left ankle to left knee
right calf vector	$\mathbf{v_{clf}} = k_{\text{rk}} - k_{\text{ra}}$	right ankle to right knee
eyes vector	$\mathbf{v_{eyes}} = k_{\text{le}} - k_{\text{re}}$	right eye to left eye
neck to left eye vector	$\mathbf{v_{eyeL}} = k_{\text{le}} - k_{\text{nck}}$	neck to left eye
neck to right eye vector	$\mathbf{v_{eyeR}} = k_{\text{re}} - k_{\text{nck}}$	neck to left right

Table 3.2: Important body vectors.

Each body vector in 3D space has its own direction when calculating joint angles for a specific frame of the video. For example, consider a person's arm pointing towards an object in front of them. However, when all the joint angles are set to zero, the body vector achieves a reference direction. In the case of the arm, this *reference direction* would align with the direction of gravity when the person is standing upright as if all the muscles in the arm were relaxed. The orientation of this reference frame is crucial in determining the corresponding joint angle, as it is relative to the reference direction. Therefore, when calculating the angles for a specific body part, vectors from other body parts may be necessary to establish the reference position for the vectors of the specific body part.

In the following sections, we will describe the process of calculation of the body joint angles. Sections are divided into four groups according to the corresponding body part: arm, leg, torso, and neck.

3.4.1 Arm angles

To estimate the joint angles in the arm limb, the following keypoints are needed: spine, neck, right and left shoulder, right and left elbow, and right and left wrist. An example of a visualization of these keypoints in 2D and 3D is shown in Figure 3.7. The keypoints are then used to calculate the vectors defined in Table 3.2. The vectors required to calculate arm angles are the vectors of the spine, shoulders, upper arms, and forearms.

The arm can be divided into two joints—shoulder and elbow joints—and an immovable joint considered as an end effector—wrist joint, as mentioned in Section 3.3. We can divide arm movements into two groups based on the joint position that they directly influence. The elbow position and the direction of the upper arm vector are controlled by the shoulder pitch and roll, while the wrist position and the direction of the forearm vector are controlled by the elbow flexion and shoulder yaw.

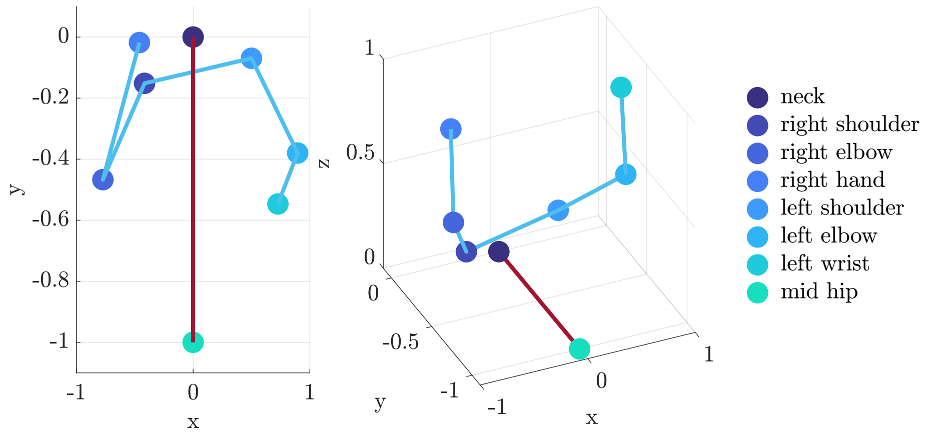


Figure 3.7: Upper body keypoints in 2D and 3D.

Shoulder pitch and shoulder roll

Firstly, the position of the elbow joint needs to be estimated and matched to the position of the elbow keypoint, which is the input. To achieve that, shoulder pitch and shoulder roll angles need to be calculated. First, the Cartesian coordinates of the upper body keypoints will be extracted, and necessary vectors will be calculated. Then, a series of plane projections will be performed, and the angles between those projections will be calculated to accurately determine the values of shoulder pitch and roll angles.

Shoulder pitch represents a front-back movement, meaning that the arm moves in a plane defined by the normal vector, represented by the shoulder vector defined in Table 3.2. The reference position for a zero angle is from the neck directed towards the location of midhip, in other words, the direction of gravity vector when standing upright. The reference vector is represented by the spine vector, which is projected onto the plane along with the upper arm vector. This procedure is visually depicted in Figure 3.8. After projection, the angle between these two projected vectors is calculated using Equation 3.14 and denoted as α_{pitch}.

Shoulder roll refers to the adduction-abduction movement of the arm. Therefore, the reference vector for the shoulder roll is the same as for the shoulder pitch, but the plane in which the movement is performed differs. During the calculation of the shoulder pitch angle, the upper arm vector projected onto the plane defined by the shoulder vector is saved and used to calculate the shoulder roll. The angle between the defined upper arm

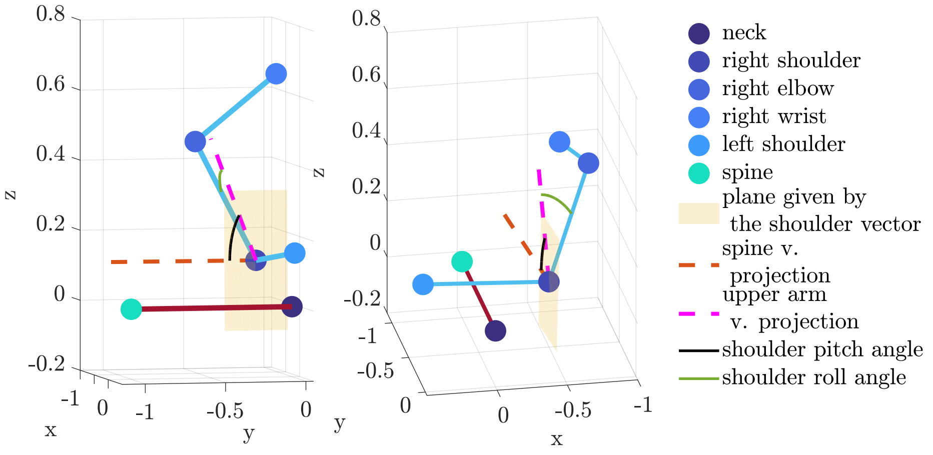


Figure 3.8: Calculation of shoulder pitch and roll angles.

vector and the projected vector is then calculated using the equation below:

$$\alpha_{\text{roll}} = \arccos \frac{\mathbf{v}_{\text{ua}} \cdot \mathbf{v}_{\text{proj}}}{\|\mathbf{v}_{\text{ua}}\| \|\mathbf{v}_{\text{proj}}\|}. \tag{3.17}$$

We considered using the same procedure to calculate shoulder roll as for pitch, which would involve projecting the upper arm vector and the spine vector to a plane given by the normal vector as the cross product of the shoulder vector and the spine vector. However, we found that the process described in the previous paragraph was more effective since it does not require any process of projection.

Shoulder horizontal and shoulder elevation

As mentioned earlier, spherical joints are typically controlled by roll, pitch, and yaw angles. However, some robotic platforms choose to replace the roll and pitch of the shoulder joint with separate joints called shoulder horizontal and shoulder elevation. This topic is discussed in more detail in Chapter 4. The shoulder elevation angle represents the angle between the upper arm vector and the spine vector, which can be calculated with the following equation:

$$\alpha_{\text{elevation}} = \arccos \frac{\mathbf{v}_{\text{ua}} \cdot \mathbf{v}_{\text{sp}}}{\|\mathbf{v}_{\text{ua}}\| \|\mathbf{v}_{\text{sp}}\|}. \tag{3.18}$$

On the other hand, the shoulder horizontal then denotes the movement of the upper arm in a plane defined by the spine vector. In this case, the reference vector is the shoulder vector. The shoulder vector is projected onto the plane and the upper arm vector is projected also onto the same plane. The angle between these two projected vectors is calculated using Equation 3.14 and is referred to as shoulder horizontal angle $\alpha_{\text{horizontal}}$, as is shown in Figure 3.9.

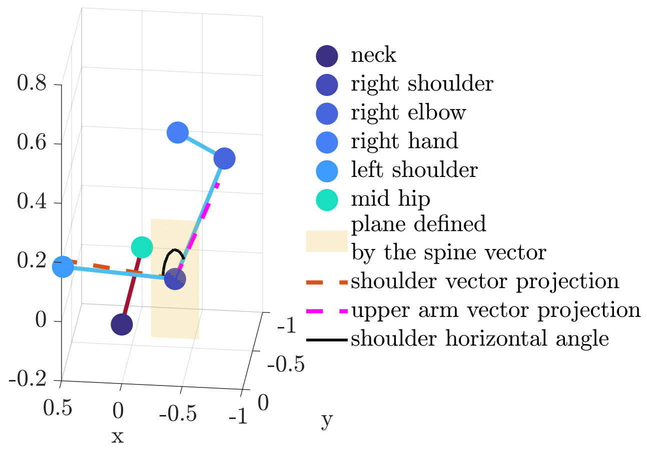


Figure 3.9: Calculation of shoulder horizontal angle.

Shoulder yaw and elbow angle

As mentioned before, two additional angles are required to match the position of the wrist or the direction of the forearm: elbow angle and shoulder yaw. First, we will focus on the elbow angle. Since a motion of an arm produced by changing the elbow angle is situated only in the plane given by the 3 exact points—shoulder, elbow, and wrist—the only thing needed for calculating elbow angle is the angle between the vector of an upper arm and the vector of a lower arm. We can then calculate the value of an angle as shown bellow:

$$\alpha_{\text{elbow}} = \arccos \frac{\mathbf{v}_{\text{ua}} \cdot \mathbf{v}_{\text{la}}}{\|\mathbf{v}_{\text{ua}}\|\|\mathbf{v}_{\text{la}}\|}. \tag{3.19}$$

Shoulder yaw refers to the rotation of the upper arm vector about itself, which only affects the position of the wrist joint when the elbow angle is not exactly 0 degrees. If the elbow angle is 0 degrees, the shoulder yaw angle cannot be determined. If the angle is not 0 degrees, the forward kinematics of an arm limb is performed, with the yaw angle set to 0. To start this process, the base coordinate system is set with its origin at the shoulder and the z axis aligned with the direction of the shoulder vector, as shown in Figure 3.10.

That is accomplished using the following equation:

$$\mathbf{B} = \mathbf{R}_{\text{ot}}\mathbf{R}_{\mathbf{z}}\left(-\frac{\pi}{\mathbf{2}}\right)\mathbf{R}_{\mathbf{x}}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{R}_{\mathbf{y}}\left(-\frac{\pi}{\mathbf{2}}\right) \quad \text{for the left arm,} \tag{3.20}$$

$$\mathbf{B} = \mathbf{R}_{\text{ot}}\mathbf{R}_{\mathbf{z}}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{R}_{\mathbf{x}}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{R}_{\mathbf{y}}\left(\frac{\pi}{\mathbf{2}}\right) \quad \text{for the right arm,} \tag{3.21}$$

where $\mathbf{R}_{\text{ot}}$ represents the axis-angle rotation (converted to rotation matrix by Equation 3.5) of the $\mathbf{y}$ vector $\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$ about an axis gained with the cross product of the $\mathbf{y}$ vector and the shoulder vector. The angle of rotation is determined by the angle between the $\mathbf{y}$ vector and the shoulder vector. The symbol $\mathbf{R}_{\mathbf{a}}$ represents a rotation about the axis denoted as

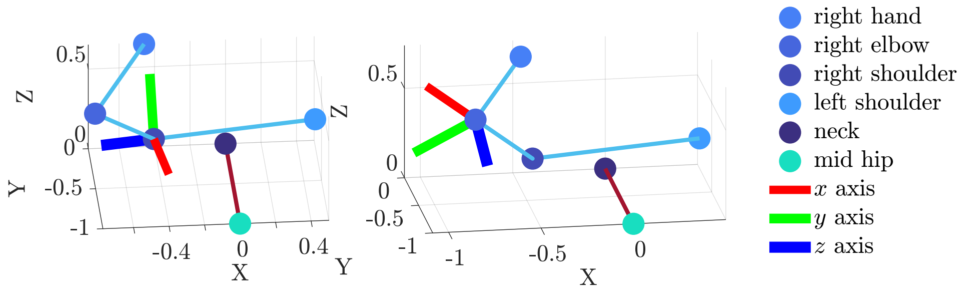


Figure 3.10: Steps while calculating the shoulder yaw. Setting the base frame on the left and applying pitch and roll on the right.

a. $\mathbf{B}$ is a base matrix, which includes the x, y, and z axis of the coordinate frame that it represents in its upper-left 3×3 matrix, the last column of this matrix corresponds to the coordinates of the origin of an axis system represented by $\mathbf{B}$.

Then the rotation for pitch angle and roll angle is applied, followed by the translation to the elbow joint as you can see in Figure 3.10:

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(-\alpha_{\text{pitch}})\mathbf{R_x}\left(\frac{\pi}{2}\right)\mathbf{R_z}(\alpha_{\text{roll}})\mathbf{T_x}(\|\mathbf{v_{rua}}\|), \quad \text{for the right arm} \tag{3.22}$$

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(-\alpha_{\text{pitch}})\mathbf{R_x}\left(-\frac{\pi}{2}\right)\mathbf{R_z}(\alpha_{\text{roll}})\mathbf{T_x}(\|\mathbf{v_{lua}}\|), \quad \text{for the left arm.} \tag{3.23}$$

Next, the rotation for the elbow angle is performed followed by the translation to the wrist joint:

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(\alpha_{\text{elbow}})\mathbf{T_x}(\|\mathbf{v_{rla}}\|), \quad \text{for the right arm,} \tag{3.24}$$

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(\alpha_{\text{elbow}})\mathbf{T_x}(\|\mathbf{v_{lla}}\|), \quad \text{for the left arm.} \tag{3.25}$$

From this the zero-yaw wrist position can be extracted as the origin of the coordinate system represented by $\mathbf{B}$, and the zero-yaw forearm vector can be calculated. Then, this vector and the input lower arm vector are projected to the plane defined by the x axis of the coordinate system obtained after completing the pitch and roll rotations. The angle between them is calculated using Equation 3.14 and is indicated as α_{yaw}, as shown in Figure 3.11.

However, this procedure can only be used to calculate the angles of the arm joints with the motions of the shoulder pitch and roll. For the second approach, shoulder horizontal and elevation movements, the forward kinematics differs. The Equations 3.20 and 3.21 are

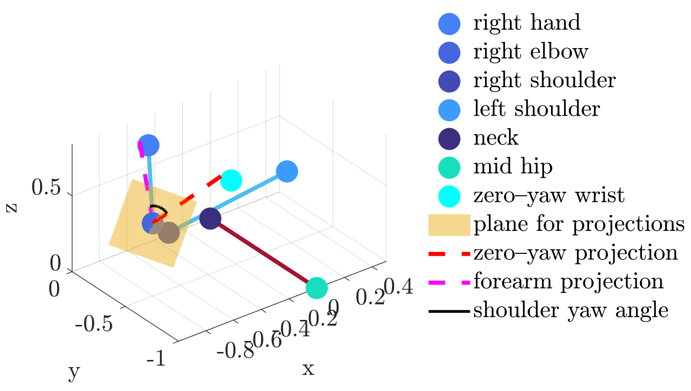


Figure 3.11: Calculating yaw shoulder yaw.

replaced with the following equations to set the position of the base coordinate system.

$$\mathbf{B} = \mathbf{R}_{\mathrm{ot}}\mathbf{R_z}\left(-\frac{\pi}{\mathbf{2}}\right)\mathbf{R_x}\left(\frac{\pi}{\mathbf{2}}\right) \quad \text{for the left arm,} \tag{3.26}$$

$$\mathbf{B} = \mathbf{R}_{\mathrm{ot}}\mathbf{R_z}\left(\frac{\pi}{\mathbf{2}}\right)\mathbf{R_x}\left(\frac{\pi}{\mathbf{2}}\right) \quad \text{for the right arm.} \tag{3.27}$$

The Equations 3.22 and 3.23 for applying the pitch and roll are changed to equations characteristic for applying the horizontal and elevation angle:

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(\alpha_{\mathrm{horizontal}})\mathbf{R_x}\left(\frac{\pi}{2}\right)\mathbf{R_z}\left(\frac{\pi}{2} - \alpha_{\mathrm{elevation}}\right)\mathbf{R_x}\left(\frac{-\pi}{2}\right)\mathbf{T_x}(\|\mathbf{v_{ua}}\|) \tag{3.28}$$

for both arms.

Finally, using Equations 3.25 and 3.24 the elbow flexion is applied and the translation to the wrist joint is done. Then the zero-yaw wrist position is extracted, and the process continues as described above.

3.4.2 Leg joint angles

To estimate the joint angles in the leg limb, the following keypoints are needed: spine, neck, right and left hip, right and left calf, and right and left foot. From the 3D coordinates of these keypoints, the following vectors from Table 3.2 are calculated: hip vector, thigh vector, calf vector, and lower spine vector. An example of 2D and 3D visualizations of these keypoints and vectors is shown in Figure 3.12. The same as for the arm angles, for every position of a joint, two angles are needed. For the leg limb, there are three joints, the hip, knee, and ankle joint. The position of the hip is set by the body proportions of the infant or the robot configuration, so only the estimation of the position of the knee and ankle joints is needed. To estimate the position of the knee and the direction of a thigh vector, the hip pitch and hip roll must be calculated. Then, when estimating the position of an

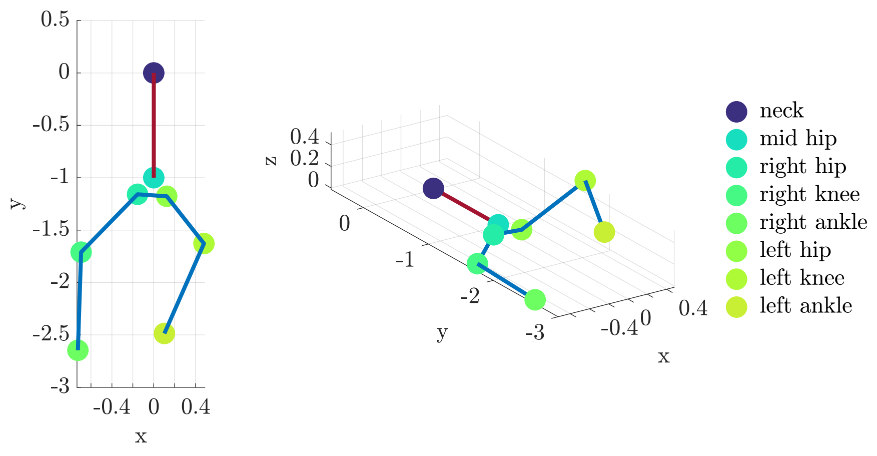


Figure 3.12: Lower body keypoints in 2D and 3D.

ankle and the direction of a calf vector, two additional angles must be calculated: hip yaw and knee angle.

Hip pitch and hip roll

Hip pitch is a front-back movement when the principal axis of the leg is aligned with gravity. The leg moves in a plane given by the hip vector $\mathbf{v_{hp}}$. Let us consider the lower spine vector as representing the gravity vector; therefore, the reference vector. Our goal is to rotate the lower spine vector to the same direction as the thigh vector. First, the projection of the thigh vector to the plane defined by the hip vector is calculated. Subsequently, the lower spine vector is also projected onto the same plane, and the angle between the two projected vectors is calculated using Equation 3.14. The calculated angle is labeled as hip pitch β_{pitch}. A visual representation of this process can be found in Figure 3.13.

A hip roll is a lateral movement when the leg is aligned with gravity. The same procedure as that used to estimate hip pitch could be used, with the difference of considering the plane given by the cross product of the spine and the hip vectors as the movement plane. However, better results appear to be obtained when the hip roll is calculated as follows. The projection of the thigh vector to the plane given by the hip vector is calculated as in the previous paragraph, and then the angle between the thigh vector and this projection is estimated and labeled hip roll β_{roll}. The visual representation of this process can be found in Figure 3.13.

Hip yaw and knee angle

Once the position of the knee joint is matched with the input data, the position of the ankle joint needs to be estimated. For that matter, knee angle and hip yaw angles need to be found. The knee angle has fairly simple calculations. The angle of the knee is the angle between the thigh vector and the calf vector. It is calculated similarly to the elbow angle

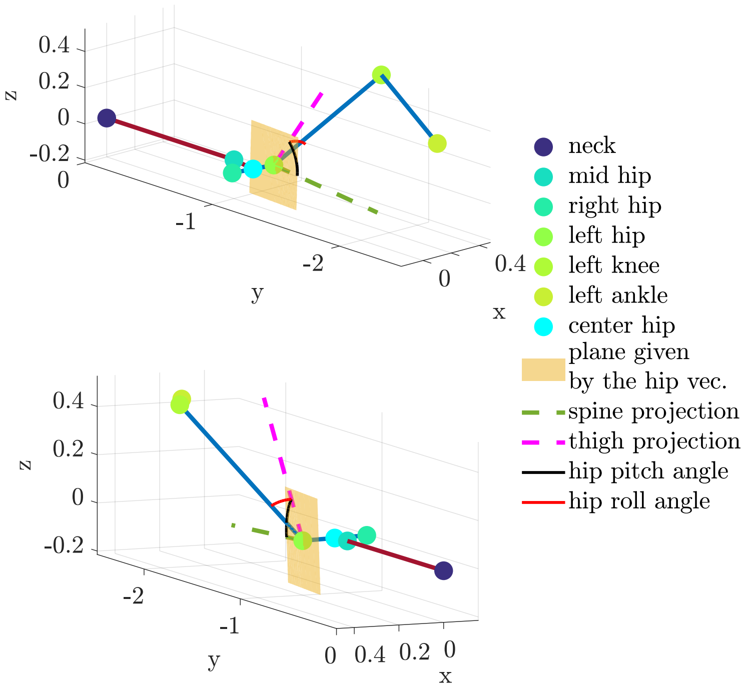


Figure 3.13: Calculation of hip pitch (top) and roll (bottom) angles.

in Subsection 3.4.1.

Hip yaw is the rotational movement of the thigh along its principal axis. We use forward kinematics to calculate the hip yaw angle, which is the same solution as for the shoulder yaw. The position of a knee and ankle is calculated by applying forward kinematics with the hip pitch and roll calculated before. The process of applying forward kinematics begins with the following equations:

$$\mathbf{B} = \mathbf{R}_{\text{ot2}}\mathbf{R}_{\text{ot1}}\mathbf{R}_{\mathbf{x}}\left(\frac{\pi}{2}\right) \qquad \text{for the left leg,} \tag{3.29}$$

$$\mathbf{B} = \mathbf{R}_{\text{ot2}}\mathbf{R}_{\text{ot1}}\mathbf{R}_{\mathbf{x}}\left(-\frac{\pi}{2}\right) \qquad \text{for the right leg,} \tag{3.30}$$

where $\mathbf{R}_{\text{ot1}}$ denotes the rotation matrix gained from the axis-angle rotation of $\mathbf{y}$ vector $\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$, where the axis of rotation is cross product of $\mathbf{y}$ vector and hip vector $\mathbf{v}_{\text{hp}}$. The angle of rotation is determined by the angle between the $\mathbf{y}$ vector and hip vector $\mathbf{v}_{\text{hp}}$. The $\mathbf{R}_{\text{ot2}}$ is also rotation matrix gained from the axis-angle rotation, this time the two vectors for determining the angle and axis of rotation are x base vector of $\mathbf{R}_{\text{ot1}}$—first column, first three rows—and the low spine vector $\mathbf{v}_{\text{lsp}}$. The multiplication of these two matrices allow us to align the base correctly—z axis almost aligned with the hip vector, x axis aligned with the low spine vector—as shown in Figure 3.14.

After the base frame is set, hip pitch and hip roll are applied to rotations and the whole base frame is then translated by the norm of thigh vector into the found knee joint by

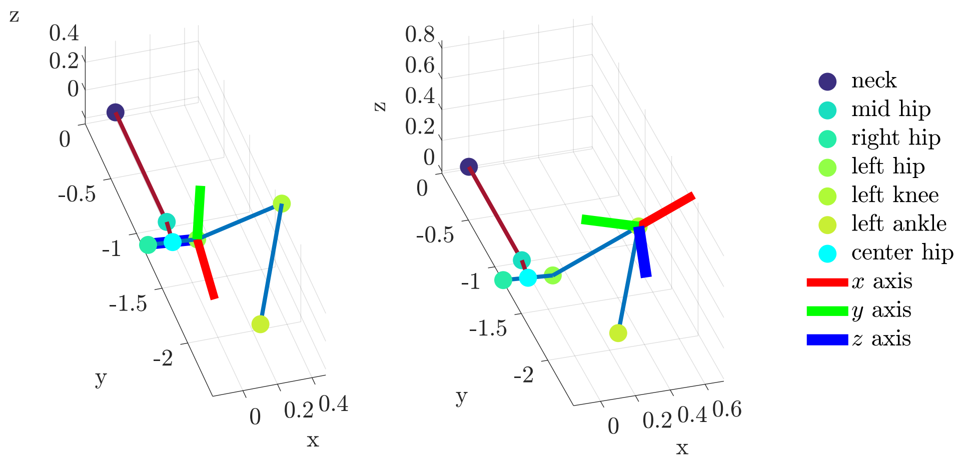


Figure 3.14: Steps while calculating the hip yaw. Setting the base frame on the left. Applying pitch and roll on the right.

following equations:

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(\beta_{\text{pitch}})\mathbf{R_x}\left(\frac{\pi}{2}\right)\mathbf{R_z}(-\beta_{\text{roll}})\mathbf{T_x}(\|\mathbf{v}_{\text{th}}\|) \qquad \text{for the left leg,} \tag{3.31}$$

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(-\beta_{\text{pitch}})\mathbf{R_x}\left(\frac{\pi}{2}\right)\mathbf{R_z}(-\beta_{\text{roll}})\mathbf{T_x}(\|\mathbf{v}_{\text{th}}\|) \qquad \text{for the right leg.} \tag{3.32}$$

Finally, the knee angle is applied and then a translation with the norm of calf vector is performed:

$$\mathbf{B} = \mathbf{B}\mathbf{R_z}(\beta_{\text{knee}})\mathbf{T_x}(\|\mathbf{v}_{\text{clf}}\|) \tag{3.33}$$

this leads to no-yaw ankle joint, its coordinates can be found as the origin of the coordinate system represented by $\mathbf{B}$. With that joint, the no-yaw calf vector is calculated and projected onto the plane defined by the x axis of the coordinates frame after applying roll and pitch. The input calf vector is also projected onto that plane, the angle between those two projected vectors is then calculated by Equation 3.14 and is denoted as hip yaw β_{yaw} as shown in Figure 3.15.

3.4.3 Torso joint angles

The movement of a child is not complete without the movement of a torso. For estimating torso joint angles, the following keypoints are needed: neck, spine, right and left hip, right and left shoulder. From these keypoints, the following vectors from Table 3.2 are extracted: shoulder vector, hip vector, low spine vector, spine vector. Our simplified body skeleton works in the way that the hips are fixated to the one position, therefore, the hip vector will be used as the reference in the following paragraphs. We consider a spherical joint in the torso, located just above the hip, which can be moved by the torso pitch, roll, and yaw defined in Section 3.3.

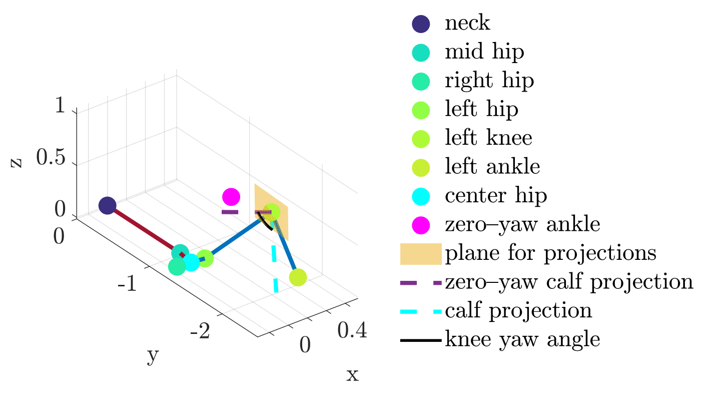


Figure 3.15: Calculation of the knee yaw angle.

In order to calculate the torso pitch angle, first, the low spine vector is projected to the plane given by the hip vector. Then the spine vector is projected onto that plane as well, and the angle between these two projected vectors is calculated using Equation 3.14 and denoted as the torso pitch γ_{pitch}. The visual representation of this process can be seen in Figure 3.16.

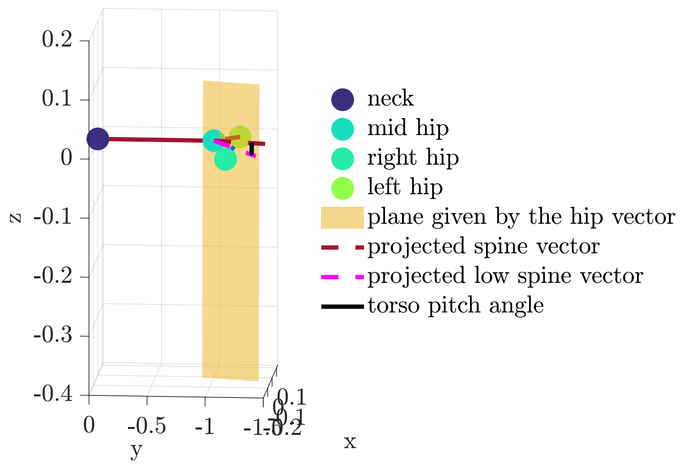


Figure 3.16: Calculation of torso pitch angle.

The estimation of the torso roll angle is fairly the same as the estimation of the torso pitch angle. The only difference is in the plane onto which the vectors are projected. The plane normal for the torso roll angle is constructed as a cross-product of the hip vector and the spine vector. Then the low-spine and spine vectors are projected, and the angle between them is calculated, creating a torso roll angle.

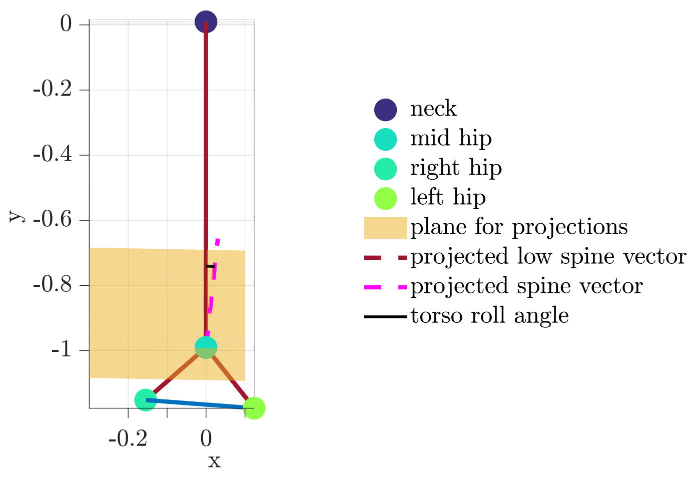


Figure 3.17: Calculation of torso roll angle.

To calculate the torso yaw angle, the plane defined by the spine vector is needed. The hip vector and the shoulder vector are then projected onto this plane and the angle between these two projected vectors is calculated and labeled torso yaw γ_{yaw}, as shown in Figure 3.18.

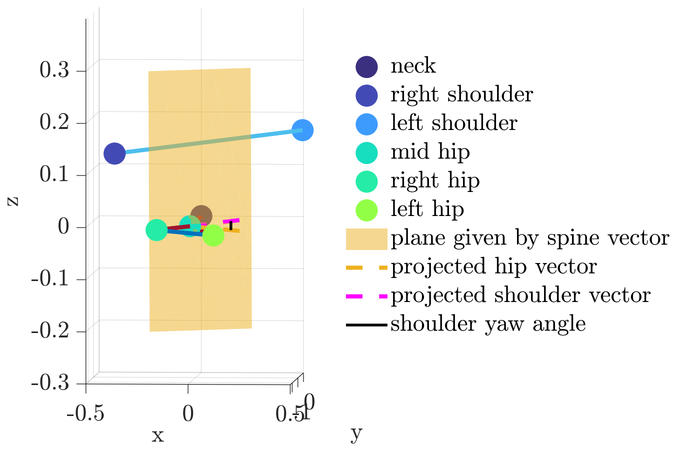


Figure 3.18: Calculation of torso yaw angle.

3.4.4 Neck joint angles

In this work, one of our objectives is to explore the infant's perspective by simulating the opening of the eyes. To achieve a more accurate viewpoint, the neck joint angles play an important role in the overall body movement. The keypoint locations involved in extracting the angles include the neck, spine, left and right eyes, and the nose vertex. From the selected keypoints, the following vectors from Table 3.2 are extracted: shoulder vector, spine vector, eyes vector, neck-to-left-eye vector, neck-to-right-eye vector.

However, it is worth mentioning that estimating the head pose accurately requires a larger number of keypoints (up to 68, as discussed in Chapter 2). Our approach, although not computationally demanding, provides only the approximate values for the neck joint angles. The neck joint is a spherical joint that can be divided into three revolute joints, neck roll, pitch, and yaw. Since the neck is connected to the torso and mimics its movements, the angles are calculated with respect to the torso vectors.

The neck roll refers to the right-to-left tilting movement of the head. This movement is performed in the plane defined by the cross-product of the spine and the shoulder vectors. Similar to the torso roll, which is calculated in the plane defined by the cross-product of the spine and hip vectors. In this case, the reference vector is the spine vector, which is already contained within the plane, eliminating the need for projection. Then a new vector is extracted with the direction from the neck to the point in the middle of the ears:

$$\mathbf{v} = \frac{k_{\text{rer}} + k_{\text{ler}}}{2} - k_{\text{nck}}. \tag{3.34}$$

This vector is projected onto the plane, and the angle between the projection and the reference vector is calculated using Equation 3.14 representing the neck roll angle δ_{roll} as shown in Figure 3.19.

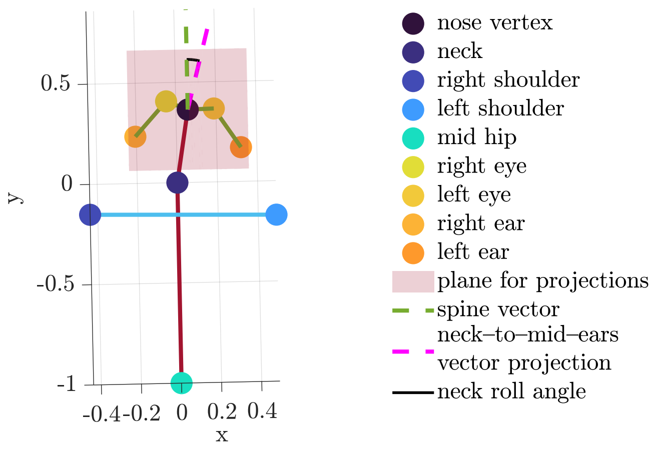


Figure 3.19: Calculation of neck roll.

The neck yaw corresponds to the head rotation. The movement is performed in the plane defined by the spine vector. To calculate the neck yaw angle, the projection of a shoulder vector and eye vector onto that plane is performed. The angle between these two projections is then calculated and labeled as the neck yaw angle δ_{yaw} as shown in Figure 3.20.

The calculation of neck pitch angle is more challenging compared to the previous head angles. This is because there are no detected keypoints on the head, that would establish a plane, enabling us to determine a defined angle between the plane and the spine vector

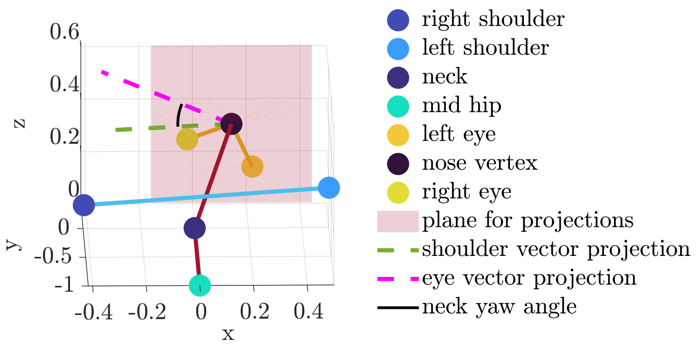


Figure 3.20: Calculation of neck yaw.

when the neck pitch is set to zero. In other words, there is no reference vector available for calculating the pitch angle. However, we managed to construct a plane using two vectors—neck to left eye vector and neck to right eye vector. By taking the cross product of these two vectors, we obtain the normal vector of the plane. The angle between this normal vector and the spine vector is calculated, resulting in a relative angle δ_{relative} as shown in Figure 3.21.

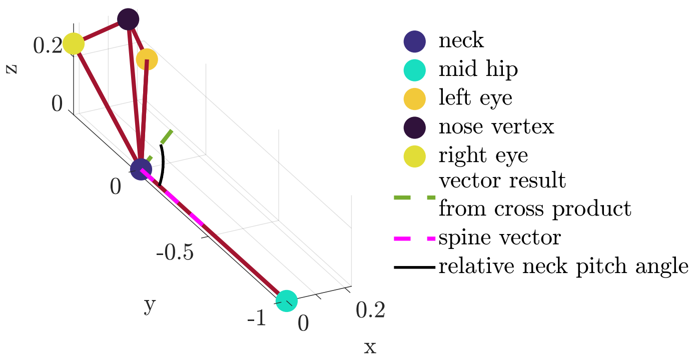


Figure 3.21: Calculation of neck pitch.

After estimating the relative neck pitch angle, the reference needs to be found. Initially, we calculated the relative neck pitch angle across all frames in the BBHMR dataset and set the mean value as the reference. However, this approach resulted in infants tilting their heads backward excessively in many sequences, which appeared unnatural. Therefore, we decided to calculate the joint angle by processing data from anthropometric measurements of infants which were introduced in Subsection 3.5 in Table 3.3. This analysis resulted in a reference angle of 81 degrees. Consequently, the neck pitch can now be computed using the following formula:

$$\delta_{\text{pitch}} = \delta_{\text{relative}} - 81. \tag{3.35}$$

3.5 Infant dummy figure

With the assumptions and definitions of the types of joint angles, their respective movements, and the positive and negative directions of movement made in Subsection 3.3, We have constructed a simplified body skeleton as an infant model defined in Unified Robot Description Format (URDF) as shown in Figure 3.22. The infant dummy figure is directly controlled by the joint angles and can be visualized in a Matlab environment. Since URDF

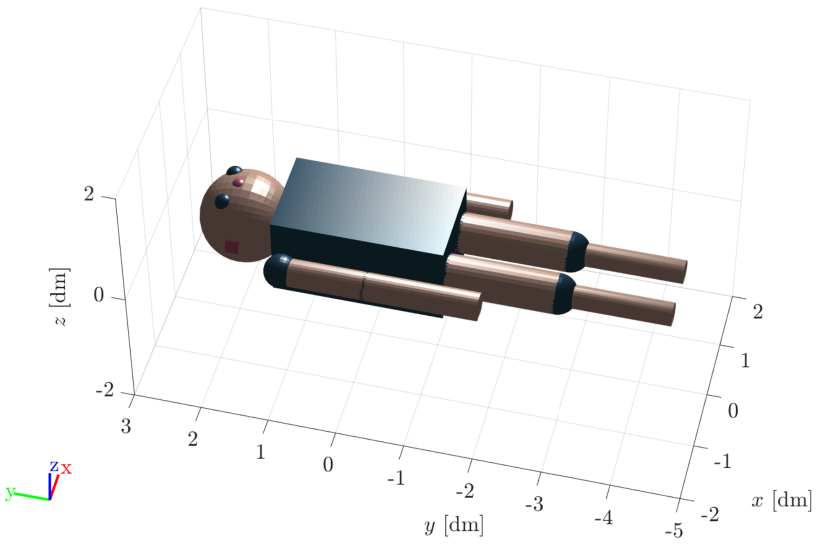


Figure 3.22: Infant dummy figure based on anthropometric measurements

is generally recognized Extensible Markup Language (XML) specification to describe a robot or a model for simulator, it makes it easier to transfer the infant model to any robotic simulation engine. The whole URDF specification usually covers kinematic and dynamic chains, visual representation, and collision model [31]. However, for our purposes, only the kinematic chain and visual representation were important and therefore implemented.

The respective body lengths of the dummy figure are based on real infant body measurements obtained from the Anthrokids dataset [32], which contains anthropometric measurements of children and infants of various ages. The infant dummy figure is based on average body lengths for infants aged 4 months on average, taking into account both males and females. However, head measurements for infants of that specific age were not available, therefore the youngest age available was taken for the head proportions. The body lengths used for the construction of the infant model are shown in Table 3.3.

The infant dummy figure serves as a way to visualize the joint angles calculated in

body measurement	length [dm]	body measurement	length [dm]
shoulder breadth	1.87	head width	1.14
shoulder-elbow distance	1.23	frontal breadth	0.83
elbow-wrist dist.	1.66	head breadth	1.24
crown-rump dist.	4.23	head height	1.73
mid-thigh depth	0.59	tragion to top of head	8.30
hip breadth	1.43	mid thigh radius	0.30
hip-knee dist.	1.59	ankle radius	0.17
knee-ankle dist.	1.65	upper arm radius	0.20
face height	1.42	forearm radius	0.20
nose length	0.29	crown hip dist.	3.94
lower face height	0.81		

Table 3.3: Selected important body measurements of infants aged 3—5 months obtained from [32].

Section 3.4. It provides a quick and convenient assessment of the retargeted motion, without the need for integration into any additional platforms. An example of two reconstructed body poses with the use of calculated joint angles is shown in Figure 5.1.

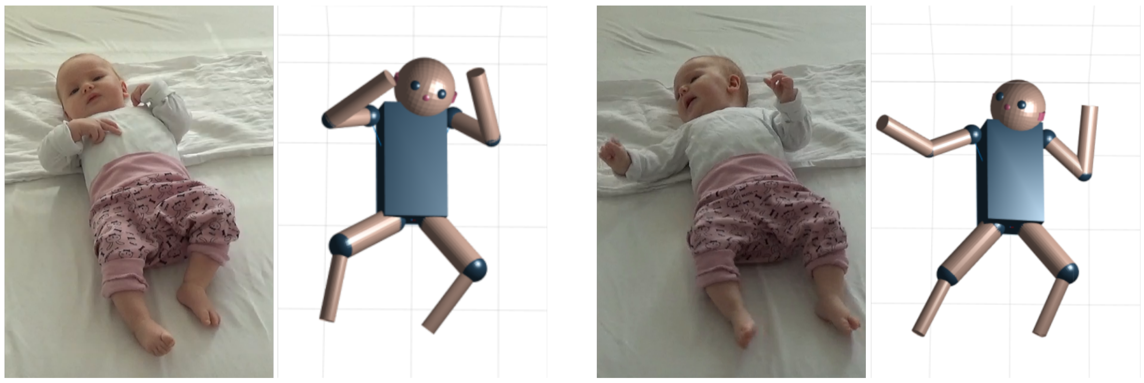


Figure 3.23: Transferred motion from baby to infant dummy figure.

Chapter 4

Target Platforms

In this work, the target platforms for transferring infant movements using joint angles are A Multi–Modal Infant Model (MIMo) [33], Fetus Simulator [34], and the iCub robot simulator [35]. MIMo and Fetus Simulator are running on the MuJoCo physics engine [36], specifically, its Python version, and are based on real infants or children and their corresponding limb lengths. Their main purpose is to study the development of children and infants in the early years of their lives. The iCub Simulator runs on a Gazebo physics engine [37], it is modeled after the real iCub robot [38] and aims to simulate any activity one wants to do on the iCub robot, before transferring it to the physical version.

This chapter aims to provide a detailed description of each platform, with a focus on its kinematic structure and relevant joint angles that are important for retargeting infant movements. Additionally, we will discuss the limitations that should be taken into account when using each simulator for this purpose. These may include factors such as model inaccuracies or variations in rotational axis directions. For each platform, the Range of motion (ROM) is listed, however, in the end, we had to disable the limits due to various reasons described in Chapter 5.

4.1 iCub simulator

Unlike the Fetus Simulator and MIMo, which are modeled after infants and young children, the iCub Simulator [35] is a simulator based on an actual physical robot called the iCub. It is used inside the Gazebo physics engine and utilized to test the algorithms before deploying them to a physical robot. The iCub robot was designed to evaluate the neural capabilities of a 4-year-old human child and to enable researchers to study fields of autonomous exploration and social interaction [38].

Additionally, the iCub robot is also equipped with two basic human senses: touch and vision. Touch is integrated via a tactile sensor network that is organized into patches and distributed among the robot's body parts, including the left and right hands, the forearm and arm, upper and lower legs, and the torso. However, this integration is not reflected in the simulator. The iCub's vision is provided by two RGB cameras integrated into its eyes both on the real robot and in the simulation [39].

Kinematics

The iCub is equipped with a total of 53 DoF, the important ones are shown in Figure 4.1, to accurately mimic human motion [39], providing us with enough DoFs to effectively retarget infant motion. While the joints are modeled with pitch, roll, and yaw angles, the center of rotation cannot be set in the same location. This is because the motor actuators controlling the joints cannot be placed in the same spot. For example, in the human shoulder joint, the center of rotation for shoulder flexion and abduction is nearly the same point, but placing two motors in one location is not possible. That means that the joints are modeled as revolute, not spherical, unlike in the MIMo (Section 4.2) or the Fetus Simulator (Section 4.3).

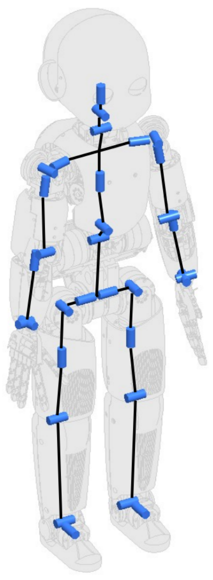


Figure 4.1: The kinematic structure of iCub. Image obtained from [40].

Both the simulator and the real iCub robot implement joint angle limits. During development, the range of motion of a human was taken into account, as well as the restrictions given by the physical implementation of the robot's tendons and motors. The resulting joint angle limits are shown in Tab. 4.1 and are provided by the official documentation of iCub [39].

The iCub's kinematics is then described by the Denavit–Hartenberg (DH) convention. This convention involves using four parameters denoted $a_i, d_i, \alpha_i, \theta_i$ to describe every link i in the robot structure. The DH-parameters are used to calculate the position and orientation of each link relative to the previous one. The final position and orientation of the end effector (wrist or ankle, in our case) can be determined from these values.

In Table 4.2, the DH-parameters for iCub's arms are provided. The first three links correspond to the torso's pitch, roll, and yaw, while the remaining links represent shoulder pitch, roll, yaw, elbow angle, and then finally, a joint that we consider the wrist. Table 4.3 provides the DH-parameters for iCub's legs, where the first three links correspond to hip

Joint	ROM [°]
Neck flexion/ext.	-30 to 22
Neck lateral flex.	-20 to 20
Neck rotation	-45 to 45
Torso flexion/ext.	-20 to 70
Torso lateral flex.	-30 to 30
Torso rotation	-50 to 50
Shoulder pitch	-95.5 to 8
Shoulder roll	-0 to 160
Shoulder yaw	-32 to 80
Elbow flexion/ext.	15 to 106
Hip pitch	-30 to 92
Hip roll	-15 to 92
Hip yaw	-72 to 72
Knee	-100 to 0

Table 4.1: Joint range of motion of iCub obtained from [39].

Link (i)	a_i [mm]	d_i [mm]	α_i [°]	θ_i [°]	a_i [mm]	d_i [mm]	α_i [°]	θ_i [°]
0	32	0	90	0	32	0	90	0
1	0	-5.5	90	-90	0	-5.5	90	-90
2	23.3647	-143.3	-90	105	-23.3647	-143.3	90	-105
3	0	107.74	-90	90	0	-107.74	90	90
4	0	0	90	-90	0	0	-90	-90
5	15	152.28	-90	75	15	152.28	-90	75
6	-15	0	90	0	-15	0	90	0
7	0	137.3	90	-90	0	137.3	90	-90

(a) : Left arm **(b) :** Right arm

Table 4.2: DH-parameters describing links in iCub's arms from [39].

Link (i)	a_i [mm]	d_i [mm]	α_i [°]	θ_i [°]	a_i [mm]	d_i [mm]	α_i [°]	θ_i [°]
0	0	0	-90	90	0	0	90	90
1	0	0	-90	90	0	0	90	90
2	-0.9175	-234.545	90	-90	-0.9175	234.545	-90	-90
3	-200.5	0	180	90	-200.5	0	180	90
4	0	0	-90	0	0	0	90	0

(a) : Left leg **(b) :** Right leg

Table 4.3: DH-parameters describing links in iCub's legs from [39].

pitch, roll, and yaw, followed by knee angle, and link 4 represents joint that we consider as ankle. Finally, Table 4.4 gives the DH-parameters for the iCub's head, where the links represent the pitch of the torso, the roll, the yaw, and then the pitch of the neck, the roll, and the yaw.

From the tables mentioned above, it is evident that not all joints of the iCub can be treated as spherical; rather, they are revolute. For example, the torso pitch and roll have

Link (i)	a_i [mm]	d_i [mm]	α_i [°]	θ_i [°]
0	32	0	90	0
1	0	-5.5	90	-90
2	0	-223.3	-90	-90
3	9.5	0	90	90
4	0	0	-90	-90
5	-50.9	82.05	-90	90

Table 4.4: DH-parameters describing links in iCub's head from [39].

different centers of rotation. However, the shoulder pitch and roll and the hip pitch and roll are located at the same place due to motor placement. The goal was to model the shoulder and hip joints as "quasi"-spherical [40]. An example of this can be seen in Figure 4.2, where the shoulder pitch and roll motor actuators on the iCub's arm are placed in such a way that their axes cross in the middle.

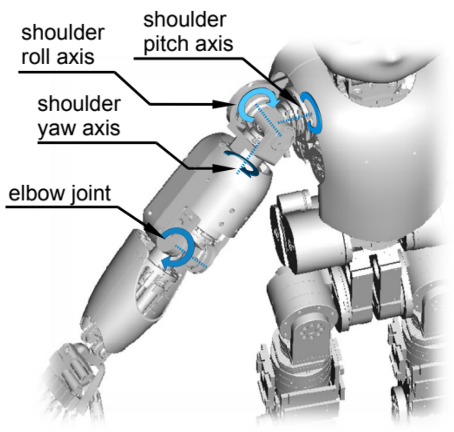


Figure 4.2: Joint axes on iCub's arm [40].

For other constraints, in Table 4.2, there is a non-trivial offset $\theta_5 = 75$ degrees between the shoulder yaw and elbow angle joints, which can affect the final results, as we are considering only trivial offsets as multiples of 90 degrees. Additionally, the shoulder pitch joint does not function as a shoulder flexion representation defined in Section 3.3, as it does not raise the arm directly in front of the shoulder joint, but rather in front of the torso; the tilted orientation of the shoulder pitch joint can be seen in Figure 4.1.

4.2 MIMo: A Multi-Modal Infant Model

MIMo is a simulator developed to study cognitive development in humans and Artificial Intelligence (AI). It is based on the Mujoco physics engine which is used for simulation in robotics and biomechanics and also used for RL Methods [36]. The code for MIMo is written in Python and constructed as an OpenAI gym to ensure an easy process of integrating learning methods and algorithms [33].

MIMo is provided with multiple sensing modalities, including proprioception, vestibular system, binocular vision, and touch perception. The last two allow us to extract sensory information related to touch and vision and add those to our motion retargeting process, simulating what the infants are seeing and when and where the infants are performing self-contact (also called self-touch) events, which are believed to be necessary for them to learn more about their bodies. The binocular vision is implemented via two RGB cameras, obtaining a 60° field of view, situated on the MIMo's eyeballs. The touch is implemented by sensors spread over the whole body, and then when a touch occurs, the forces generated by the contact are distributed among the nearest sensors on the MIMo's body [33]. In Figure 4.3 there is a MIMo model and in Figure 4.4 there is an example of MIMo's binocular vision and touch perception. In Figure 4.4b, there is a visualization of touch perception with touch forces. However, in our implementation, we had to limit the output of the sensor to *touched* or *not touched*, because of the limitations discussed in Section 5.2.2.

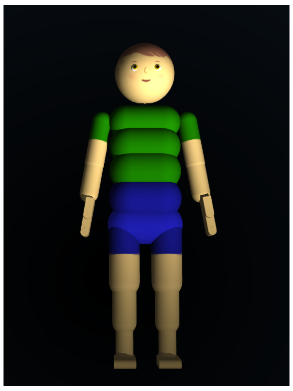
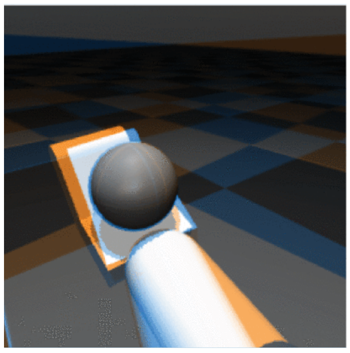
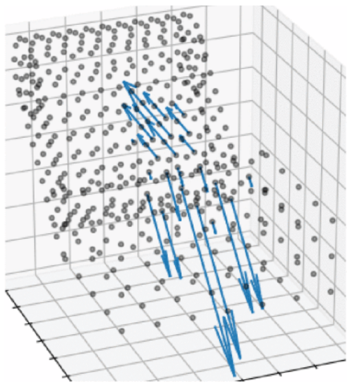


Figure 4.3: MIMo model.



(a) Binocular vision.



(b) Touch perception.

Figure 4.4: Example of multimodal perception while holding a ball. Images from [33].

Kinematics

For the purpose of improving the realistic factor of simulation, MIMo's body lengths were extracted from anthropometric measurements of 16-19 months old infants, allowing our motion transfer to be more accurate than when using platforms with characters' limb lengths based on adult humans. MIMo's joints are all modeled as series of 1-DOF hinge joints representing three movements—flexion/extension, abduction/adduction, and internal/external rotation, which is typical for human representation as was stated in Section 3.3. The only joint not modeled as stated is the shoulder joint. For better simulation purposes, and to apply joint angle limits, the three original movements were transformed into abduction/adduction movement, horizontal flexion, and internal/external rotation [33]. For our work, that means the task to implement calculating these two additional joint angles. This modification is not specific to a single target platform but is also applicable to the Fetus Simulator 4.3.

In the MIMo simulator, extension, adduction, and internal rotation are treated as positive, whereas flexion, abduction, and external rotation are treated as negative [33]. In our work, the movements are treated almost the same way, with the exception of flexion and extension movement, where we treat flexion as positive and extension as negative. When implementing the joint angle values, the treatment of flexion and extension, respectively, must be considered.

Joint angle limits are also implemented as shown in Table 4.5. They were obtained from multiple sources since there is not a single source containing a full set of range of motion of 18 months old infants, which was the target when constructing the MIMo simulation. When the data for this age were not present, the value for that ROM was obtained from the youngest age possible to find [33].

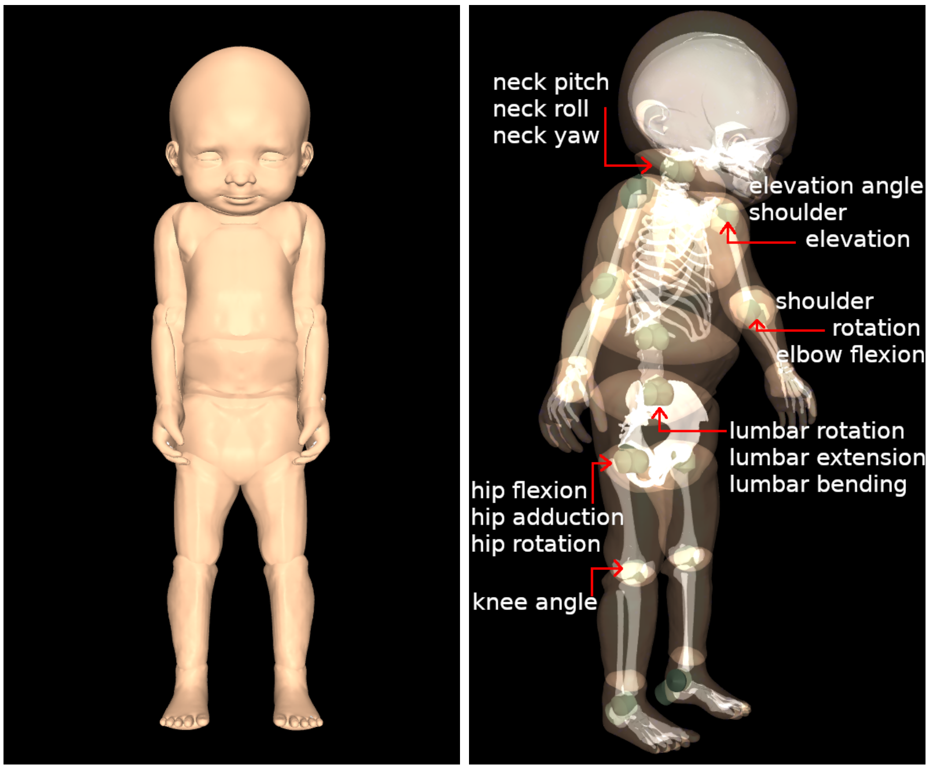
Joint	**ROM [degrees]**
Neck flexion/ext. (pitch)	-70 to 80
Neck lateral flex. (roll)	-70 to 70
Neck rotation (yaw)	-111 to 111
Torso flexion/ext. (pitch)	-61 to 34
Torso lateral flex. (roll)	-41 to 41
Torso rotation (yaw)	-36 to 36
Shoulder horizontal	-118 to 28
Shoulder flexion/ext.	-183 to 84
Shoulder rotation (yaw)	-99 to 67
Elbow flexion/ext.	-146 to 5
Hip flexion/ext. (pitch)	-133 to 20
Hip ab-/adduction (roll)	-51 to 17
Hip rotation (yaw)	-32 to 41
Knee flexion/ext.	-145 to 4

Table 4.5: Joint range of motion of MIMo obtained from [33].

4.3 Fetus Simulator

Fetus Simulator was originally developed to study the sensorimotor development of humans in the early stages of development and includes a realistic musculoskeletal model of a human fetus in a soft urine environment as described by Kim *et al.* in [34]. However, we will not be using the soft urine environment for our work, as we aim to reproduce the movements of already-born children. The body proportions of this model are taken from the measurements of 22 real infants with a mean of 1.36 months of age. The body proportions together with the model's dynamics of joints and implemented muscles, as well as its realistic meshes, provide us with an excellent target platform.

The model is based on the MuJoCo physical engine, which, as mentioned before, is used in RL applications. Both the skeleton and muscle structure of the model are based on the OpenSim model, which allows biological coordination, which means that some bones, *e.g.*, patella move cooperatively with a certain rhythm based on the angle of the corresponding joint, *e.g.*, knee. As cooperative coordination is not implemented in MuJoCo, in the Fetus simulator there are parts called coupling joints that allow the fetus's skeleton to have realistic movements of its bones [34].



(a) Non-transparent as will be used for retargeting. **(b)** Transparent with joint actuators.

Figure 4.5: Visual representation of fetus simulator.

Kinematics

The fetus' body is put together from 45 rigid body parts, altogether providing us with 15 joints and 31 DoF [34]. From Figure 4.5b, it is evident, that shoulder, hip, neck and torso joints are all modeled as “quasi“-spherical (3 revolute joints in one place, creating the spherical joint), which is consistent with our modeling approach. Figure 4.5 provides a visual representation of the fetus, including a transparent view highlighting the skeleton and the relevant joint actuators for our work.

Fig. 4.5b shows joint actuators that are not labeled, and these will not be used for our purposes. The torso has two actuators for each movement; lower and upper; however, only the lower actuators will be used since we only have two input points for the torso (neck and then mid-hip point as shown in Table 3.1).

In Figure 4.5b it is evident that the hip joint can be considered spherical, following the previously introduced pitch, roll, and yaw representation convention for the typical movement of the joints. The torso joint can also be considered spherical, rotating along all three main axes. As for the shoulder joint, the position of the shoulder rotation joint actuator does not affect its movement, making it spherical. However, instead of using the standard pitch, roll, and yaw angles representing flexion, abduction, and rotation, it uses elevation angle and shoulder elevation, which is similar to the MIMo's representation of shoulder joint movement described in Section 4.2, where elevation angle corresponds to shoulder horizontal movement and shoulder elevation corresponds to shoulder flexion. However, there is one significant difference between these two platforms' implementation of shoulder joint. MIMo's shoulder horizontal movement does not influence the position of the end effector (wrist), but the Fetus simulator's elevation angle affects it in the negative direction. This needs to be considered when implementing the joint angle control—the elevation angle value needs to be added to the shoulder yaw angle value.

Table 4.6 shows the joint angle limits implemented, which are also present in the Fetus Simulator. The limits for torso joint angle can be extended by utilizing the higher torso actuators shown in Figure 4.5b. As the simulator models a child younger than MIMo, the joint ranges are smaller than those of MIMo.

Joint	**ROM** [°]
Neck flexion/ext. (pitch)	-44 to 44
Neck lateral flex. (roll)	-50 to 50
Neck rotation (yaw)	-89 to 89
Torso flexion/ext. (pitch)	-30 to 10
Torso lateral flex. (roll)	-5 to 5
Torso rotation (yaw)	-30 to 30
Shoulder horizontal	-90 to 130
Shoulder flexion/ext.	0 to 90
Shoulder rotation (yaw)	-90 to 20
Elbow flexion/ext.	0 to 130
Hip flexion/ext. (pitch)	0 to 142
Hip ab-/adduction (roll)	-73 to 18
Hip rotation (yaw)	-55 to 18
Knee flexion/ext.	0 to 144

Table 4.6: Joint range of motion of Fetus Simulator.

Chapter 5

Experiments and Results

5.1 Process of evaluation

Before we proceed to transferring the joint angles extracted in Chapter 3 to target platforms presented in Chapter 4 we will describe the evaluation process that will be used later. The main objective of this work is to accurately retarget the direction of body segments or vectors. Our focus is not on the precise location of the end effector, as our goal is to simulate the motion and sensory perspective of infants of different ages. For instance, if an infant is looking at their elbow, our aim is to replicate that perspective. However, if we were to prioritize the exact position of the end effector, it could result in a different arm configuration, leading to a different elbow position and ultimately deviating from replicating the infant's viewpoint.

5.1.1 Scaling and normalizing body lengths for joint comparison

Due to the variations in body lengths between infants of different ages and the lengths of the body parts of simulated robots or characters from the target platforms, a scaling process is necessary to make valid comparisons between the input data and the data from the target platforms. We have adopted the same process used to create the dataset BBHMR, which involves rescaling and rotating the data to establish a reference system with respect to two keypoints.

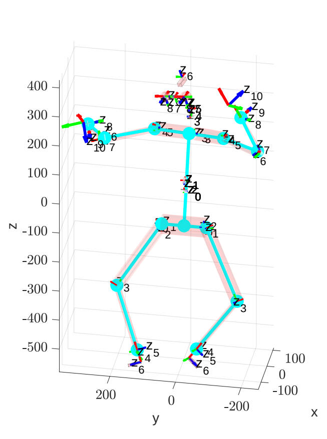
The first step is to determine the point right in the middle of the shoulders, which is calculated as:

$$k_{\text{middle-shoulder}} = \frac{k_{\text{lsh}} + k_{\text{rsh}}}{2}. \tag{5.1}$$

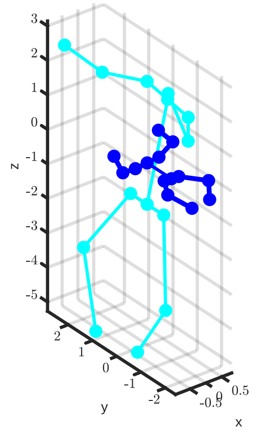
This point is set to be the origin of the new coordinate system. The second point is located in the middle of the two hip joints:

$$k_{\text{middle-hip}} = \frac{k_{\text{lh}} + k_{\text{rh}}}{2}. \tag{5.2}$$

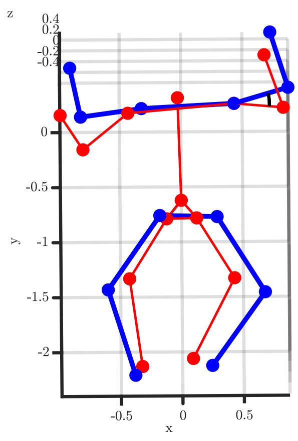
This point is then positioned at $p = [0, -1, 0]$, aligning the line connecting the two points with the y axis. By establishing this reference system, we obtain a translation matrix $\mathbf{T}$ and a rotation matrix $\mathbf{R}$, which are applied to all other keypoints to align them with the reference system created as shown in Figure 5.1b. This process normalizes the body lengths based on the middle-shoulder-middle-hip ratio, enabling more accurate and normalized comparison of joint positions and body vectors across different platforms.



(a) Mapping the joints of the target platform (in this case of the iCub robot).



(b) Rescaling the extracted joints (cyan) to the middle-shoulder-middle-hip ratio (blue).



(c) Calculating the error metric (black) between the input data (red) and the scaled data (blue).

Figure 5.1: Canonization process and the error metric calculation.

5.1.2 Metric

To assess the accuracy of the motion transfer process, we utilize a metric based on error angles. First, we extract the directions of important body vectors. Next, we calculate the angle between the reference body vector, derived from the keypoints obtained from the dataset (Section 3.1), and the body vector obtained by transferring the joint angles to the target platforms. This angle is computed using the following equation:

$$\phi = \arccos \frac{\mathbf{r} \cdot \mathbf{t}}{||\mathbf{r}||\,||\mathbf{t}||}, \tag{5.3}$$

Here, ϕ represents the angle, $\mathbf{r}$ represents the reference body vector and $\mathbf{t}$ represents the body vector obtained from the target platforms. The angle ϕ indicates the error in the direction of the vector. An example of this process is shown in Figure 5.1c.

To evaluate the overall accuracy, we calculate the **MAE** metric. the MAE is calculated as the median of all error angles, representing the central tendency of the errors. It is the value ϕ_{m} that lies at the midpoint of all the error angles such that there is an equal probability of falling above or below it.

In addition to MAE, we also calculate the mean value of the error angles using Equation 5.4. This provides a measure of the average error, but the MAE is preferred as it is less sensitive to outliers. Furthermore, we compute the standard deviation using Equation 5.5 to measure the dispersion of the data in relation to the mean.

$$\overline{\phi} = \frac{1}{n}(\phi_1 + \phi_2 + ... + \phi_n) = \frac{1}{n}\sum_{i=1}^{n}\phi_i, \tag{5.4}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} \left|\phi_i - \overline{\phi}\right|^2}{n}}. \tag{5.5}$$

By using this error metric, we can evaluate the accuracy of the motion transfer process and assess the deviations between the reference body vectors and the transferred vectors on the target platforms.

5.2 Implementation to target platforms

In the following paragraphs, the process of implementation on the target platforms will be described. First, we will present the joint angle trajectories and their smoothing. Then we will focus on each target platform defined in Chapter 4: the iCub robot, the MIMo simulator and the Fetus simulator. For each of the simulators, the control of joints will be described, modifications of joint angle values, regarding their values or positive value of their directions, will be described and, the metrics presented in Section 5.1 will be calculated and discussed. Furthermore, for each simulator, there will be shown retargeted body poses. For the videos of retargeted motion of the robot together with the input and sensory outputs refer to the dedicated Gitlab repository [19].

5.2.1 Joint angles trajectories filtering and smoothing

Once the joint angles were calculated using the procedure explained in Chapter 3 for each frame, the values were concatenated to form the joint angle trajectory. However, it was observed that the resulting joint angle trajectories sometimes exhibited a sudden peak and trough noise, as shown in Figure 5.2. The reason for the appearance of this noise is the inconsistency and variability of the input data, such as differences in body lengths between frames or significant variations in the orientation of the hips or shoulders from frame to frame. To address this issue, it is necessary to apply filtering techniques to obtain smoother movement trajectories.

Crena *et al.* [41] provided experiments to compare three filtering methods, which were Moving Average Filters, Linear Filters (for example, Butterworth filter), and Polynomial Filters. The most robust performance was given by the Butterworth zero-phase low-pass filter introduced by Winter in [42], which is a filter that is widely used in biomechanics; therefore, we decided to use that kind of filter for our work. To determine the optimal cutoff frequency for this filter, Yu *et al.* [43] presented a procedure that estimates the cutoff frequency without the need for residual analysis using only the camera frame rate that captures movement. The procedure adopted from [43] is shown below.

1. Estimate the mean optimum cutoff frequency $f_{\mathrm{c},1}$ for a given sampling frequency f_{s} using the following equation:

$$f_{\mathrm{c},1} = 0.071 f_{\mathrm{s}} - 0.00003 f_{\mathrm{s}}^2. \tag{5.6}$$

2. Filter the raw data at the estimated mean optimum cutoff frequency.

3. Calculate the relative mean residual between the filtered data x_f and raw data x_n using the following equation:

$$\epsilon = \sqrt{\frac{\sum_{n=0}^{N}(x_\mathrm{n} - x_\mathrm{f})^2}{\sum_{n=0}^{N}(x_\mathrm{n} - \overline{x})}} \times 100\%, \tag{5.7}$$

where $\overline{x}$ is the mean of x_n.

4. Estimate the final optimum cutoff frequency $f_{\mathrm{c},2}$ using the equation:

$$f_{\mathrm{c},2} = 0.06 f_\mathrm{s} - 0.000022 f_\mathrm{s}^2 + 5.95 \frac{1}{\epsilon} \tag{5.8}$$

5. Filter the raw data using the second estimate of optimum cutoff frequency to have the final filtered output.

In our implementation, we used the typical framerate of a camera that captured the input data, which was 25 FPS. By applying a fourth-order Butterworth zero-phase low-pass filter with a frequency estimated using the algorithm, we achieved less noisy results with reduced sudden peaks, as shown in Figure 5.2. The figure shows frame numbers between 1 and 100, corresponding to 4 seconds of movement at 25 FPS.

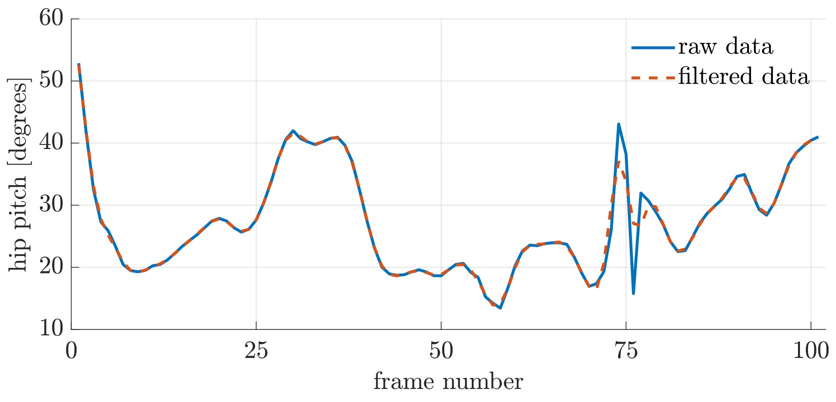


Figure 5.2: Example of filtered hip pitch angle trajectory.

5.2.2 iCub simulator

The iCub simulator, as described in Section 4.1, was chosen as the first platform to transfer the extracted joint angles. However, it is important to note that the iCub robot simulator is modeled after a 4-year-old child, which results in significant differences in body lengths compared to infants.

The iCub simulator allows direct positional joint control, meaning that the desired value of a joint angle can be set and the motor will move to that position. However, the joint limits of the iCub robot significantly restrict the ROM, as shown in Table 5.1. To overcome this limitation, we decided to deactivate the joint limits, allowing the use of the full range of motion extracted from the input data. However, this approach has a drawback: it prevents the direct transfer from the simulation to the real iCub robot without modifying or cropping the joint values to fit within the robot's physical limitations.

	left	**right**
shoulder pitch	6.87 %	14.62 %
shoulder roll	0.12 %	0.39 %
shoulder yaw	46.38 %	31.24 %
elbow	42.51 %	51.17 %
hip pitch	0.48 %	0.78 %
hip roll	3.69 %	0.63 %
hip yaw	20.69 %	28.29 %
knee	5.22 %	5.67 %

Table 5.1: Percentage the frames, where the joint limits cropped the movement when processing BBHMR dataset for selected iCub joints.

In terms of the directions of joint movements, as we defined in Section 3.3, the positive directions of movement of the iCub align with ours for most of the joints, except for knee angle, hip yaw and torso yaw, where the positive direction is opposite to ours. Examples of body poses transferred for the chosen frames are shown in Figure 5.3.

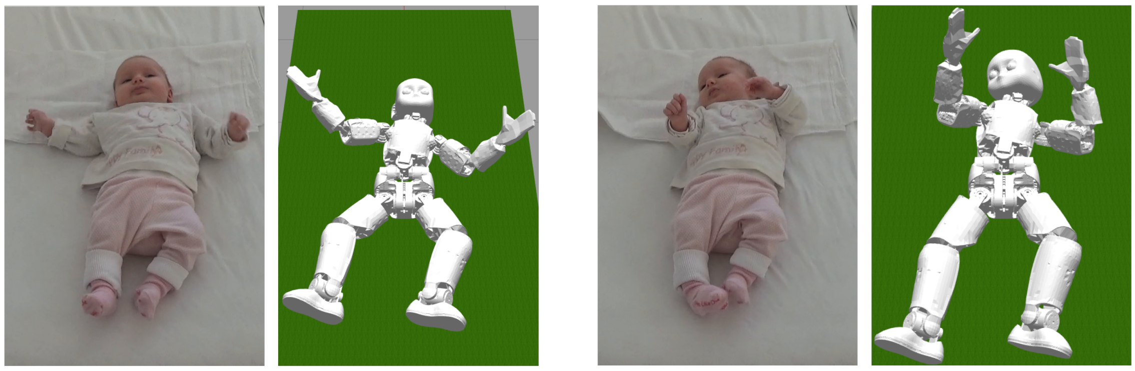


Figure 5.3: Transferred motion from baby to iCub.

As mentioned earlier, our goal was to replicate the infant's perspective by incorporating vision and touch sensors activation. However, due to the limitations in the Gazebo simulator that we used for these experiments, where the touch activation interface is not implemented, we were unable to extract the touch sensor outputs. On the other hand, we were able to obtain the visual output of two RGB cameras placed in the eyes of the iCub within the simulation, as depicted in Figure 5.4. Considering the limitations in obtaining touch sensor outputs and the desire for more realistic simulations that align with the infants' body measures, we transitioned to using the MuJoCo-based infant simulators: the MIMo simulator (Section 4.2) and the Fetus simulator (Section 4.3). These simulators provided us with

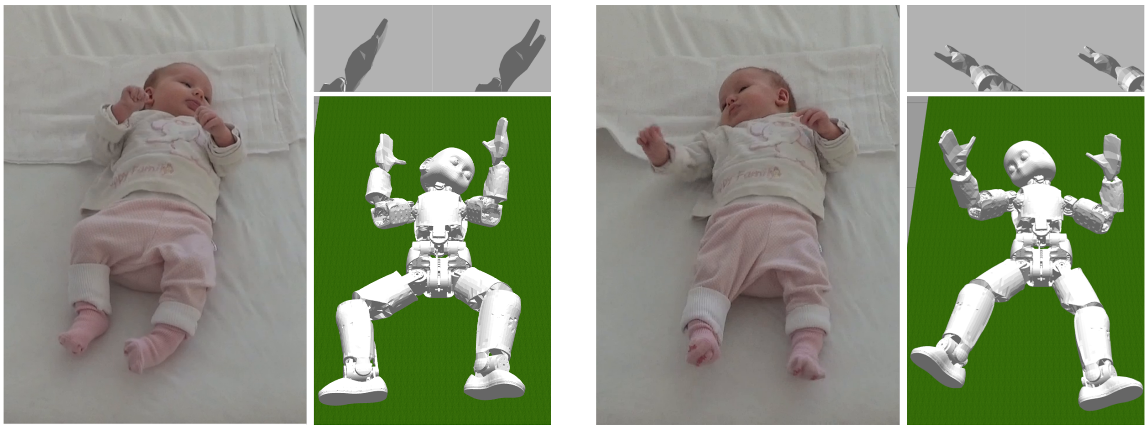


Figure 5.4: Transferred motion from baby to iCub with vision output.

improved capabilities to achieve realistic motion transfer and the potential incorporation of touch sensor activation.

Results

To calculate the position of the end effectors, specifically the wrists and ankles, iCub forward kinematics using the DH-notation was used. However, the positions of the elbow and knee joints were more difficult to obtain due to the varying distance between the motor actuators controlling their movement and the shoulder and wrist joints. The distance is not constant and changes with the yaw movement (it is denoted in Tables 4.2 and 4.3, by the parameter a). Therefore, these motors cannot be considered directly as elbow and knee joints; instead they approximate the position. From the positions of the extracted joints, the MAE metric was calculated as described in Section 5.1 and the results are shown in Table 5.2.

body vector	**angle error [degrees]**		
	mean	standard deviation	median
right upper arm	20.73	11.09	18.91
left upper arm	16.21	9.52	15.17
right forearm	13.58	11.19	10.86
left forearm	10.94	9.60	8.69
right thigh	7.54	8.61	5.12
left thigh	17.16	17.67	10.60
right calf	7.39	8.34	5.16
left calf	15.83	11.21	11.21

Table 5.2: Angle error in degrees when transferring motion to the iCub simulator.

As depicted in Table 5.2, the mean angle error values range from 7.39 to 20.73 degrees, and the median error values range from 5.12 to 18.91. The reason for these results lies in the difficulty in obtaining the positions of the elbow and knee joints, which was previously described. Furthermore, the kinematic chain of iCub slightly varies from the one defined by us. For instance, the shoulder pitch movement by our definition brings the wrist straight in front of the shoulder joint, when set at 90 degrees. However, iCub's pitch movement brings the wrist slightly in front of the torso, varying from our definition.

5.2.3 MIMo

In this section, we will provide additional details on motion transfer to the MIMo simulator, which was introduced in Section 4.2. By default, MIMo's joints are controlled using input torque, which does not allow for direct application of our extracted joint angles. The alternative position control method is not accurate and tends to oscillate for a significant duration of time, resulting in noisy output and prolonged duration of simulation. To overcome this issue and enable direct control of joint positions using the extracted joint angles, we had to disable simulation physics and directly adjust the joint values for each frame. However, the disabling of the physics simulation comes with certain drawbacks. One notable drawback is the disregard for the joint limits specified for MIMo, as described in Table 4.5. Consequently, there may be instances where certain body segments cross over each other or even appear inside each other, deviating from the intended anatomical constraints.

Regarding the implementation of joints in MIMo, the shoulder pitch and roll are replaced by the shoulder elevation and shoulder horizontal joints, as described in Section 4.2. Positive directions of arm movements generally align with our definition in Section 3.3, except for elbow flexion-extension, where extension is considered the positive direction. However, the directions of leg movements are opposite to our definition, except for hip yaw movement, which follows the same direction. The torso joint movement follows our defined positive directions, as well as the neck movements. Some examples of reconstructed poses are shown in Figure 5.5.

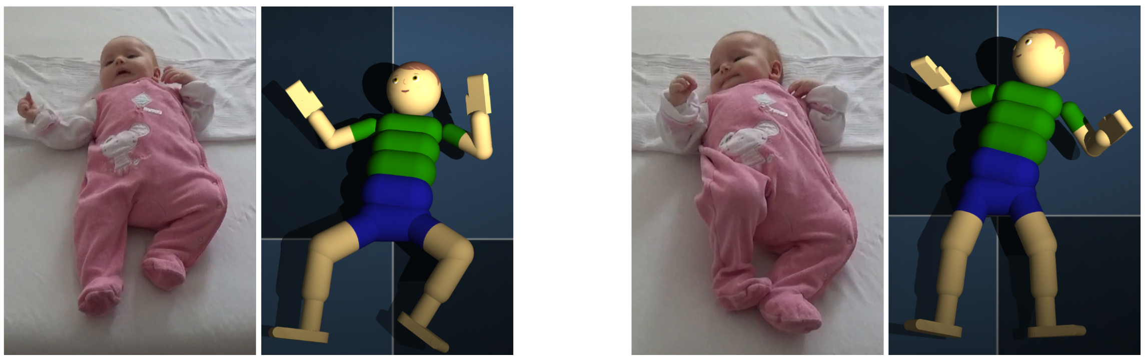


Figure 5.5: Transferred motion from baby to MIMo.

Sensory outputs

To capture the sensory outputs related to vision, MIMo's cameras, positioned in its eyes, were utilized. Camera outputs were collected for each frame and then concatenated to create a sequence that matches the sequence of input frames. An example of a visual output is shown in Figure 5.7. Regarding the activation of touch sensors, the simulator physics in MIMo needs to be activated to detect contact between two distinct body segments, such as the forearm and torso. However, to ensure the maintenance of the transferred body pose, physics is activated only for a brief period of time. During this interval, the contact is detected, resulting in activation of the corresponding touch sensors on the body of the MIMo. Figure 5.7 provides a visual representation of the activation of the touch sensor when contact is detected.

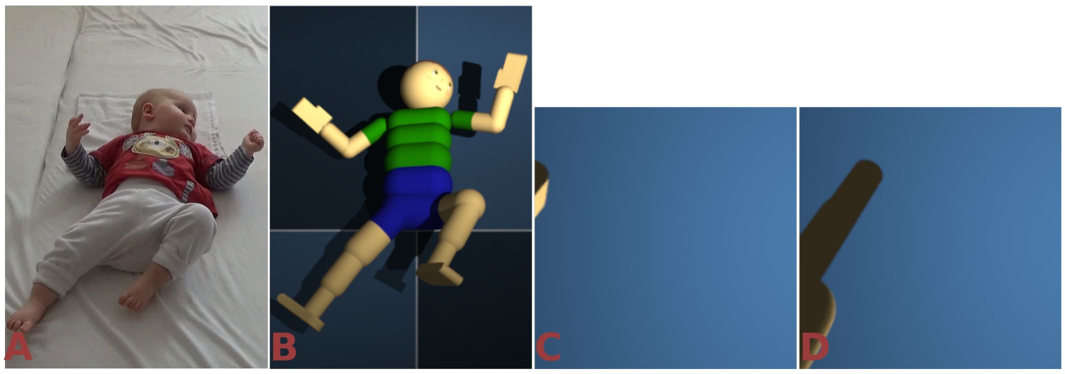


Figure 5.6: Transferred motion from baby to MIMo with a visual output. (A) real infant, (B) MIMo, (C) right eye, (D) left eye.

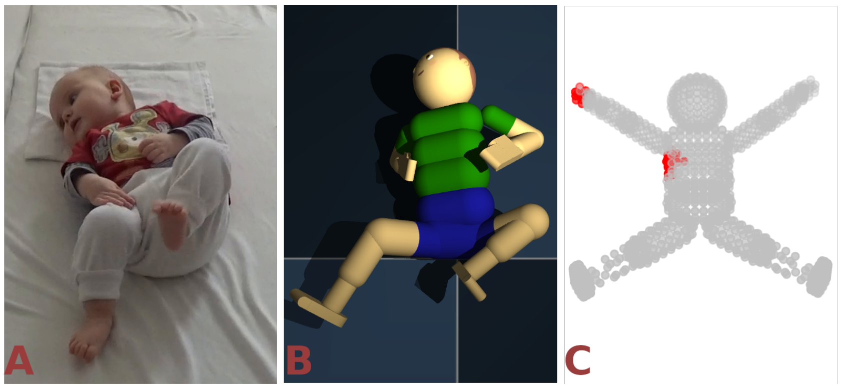


Figure 5.7: Transferred motion from baby to MIMo with a sensory output from tactile sensors. (A) real infant, (B) MIMo, (C) tactile output, red color indicates the location of touch.

Results

We obtained the 3D coordinates of the following joints for MIMo—left and right shoulder, elbow and wrist, left and right hip, knee, and ankle—for each of the 129,914 frames of the BBHMR dataset. These coordinates were then transformed using the canonization process described in Section 5.1. The upper arm, forearm, thigh and calf vectors were calculated as defined in Table 3.2. The MAE was estimated as described in Section 5.1, and the results are presented in Table 5.3. The body vector of the torso was not included in Table 5.3, as it directly affects the orientations of the arm vectors, therefore the error is mirrored in them. The mean angle error values range from 7.93 degrees to 11.97 degrees, while the median angle error values range from 6.90 to 11.49 degrees. These values indicate a reasonable alignment between the desired motion and the motion observed in the simulator, and this alignment was also visually confirmed during the evaluation process.

body vector	angle error [degrees]		
	mean	standard deviation	median (MAE)
right upper arm	11.97	4.08	11.49
left upper arm	9.08	4.67	8.21
right forearm	11.89	6.41	11.09
left forearm	7.93	5.56	6.90
right thigh	9.85	7.08	7.81
left thigh	10.90	6.98	8.77
right calf	10.18	6.67	8.39
left calf	10.83	7.04	8.92

Table 5.3: Angle error in degrees when transferring motion to the MIMo simulator.

5.2.4 Fetus simulator

The Fetus simulator is based on the MuJoCo physics engine, similar to the MIMo simulator, as stated in Section 4.3. Consequently, the implementation is quite similar. Unlike MIMo, the Fetus simulator has functional position control, allowing for successful motion transfer using the calculated joint angles directly. However, when position control is enabled, it is necessary to consider the joint limits, which results in cropping the movement to stay within those joint limits. The percentage of overall movement cropped for each joint can be found in Table 5.4.

	left	right
shoulder horizontal	0.39 %	0.17 %
shoulder elevation	2.26 %	6.59 %
shoulder yaw	13.65 %	22.42 %
elbow flexion/extension	4.54 %	3.10 %
hip pitch	3.5885 %	7.03 %
hip roll	0.24 %	1.06 %
hip yaw	38.86 %	46.48 %
knee flexion/extension	0.49 %	0.09 %

Table 5.4: Percentage of the frames, where the joint limits cropped the movement when processing BBHMR dataset for selected joints in the Fetus simulator.

In terms of joint movements, the Fetus simulator also replaces the movements of shoulder pitch and roll with the horizontal movements of the shoulders and the elevation of the shoulders, as mentioned in Section 4.3. The positive directions of arm movements on this platform align with our definition of positive directions. The positive directions of hip pitch and knee movement also align with our definition. However, the positive directions of hip roll, hip yaw, torso pitch, neck pitch, and neck roll movements are opposite to our definition. The remaining joints, not specifically mentioned, align with our definition. Figure 5.8 clearly shows one of the limitations of our work, since the input data is canonized, the spatial information like body orientation is lost out.

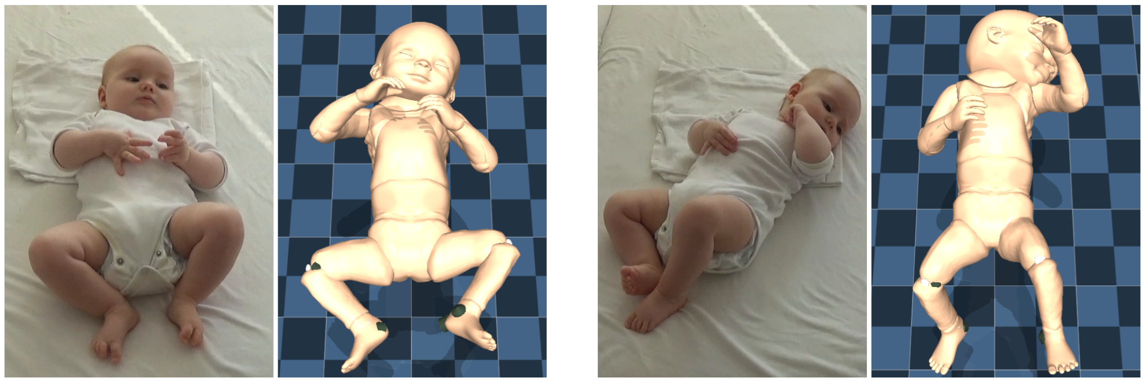


Figure 5.8: Transferred motion from baby to the Fetus simulator.

Sensory outputs

In terms of sensory outputs, we implemented cameras in the eyes of the fetus simulator to provide a visual perspective from the point of view of the infant. Activating the touch sensor was not a problem, since the physics simulation was already active, allowing for contact detection. However, we encountered limitations with the joint limits of the fetus simulator, which prevented certain parts of the body from making contact. To address this, we followed a similar approach as with the MIMo simulator: we disabled the physics simulation, directly manipulated the joint angle values, which resulted in ignoring the joint limits, and then temporarily enabled the physics simulation to detect contact. The results of the sensory output are shown in Figures 5.9 and 5.10.

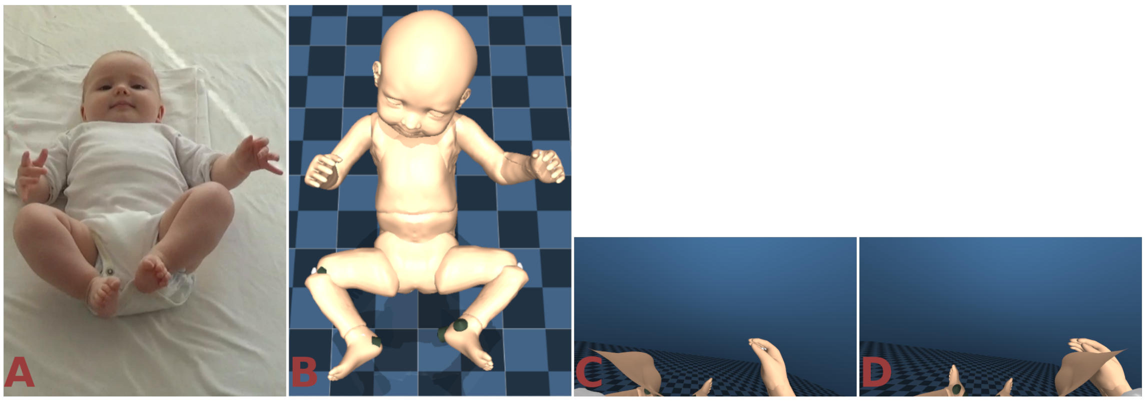


Figure 5.9: Transferred motion from baby to the Fetus simulator with visual output. (A) real baby, (B) Fetus simulator, (C) right eye, (D) left eye.

Results

For each input frame, we extracted the 3D coordinates of the following joints for the Fetus Simulator: left and right shoulder, elbow, and wrist; left and right hip, knee, and ankle. Using the canonization process described in Section 5.1, we transformed the coordinates and calculated the vectors for the upper arms, forearms, thighs, and calves. Then the MAE was calculated, and the results are presented in Table 5.5.

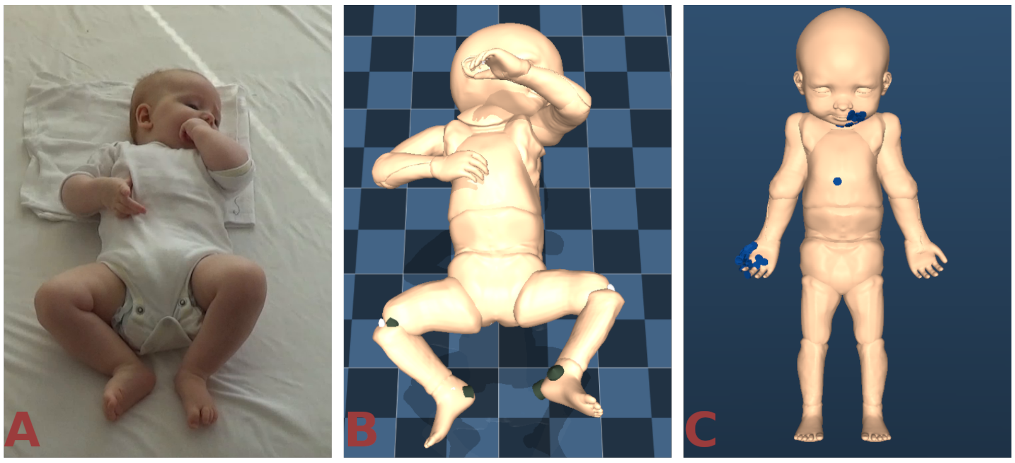


Figure 5.10: Transferred motion from baby to Fetus simulator with sensory output from tactile sensors. (A) real baby, (B) Fetus simulator, (C) tactile output, blue indicates contact.

body vector	angle error [degrees]		
	mean	standard deviation	median
right upper arm	6.44	3.22	6.10
left upper arm	6.62	3.92	6.05
right forearm	11.18	4.98	10.83
left forearm	10.21	5.33	9.52
right thigh	8.76	6.44	6.44
left thigh	9.04	6.91	6.91
right calf	8.66	6.28	6.28
left calf	9.94	6.44	6.44

Table 5.5: Angle error in degrees when transferring motion to the Fetus simulator.

The mean angle error values range from 6.44 to 11.18 degrees, while the median angle error values range from 6.05 to 10.83 degrees. It should be noted that the worst results were obtained for the forearm vectors, which is probably due to the complex structure of the arm in the Fetus simulator involving coupled joints. The reason for the improved accuracy in the upper arm and leg vectors could be attributed to the similarity in body lengths between infants and the Fetus simulator, as well as the exact replication of biological functioning and realism in the simulator.

5.3 Comparation of the platforms

The best results in the motion retargeting process from the MAE metric were obtained for the Fetus simulator. We attribute it to the realistic aspect of that simulator as well as the accurate biological representation. Additionally, the Fetus simulator's body lengths are adopted from infants of 1.36 months of age on average, which makes it closest to our BBHMR dataset consisting of infants aged between 8 and 23 weeks old. The results for MIMo (modeled after infants aged between 16 and 18 months) also showed low mean and median error angles. The worst results were obtained for the iCub robot, which we attribute to the implementation of iCub's kinematics, where the joints are not modeled as spherical and have a nontrivial offset of rotation between joint motor actuators. Overall, the results for the leg movement were better than for the arm movement. It is given by the fact, that the position of arm joints is directly influenced by the torso joints, so the task to accurately retarget the arm motion is more challenging.

Figure 5.11 shows one retargeted body pose from the infant and corresponding 3D keypoints to a dummy figure, iCub robot, MIMo, and Fetus simulator. The position of the legs is not exactly mirrored from the picture, which is given by the keypoint detection.

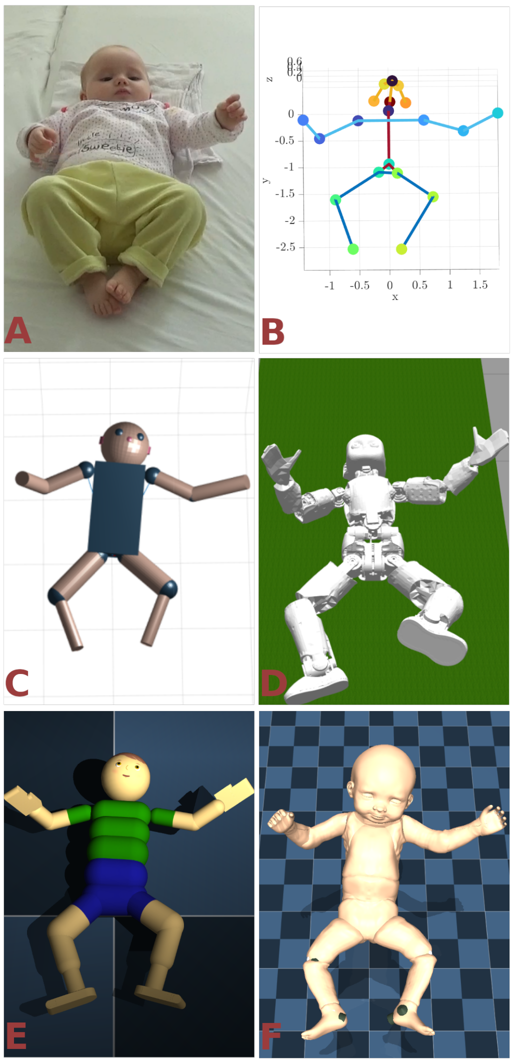


Figure 5.11: Retargeted body pose. (A) real infant, (B) 3D keypoints from BBHMR dataset (C) dummy figure, (D) iCub, (E) MIMo, (F) Fetus.

Chapter 6

Conclusion

In this thesis, we presented a method for retargeting infant movement from extracted 3D keypoints to humanoid robots and simulators. Such a method makes it possible to replay the movements of real infants in humanoid platforms and enhance the research with other sensory feedback. Sensory feedback may include vision collected from the eyes of the robot, touch from the artificial skin or tactile sensors placed on the robot body, or proprioception given by joint angles.

We began by defining the joint angles for the important body joints and their corresponding positive and negative directions. We then demonstrated the calculation of these joint angles using three-dimensional algebra techniques, such as rotation matrices and vector projections. To test the extracted angles, we constructed a whole body dummy figure in URDF format. The joint angles for each frame were concatenated to form joint angle trajectories. Such trajectories were then filtered using a fourth-order zero-phase low-pass Butterworth filter to reduce noise.

Next, we conducted motion transfer experiments on three target platforms: the iCub robot simulator, the MIMo simulator, and the Fetus simulator. Our goal was to investigate the feasibility of transferring the extracted joint angles and to assess the accuracy of the transferred motion. To evaluate the motion transfer accuracy, we used a metric that includes the median and mean angle errors between the reference body vector derived from the input and the body vector obtained from the target platforms. We calculated the mean, standard deviation, and median of these angle errors to assess overall performance.

The results of our experiments demonstrated promising outcomes in terms of motion retargeting accuracy. We observed relatively low mean and median angle errors on the MIMo simulator and the Fetus simulator, which represent the angle between the body segment of the input and the body segment of the target platform. Furthermore, we incorporated sensory feedback outputs, vision, and touch, which allowed us to inspect motion from the infant's perspective.

In conclusion, this thesis contributes to the field of motion transfer by presenting a method for calculating and transferring the joint angles to various target platforms. Our findings highlight the potential applications of motion transfer and lay the groundwork for further advancements in understanding and replicating infant motion.

Chapter 7

Discussion

The input dataset BBHMR consisted of 25 keypoints for the infants body. Due to the absence of hand keypoints (such as pinky and pointer finger), the wrist joint was treated as an immovable end effector, resulting in the loss of 3 DoF. Additionally, to ensure consistency between the upper body and lower body limbs, the ankle joint was also considered an immovable end effector, despite the availability of foot keypoints. As a result, the motion retargeting process excluded the motion of wrists and ankles, leading to a reduction in the overall DoF involved in motion transfer. During the construction of the BBHMR dataset, the canonization process is used, which results in losing the information on the body orientation of the infant, therefore all the retargeted body poses had fixed hips in the same location, differing from the real videos.

Throughout the experiments, several challenges were encountered, primarily related to the control of the robot platforms. The transfer of motion to the iCub platform was particularly problematic due to joint limits, which resulted in a significant proportion of cropped movements. To overcome this issue, the joint limits had to be disabled, allowing the successful transfer of the extracted joint angles. However, this modification had consequences, such as the occurrence of limbs penetrating each other. In MIMo simulation, which does not have positional joint control implemented, we were obliged to use a manual modification of the joint values without simulating physics. Consequently, when extracting touch sensor activation, the force values resulting from the contact between two body parts could not be obtained, limiting the output to binary values—*touched* or *not touched*. Similarly, although the Fetus simulator is equipped with functioning positional joint control, it was not used due to the joint control limits, which prevented certain body parts from touching each other, *e.g.*, hands could not touch the infant's torso. Therefore, the same process used for MIMo was applied, resulting in similar limits for touch sensor activation.

Despite the challenges encountered, we were able to achieve relatively low mean and median angle errors on the target platforms. The best results were obtained while retargeting the movement to the Fetus platform. We attribute this to the fact that the Fetus simulator is extremely realistic and follows the kinematic structure of the infant or fetus accurately. Low mean and median errors were also obtained for the MIMo platform, which indicates a reasonable alignment between the desired motion and the motion observed in the simulator. In general, the overall retargeting process proved to be more accurate for the legs than for the arms. However, the position of the arm is directly influenced by the torso joint angles, so the task is more complex.

Chapter 8

Future Work

Adding dynamic control to the simulations

Our approach focused primarily on the kinematic aspects of simulations, disregarding the joint limits, velocities, and torques. As a result, an area for future work lies in optimizing the joint angle trajectories to align with the range of motions within the simulation and incorporating the dynamical aspects of the simulations. By doing so, additional valuable information about infant movement could be extracted, such as muscle activation, forces values during touch, and maximal limb velocities. This would contribute to a deeper understanding of infant movement and provide enhanced insights into their motor capabilities.

Dynamic changes of body lengths of simulators and optimizing the shape of the model

One of the main challenges we encountered during the experiments was the differences between the body lengths of real infants and the body lengths of the models on the target platforms. In case the model is constructed using basic geometrical shapes, as is done in the case of MIMo, the sizes of these shapes are predefined in the XML files. When processing the infant's motion, it could be beneficial to extract the infant's body lengths and dynamically adjust the sizes of shapes to accurately represent the infant's body. Furthermore, it could be also beneficial to extract the shape of an infant captured in the video and adjust the shape of the models in simulators according to it. These modifications would enable for more precise extraction of sensory output and provide a more realistic representation of the infant's body within the simulation environment.

Trajectory optimization based on touch location

During the extraction of sensory output related to touch, we observed that the occurrence of touch was not completely accurate. There were instances where touch was detected even when the infant's body parts were not in contact and, contrarily, instances where the touch was not detected while the infant's body parts actually were in contact. To address this issue, one possible solution would be to annotate the frames where the touch actually occurred and implement trajectory optimization techniques in order to replicate the occurrence on a target platform. However, it is important to note that such trajectory optimization would require specific knowledge of the target platform and its body shapes to accurately recreate the desired action. Consequently, the resulting joint trajectories may not be generally applicable to any target platform.

Matching the vision output to infant's abilities of vision

Another sensory output we extracted in this thesis was a vision. The output was obtained from RGB cameras located in the eyes of the models in simulators. However, it is important to consider that the vision of infants is actively developing in the early stages of their lives. McCulloch in [44] says that infant vision can initially be described as foggy, while the fog gradually disappears as the infant ages. Additionally, sensitivity to colors, brightness, and contrast is reduced in newborns and improves over the first months of life. It could be useful to alter the images from RGB cameras according to the knowledge of the infant's visual development to improve the realistic factor of the vision.

The head pose estimation improvement

As stated in Chapter 2, the estimation of the head pose typically relies on a large number of keypoints on the head to ensure accuracy. The direction of the visual output is directly influenced by the angles of the neck and head. To further improve the infants' perspective, it would be useful to use a different input dataset, which contains a greater number of keypoints on the head. Furthermore, some of the robot platforms also provide control of the eyes, such as eye vergence or tilt. To improve the realistic factor of visual output, it would also be beneficial to extract these features from the videos of infants and incorporate them into the whole motion transfer process.

Bibliography

[1] A. Cangelosi and M. Schlesinger, *Developmental robotics: From babies to robots.* MIT press, 2015.

[2] M. Hoffmann and R. Pfeifer, “Robots as powerful allies for the study of embodied cognition from the bottom up,” in *The Oxford Handbook of 4E Cognition.*

[3] A. DiMercurio, J. P. Connell, M. Clark, and D. Corbetta, “A naturalistic observation of spontaneous touches to the body and environment in the first 2 months of life,” *Frontiers in Psychology*, vol. 9, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2018.02613

[4] A. L. H. van der Meer, F. R. van der Weel, and D. N. Lee, “The functional significance of arm movements in neonates,” *Science*, vol. 267, no. 5198, pp. 693–695, 1995. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.7839147

[5] H. Kanazawa, Y. Yamada, K. Tanaka, M. Kawai, F. Niwa, K. Iwanaga, and Y. Kuniyoshi, “Open-ended movements structure sensorimotor information in early human development,” *Proceedings of the National Academy of Sciences*, vol. 120, no. 1, 2023.

[6] M. Gleicher, “Retargeting Motion to New Characters,” in *Proc. of the 25th Annual Conference on Computer Graphics and Interactive Techniques.* ACM, 1998, pp. 33–42. [Online]. Available: https://doi.org/10.1145/280814.280820

[7] E. C. Gonen, Y. Jung Chae, and C. Kim, “Imitation of Human Upper-Body Motions by Humanoid Robots,” in *2019 16th International Conference on Ubiquitous Robots (UR)*, 2019, pp. 334–341.

[8] N. Pollard, J. Hodgins, M. Riley, and C. Atkeson, “Adapting human motion for the control of a humanoid robot,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, 2002, pp. 1390–1397 vol.2.

[9] M. Riley, A. Ude, K. Wade, and C. Atkeson, “Enabling real-time full-body imitation: a natural way of transferring human movement to humanoids,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 2003, pp. 2368–2374 vol.2.

[10] J. Koenemann, F. Burget, and M. Bennewitz, “Real-time imitation of human whole-body motions by humanoids,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2806–2812.

[11] L. Penco, B. Clement, V. Modugno, E. Mingo Hoffman, G. Nava, D. Pucci, N. G. Tsagarakis, J. B. Mouret, and S. Ivaldi, "Robust Real-Time Whole-Body Motion Retargeting from Human to Humanoid," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018, pp. 425–432.

[12] K. Darvish, Y. Tirupachuri, G. Romualdi, L. Rapetti, D. Ferigo, F. J. A. Chavez, and D. Pucci, "Whole-Body Geometric Retargeting for Humanoid Robots," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 679–686.

[13] K. Khan, R. U. Khan, R. Leonardi, P. Migliorati, and S. Benini, "Head pose estimation: A survey of the last ten years," *Signal Processing: Image Communication*, vol. 99, p. 116479, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596521002332

[14] P. Barra, S. Barra, C. Bisogni, M. De Marsico, and M. Nappi, "Web-Shaped Model for Head Pose Estimation: An Approach for Best Exemplar Selection," *IEEE Transactions on Image Processing*, vol. 29, pp. 5457–5468, 2020.

[15] A. Nikolaidis and I. Pitas, "Facial Feature Extraction and Determination of Pose," in *Noblesse Workshop on Non-Linear Model Based Image Analysis*, S. Marshall, N. R. Harvey, and D. Shah, Eds. London: Springer London, 1998, pp. 257–262.

[16] T. Vatahska, M. Bennewitz, and S. Behnke, "Feature-based head pose estimation from images," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 330–335.

[17] D. Karch, K.-S. Kim, K. Wochner, J. Pietz, H. Dickhaus, and H. Philippi, "Quantification of the segmental kinematics of spontaneous infant movements," *Journal of Biomechanics*, vol. 41, no. 13, pp. 2860–2867, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021929008003400

[18] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, 2007.

[19] O. Fiala, "Babies to baby humanoid robots motion retargeting," 2023. [Online]. Available: https://gitlab.fel.cvut.cz/body-schema/motion-retargeting

[20] N. Hesse, C. Bodensteiner, M. Arens, U. G. Hofmann, R. Weinberger, and A. Sebastian Schroeder, "Computer Vision for Medical Infant Motion Analysis: State of the Art and RGB-D Data Set," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[21] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[22] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black, "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] N. Hesse, S. Pujades, J. Romero, M. J. Black, C. Bodensteiner, M. Arens, U. G. Hofmann, U. Tacke, M. Hadders-Algra, R. Weinberger, W. Müller-Felber, and A. Sebastian Schroeder, ""Learning an Infant Body Model from RGB-D Data for Accurate Full Body Motion Analysis"," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018.* Springer International Publishing, 2018, pp. 792–800.

[24] J. Khoury, S. T. Popescu, F. Gama, V. Marcel, and M. Hoffmann, "Self-touch and other spontaneous behavior patterns in early infancy," in *2022 Joint IEEE 12th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2022, pp. 1–8.

[25] M. H. Ang and V. D. Tourassis, "Singularities of Euler and Roll-Pitch-Yaw Representations," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-23, no. 3, pp. 317–324, 1987.

[26] A. Leonhard, "Signed angle between the two 3d vectors with the same origin within the same plane," Stack Overflow, (version: 2020-10-15). [Online]. Available: https://stackoverflow.com/a/33920320

[27] J. Narayan, B. Kalita, and S. Dwivedy, "Development of Robot-based Upper Limb Devices for Rehabilitation Purposes: A Systematic Review," *Augmented Human Research*, 12 2021.

[28] B. Kalita, J. Narayan, and S. Dwivedy, "Development of Active Lower Limb Robotic-Based Orthosis and Exoskeleton Devices: A Systematic Review," *International Journal of Social Robotics*, vol. 13, 07 2021.

[29] E. Grimpampi, V. Bonnet, A. Taviani, and C. Mazzà, "Estimate of lower trunk angles in pathological gaits using gyroscope data," *Gait & posture*, vol. 38, no. 3, pp. 523–527, 2013.

[30] T. Jantunen, J. Mesch, A. Puupponen, and J. Laaksonen, "On the rhythm of head movements in finnish and swedish sign language sentences," 05 2016, pp. 850–853.

[31] "URDF – ROS Wiki," http://wiki.ros.org/urdf, Accessed: 2023-05-21.

[32] R. Sandy, "Anthrokids–anthropometric data of children," *National Insititute of Standards and Technology*, 1977. [Online]. Available: https://math.nist.gov/~SRessler/anthrokids/

[33] D. Mattern, F. M. López, M. R. Ernst, A. Aubret, and J. Triesch, "MIMo: A Multi-Modal Infant Model for Studying Cognitive Development in Humans and AIs," in *2022 IEEE International Conference on Development and Learning (ICDL)*, 2022, pp. 23–29.

[34] D. Kim, H. Kanazawa, and Y. Kuniyoshi, “Simulating a Human Fetus in Soft Uterus,” in *2022 IEEE International Conference on Development and Learning (ICDL)*, 2022, pp. 135–141.

[35] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori, “An Open-Source Simulator for Cognitive Robotics Research: The Prototype of the iCub Humanoid Robot Simulator,” in *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, ser. PerMIS ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 57–61. [Online]. Available: https://doi.org/10.1145/1774674.1774684

[36] E. Todorov, T. Erez, and Y. Tassa, “MuJoCo: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[37] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[38] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, A. Bernardino, and L. Montesano, “The iCub humanoid robot: An open-systems platform for research in cognitive development,” *Neural Networks*, vol. 23, no. 8, pp. 1125–1134, 2010, social Cognition: From Babies to Robots. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608010001619

[39] iCub Tech Docs. [Online]. Available: https://icub-tech-iit.github.io/documentation/

[40] A. Parmiggiani, M. Maggiali, L. Natale, F. Nori, A. Schmitz, N. Tsagarakis, J. Santos-Victor, F. Becchi, G. Sandini, and G. Metta, “The Design of the iCub Humanoid Robot,” *International Journal of Humanoid Robotics*, vol. 9, 12 2012.

[41] F. Crenna, G. B. Rossi, and M. Berardengo, “Filtering Biomechanical Signals in Movement Analysis,” *Sensors*, vol. 21, no. 13, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/13/4580

[42] D. A. Winter, H. Sidwall, and D. A. Hobson, “Measurement and reduction of noise in kinematics of locomotion,” *Journal of Biomechanics*, vol. 7, no. 2, pp. 157–159, 1974. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021929074900566

[43] B. Yu, D. Gabriel, L. Noble, and K.-N. An, “Estimate of the Optimum Cutoff Frequency for the Butterworth Low-Pass Digital Filter,” *Journal of applied biomechanics*, vol. 15, pp. 319–329, 08 1999.

[44] D. L. McCulloch, “The infant patient,” *Ophthalmic and Physiological Optics*, vol. 18, no. 2, pp. 140–146, 1998.