Mapping movements and other input data from Human performance to Robots

Tony Huu Phuoc Nguyen

Thesis submitted for the degree of
Master in Robotikk og Intelligente Systemer
60 credits

Institutt for Informatikk
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Mapping movements and other input data from Human performance to Robots

Tony Huu Phuoc Nguyen

Mapping movements and other input data from Human performance to Robots

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

For humans, imitating and adapting from another human is a natural task, however the same task cannot be done for robots and offers great challenges in robotic development that researchers strive to solve. Programming the robot motion is difficult and requires analytical equations and it is time-consuming. Imitation learning in robotic research field aims to solve such problems through observation.

The thesis proposes a novel method that contributes in the field of imitation learning using motion capture as reference for the robotic manipulator to learn from. Taking inspiration from machine-learning and natural evolution, the method introduces a weight-matrix that maps the relation in terms of kinematics and other features between robotic manipulator and motion actor and using GA to optimize for the mapping relation.

In order to test the proposed method, two experiments were conducted with the aid of V-REP robotic simulator. UR5 robotic manipulator was used as the main manipulator. The first experiment was designed to capture the relation between motion capture actor and robotic manipulator. The second experiment was extracting a human limb and designed to transfer the human limb motion on the UR5-manipulator. The motion capture dataset used in both of the experiment is HDM05 which is marker-based.

The proposed method shows promising results and the motion capture dataset used in thesis is marker-based and as such, the proposed method manages to handle different amount of markers which are different topology. The first experiment managed to capture the motions unqiue characteristics and the second experiment imitated to the motion capture motion in complex motion such as workout and dancing.

Acknowledgement

Before starting the journey of my master thesis I would just like to express my gratitude to the fantastic people who have assisted me. First and foremost, Kyrre Glette, for his supportive advices and pushing me further and Kristian Nymoen for giving me suggestions on MoCap-datasets and also his further improvement of implementation on the existing MoCap-Toolbox framework made by Petri Toiviainen and Birgitta Burger. I would also like to thank the University of Oslo, Institute of Informatics for this opportunity of acquiring my master-degree with them.

Second, I would like to thank my family especially my mother, friends and other struggling master-students on motivating me and help me when I faced multiple challenges. Lastly, I would like to thank Google and StackOverflow for assisting me on my research and programming issues.

Now that I have expressed my outmost gratitude, it is now the time to start the journey of my master-thesis, I wish you a safe and pleasant journey, adventurer!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robots [1] are slowly growing to become a big part of our everyday life. They are used in hospitals for surgery, drones for package delivery, autonomous vehicle for transportation and even in small various software services and applications such as Alexa [2]. As the technology growth continues to advance so will the interaction between robots and humans being more common. Even humans have contemplated integrating robotic applications into human body such as using smart robotic prosthetic arm to replace missing limb [7] or exoskeleton for additional enchancement in mobility or physical strength for the handicapped [24].

For humans, performing various activities such as walking, running, and etc is a natural task. This comes natural from the idea humans are imitating other humans behaviour and copy them to perform the exact same motion. For robots however, it is far more difficult to make it move and recreate the same motion and offers great challenges in robotic development that researchers strive to solve. Programming every movement in a robot is tedious and time-consuming [8] and to combat this, the term *Programming by Demonstration*(PbD)[3] is technique of demonstrating a task for a robot learn from observations. This is also known as imitation learning.

The field of imitation learning within robotics uses numerous methodological where biology-based methods such as neural networks [18][56] and evolutionary computing [42][52] are used for robot learning methods. Two studies conducted by [20] and [8] shows there are frequently more applications using neural networks rather than evolutionary computing.

The motivational goal of the thesis is to contribute in the field of Programming by Demonstration or Imitation Learning by using evolutionary computing

[1] Meaning "forced labor" was first used to denote a fictional humanoid in 1920 play by Czech Writer, Karel Capek and postulated the technological creation of artifical human bodies without souls to fit the new class of manufactured, artifical workers

[2] Alexa is virtual assistant AI technology developed by Amazon first used in Amazon Echo Smart Speakers

[3] Programming by Demonstration(PbD) appeared in software development as early as 1980s and is defined as a sequence of operations where a user performs and the computer repeats the process

with motion capture data with an ambition that can enlighten other researchers of adapting to evolutionary strategies approach.

1.2 Goal of the thesis

The goal in the thesis is to investigate if human motion capture data can be mapped into robotic manipulator using variety of motion capture markers by optimizing weights using evolutionary algorithm. The solution is determined by how close the GA is approximating the distance of the robotic manipulator and motion capture data. As such, the hypothesis is if using more markers could improve the mapping. In order to answer this hypothesis, the thesis will be investigating two goals to help answer this.

The first goal is observing how motion-capture data is mapped into lower degree-of-freedom robots and if it is possible to capture the some unique characteristics or the *"essence"*[4] of the motion.

1. Create and implement weight-matrix for optimizing the parameters for Full-Body motion onto 6-DOF robot manipulator with different amount of motion-capture markers using Evolutionary Algorithm.

The second goal involves extracting a portion of the human motion capture data for example human arm and determine if the robotic manipulator can imitate the same motion trajectory with different amount of markers.

2. Create and implement weight-matrix for optimizing the parameters for Limb-body motion onto 6-DOF using different amount of motion-capture markers.

Fullfilling these two goals will achieve the goal of the thesis to be able to prove or disprove the hypothesis.

1.3 Implementation

Short summary of what was done in the thesis to achieve the goal of the thesis.

1. Preprocessing Motion Capture Data
2. Finding the Forward-kinematics for UR5-manipulator
3. V-REP Simulation Setup
4. Implementation of the two proposed methods, EJDM and BCDM
5. Multiple experiments and tests were done on UR5-manipulator using the two methods proposed

[4]Definition of Essence; The basic, real and invariable nature of a thing or its significant individual feature or features

1.4 Outline of the thesis

The structure of the thesis can be categorially be divided into a total of six chapters including the introduction chapter. The other remaining five chapters are **Theoretical Background**, **Motion Capture Data**, **Methods and Implementation**, **Experiments and Results**, and **Discussion**.

Chapter II - Theoretical Background
An introduction to the relevant theory needed for the thesis.

Chapter III - Motion Capture Data
Overview over motion capture and motion capture dataset.

Chapter IV - Methods and Implementation
Describes the method proposed and implementation of the system.

Chapter V - Experiments and Results
Describes the experimental setup and the related results for the two main experiments, full-body and BCDM using variety of motion capture markers.

Chapter VI - Discussion
Explains the general discussion around the experiments and results and finishing the thesis with a conclusion of the work in the thesis and future work.

Chapter 2

Theoretical Background

2.1 Robot Motion

Generating robotic motion[1] is not trivial task because there are many different types of motion. The various motions could be bipedal [47], four-legged [45], upperbody [64] and last for robotic arm which the thesis will focus on. Some of the challenges include how to properly map between human motion to robots. The use of input-deviced for moving the robot such as [22] [2] could be used where sensors register the movement or the signal from human and then the robotic manipulator use the data to convert it.

The restrictions between human and robot in terms of kinematic and dynamic constraints appears to be more problematic than expected [59] due to varying link-lengths, orientation and position of the joints. There are two methods of mapping movements, the first one is analyzing and finding the analytical equations or mathematical modeling of robots and that may be difficult depending on the amount of DOFs and the robots layout. The second approach involves the use of learning algorithms such as the artifical neural networks [28] for robot control or reinforcement learning [33] [19] for computing the inverse-kinematics. The idea of these mentioned machine-learning algorithms is to find the relation between the input data and ground truth to map the relationship between kinematic models between human and robot.

A form of reference from motion can be used for the robot to learn form. It is called motion-capture and will be explained in greater details at chapter 3. Even with the reference motion, there exists some underlying limitations for using motion-capture.

Matsui et al. [34] experimented that by copying the joint-angles directly from motion-capture, yielded worse results as the kinematics between human and moCap-actor being vastly different. By copying the changes in positions and using neural network, the robot managed to perform more human-like motions and realistic. Here, the authors used 20 markers on both the robot and themselves. Similar in Stanton et al. [56], neural networks were used for every robotic joint to map the relation between motion-capture and robotic. The interesting aspect of that research mentioned are the lack of having analytical model or the mathematical modeling of the robot, i.e the inverse kinematics,

[1] Motion is the phenomenon in which an object changes its position over time.

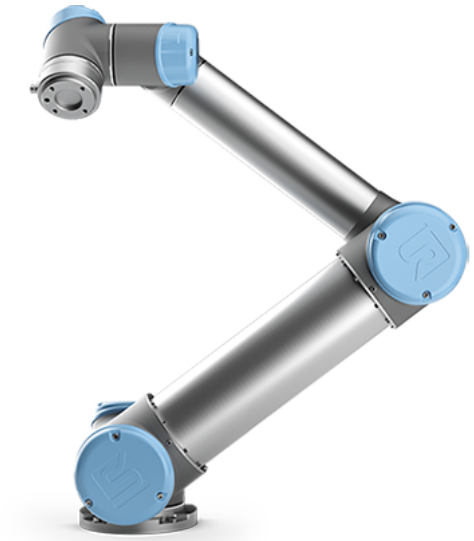


Figure 2.1: UR5 Robotic Arm. Image taken from: https://www.cobotnor.no/ur-og-ur-e-serien/ur5/

proving the capablilites of heuristic methods.

As such, using evolutionary computing or more specific, evolutionary robotics is heuristic method designed for optimization problems. One algorithm that comes to mind is genetic algorithm which will also be explained in this chapter [23] later used. The objctive function or criterion function is minimizing the traveling time and space while not exceeding torque. While the GA was able to solve their optimization problem, their fitness function required a lot of constants meaning analytical knowledge of the robotic arm is needed.

2.2 Robot Design

Robot exists in various shapes and forms depending on its purpose. Given their shape and forms they are commonly used to solve specific sets of tasks. Robots that are intending to replicate humans tends to solve human related tasks [1], [43]. Other robots draw inspiration from nature and animals such as the 6-legged spider [6] and the 4-legged dog-like [45].

2.3 Introduction To Robotic Manipulators

The robotic manipulators are robots with mechanical arm connected by rigid links connected by joints to form a kinematic chain such as the one shown in figure 2.1. The joints can vary from revolute, prismatic to far more complex joints such as sphere [37] or ball joint [13]. A revolute joint and prismatic joints have only one degree-of-freedom(DOF). The revolute joint has one DOF and it only rotates along the z-axis denoted by z_i. Robotic manipulator are commonly stationary where its initial position $o_0x_0y_0z_0$ is referred as base-frame. It is also

the following notation which is going to be used when referring to coordinate systems.

Since the links only has certain reach means the overall movement is also limited to its space. Its total space of movement is known as configuration space or workspace. In this thesis it will be focused entirely on revolute joints as it is the one that is used.

A robot manipulator will have n joints and $n+1$ links and that means the total amount of joints will be J_i $i \in 1, ..., n$ and for links l_i $i \in 0, ..., n$. This means every joint i connects link l_{i-1} to link l_i. For each joint a coordinate frame $o_i x_i y_i z_i$ is attached to the link l_i and when the link l_i move so will its attached coordinate frame.

2.3.1 Modeling Robot Manipulators

The derivation of the forward and velocity kinematics and the dynamics equations for the manipulators are well documented and established [54] whereas the most popular approach is Denavit-Hartenberg convention(DH). It was first introduced in 1955 in order to standardize the coordinate frames for spatial linkages. The procedure can be seen as step-by-step operations. The homogenous transformations and rigid motions represent the orientations and position as matrix. Homogenous transformations combines both rotations and position together it can be seen as transforming from one coordinate frame to the next one. Here, each homogenous matrix represents the orientation and position for one particular joint i.

2.3.2 Rigid motion and Euler-angles

Rigid motions and homogenous transformations are used to describe the relative positions and orientations between the coordinate systems that are assigned to each joint and link. By combining the operations of rotation and translation into single matrix it can be used to derive the forward-kinematics. That is explained later in section 2.3.5.

Rigid motions can be defined as an ordered pair (R, d) where $d \in R^3$ and $R \in \mathbf{SO(3)}$ is a rotation matrix of Special Orthogonal group of order three. This means for any $R \in \mathbf{SO(n)}$ the following property must hold $R^T = R^{-1}$ and $det(R) = 1$. Rotation matrices, R can be used to represent orientation of one coordinate frame with respect to another coordinate frame. As such it can also be used to transform from one coordinate frame to another with respect to previous coordinate frame.

For example, consider two frames $o_i x_i y_i z_i$ as the first frame and $o_j x_j y_j z_j$ as the next where j is the successor from frame i, then the next successive frame $o_k x_k y_k z_k$ can be obtained with the previous two frames as,

$$R_k^i = R_j^i R_k^j \tag{2.1}$$

where the rotation, R_k^i is the rotation on frame k with respect to i frame.

2.3.3 Euler Rotation

The orientation of frame $o_j x_j y_j z_j$ relative to frame $o_i x_i y_j z_i$ can be found with three angles $\phi = [\varphi, \psi, \vartheta]^T$ known as Euler-angles. Using following angles, the

rotation matrix can be obtained by three elemental rotations.

Rotation matrix for rotation around x-axis with θ degree and is considered *roll* rotation.

$$Rot_{x,\varphi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{2.2}$$

Rotation matrix for rotation around y-axis with θ degree and the *pitch* rotation.

$$Rot_{y,\theta} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix} \tag{2.3}$$

Rotation matrix for rotation around z-axis with θ degree and the *yaw* rotation.

$$Rot_{z,\vartheta} = \begin{bmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.3.4 Denavit-Hartenberg(DH)-Convention

Homogenous transformations are used to describe the relationship between coordinate frames $o_i x_i y_i z_i$, and $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$, $\forall i \in 1, ..., n$. To define a rigid motion, a total of six parameters are needed to derive the where three parameters are for rotation and the remaining three for position.

The Denavit-Hartenberg(DH) is standardized procedure that simplifies the handling of rigid motions as it reduces the amount of parameters required to obtain the transformations matrices. DH-convention reduces the six parameters down to four through exploitation of common geometry and choice between origin and coordinate axes. The homogenous transformation matrix becomes $H \in R^{4\times4}$

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{2.5}$$

where $R \in \mathbf{SO(3)}$ and $d \in R^3$ and finding its inverse H^{-1} is simple due to the property of Special-Orthogonal group and only requires the transposed but this is also known.

$$H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

To calculate the next transformation, the first step is to use matrix-multiplication of the homogenous matrices

$$H_j^i = H_{i+1}^i \dots H_j^{j-1} = \begin{bmatrix} R_{i-1}^i & d_{i-1}^i \\ 0 & 1 \end{bmatrix} \dots \begin{bmatrix} R_j^{j-1} & d_j^{j-1} \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

This homogenous matrix, A_i, in DH-convention is represented as a product of four basic transformations and it is known as transformation matrix

$$
\begin{aligned}
A_i^{i-1} &= Rot_{z,\theta} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha} \\
&= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\alpha_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

where the four parameters,a_i, d_i, α_i and θ_i can be seen in equation 2.8 above. These parameters are the joint angle, link-length, link-offset and link twist of joint i and link l_i. They are defined as

1. a_i = distance along x_i from the intersection of the x_i and z_{i-1} axes to coordinate frame i
2. d_i = distance along z_{i-1} from coordinate frame $i-1$ to the intersection of x_i and z_{i-1} axes
3. α_i = the angle from z_{i-1} to z_i measured about x_i
4. θ_i = the angle from x_{i-1} to x_i measured about z_{i-1}.

DH-convention requires also the following parameters to follow set of rules that must be satisfied and can be described as:

1. The axis z_i is the axis of revolute of joint j_{i+1}.
2. The axis x_i is perpendicular to the axis z_{i-1} and z_i
3. The axis x_i intersects the axis z_{i-1}
4. The coordinate frames can be used with right-hand-rule (RHR)

Not only is it applicable to positions and orientations, but the A_i matrix can also be used to derive the velocity kinematics. In this thesis, the dynamic models and velocity kinematics will be excluded as it is not used. Hence, only the forward-kinematics will be explained and derived in details. It is also assumed the base-frame is fixed and the links are considered rigid.

2.3.5 Forward-Kinematics

Forward-kinematics describes the motion of the manipulator and are used to determine the position and orientation of the end-effector by using joint variables q.

For deriving forward-kinematics, it is assumed A_i matrices from equation 2.8 are known for every joint i and the A_i matrices can be plotted into equation 2.7 as series of matrix multiplication. As such the position and the orientation of the end-effector are then given by

$$T_n^0 = A_i \dots A_n \tag{2.8}$$

Assuming all of the joints are revolute, then equation 2.8 can be written as function of joint variable q_i and this yields the following equation 2.9,

$$T_n^0(q_i) = A_i(q_i) \dots A_n(q_n) \tag{2.9}$$

In the last transformation when $T_n^0(q)$ is reached, that is the position and orientation of the end-effector with respect to base-frame.

2.4 Genetic Algorithm

Genetic Algorithm, abbreviated GA, is inspired by natural selection in biology [35] where selection of fittest individuals thrive in an environment and will have much higher probability to produce offsprings. Weaker individuals must therefore adapt to its given surrounding environment through mutation and crossover. Section below 2.4.1 will go through some of the most common terms in GAs.

2.4.1 Biological terminology

All living organisms consist of cells and each cell contains one or more of the same set of one or more chromosomes that serves as the DNA for the organism. Humans have 46 possible chromosomes while other living animals have different amount of chromosomes. It is the combination of the chromosomes that defines and shapes such personal traits and appearance as an individual. Looking at the chromose in molecular level unveils the property of natural genetics. The outside is that is the appearance of the individual is known as phenotypic features. These phenotypic features can be described by genotype. Here, the genotype represents the encoding of the sequence of genes. Every gene has possible set of values known as alleles. Through recombination and mutation will cause the phenotypic features to change.

2.4.2 Genetic Algorithm (GA)

Genetic algorithms is stochastic search method based on the prinicipal of natural genetic systems [17] where the idea is to maintain possible solutions that evolves over time and improves on its previous successor in process of competition and controlled environment. The evolution process is applied to known initial candidates where the evolution process explore for better candidates in search space where there are many possibilities of good and more efficient candidates for one particular problem. For optimzation problems, the goal is either minimizing or maximizing an objective function, the GAs has proven to yield great results.

Fitness is defined as the genetic representation of an individual and measures the quality of the represented solution. Fitness landscape can be seen as the search space of all possible solutions which we will call S. In the search space there exists a metric, or scalar fitness function defined over all the elements in S [63]. These elements can be considered as peaks where some are higher than

other as illustrated in fig 2.2. The smaller ones can be considered local optima while the highest peak in the search space S is global optima. It is the global optima in which the GA is trying to find as this is considered the best solution for a given problem and it is by searching for the highest either by maximizing or minimizing the fitness function.

2.4.3 Fitness Function

In order to verify whether the individuals are fit in the population the need of an evaluation with respect to gene-encoding is required. In GA it is often called fitness function and is a method of measuring the overall solution for the individual. The fitness value returned from the evaluation is deciding its probability of being in the next population for the next pool. Higher the fitness, the higher the probability it is included in the next gene pool with the goal of achieving optimal solutions. However, one fitness function cannot be applied to every GA's algorithm because every fitness function is dependent on the problem it is trying to solve. The reason behind lies on the individual's gene-encoding.

Knowing the fitness function is dependent on its gene-encoding, allows creation of fitness functions where function can include set of criterias for the problem-solving. As such, through these criterias it has a goal in mind, hoping to achieve with GA's algorithm for optimization. For example, given a bit-string of length L, and a fitness function whose goal is to maximize the decimal value for bits. From this example it can be deduced the optimal way is to flip all the bits to "1". However, assume the fitness function is to maximize only one particular region within the bit-string and it can done so in the fitness function adding a constraint that maximizes only the particular region. It does not in general have to be one criteria but could expand to multiple criteras. Therefore from this example, the more knowledge one has of the problem, the more can one explore individual's potential to optimal solutions. As mentioned the fitness function is an objective function and usally returns real-valued scalar value.

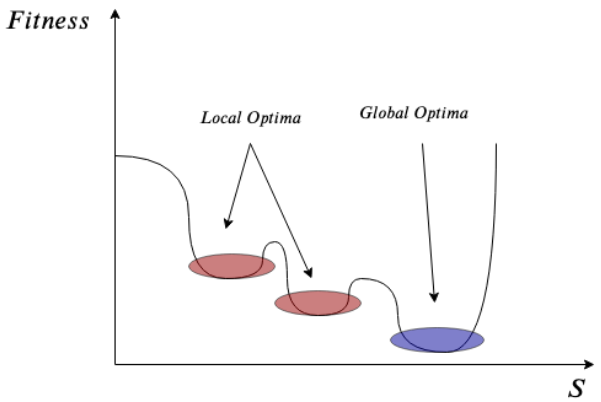


Figure 2.2: Illustration of fitness landscape where horizontal axis, S, is the search space and on the vertical axis, $fitness$, is the quality of the solution for every element in S. The red areas indicates the local optima where the solutions are subpar while the blue area is the desired fitness and known as global optima for the given search space S

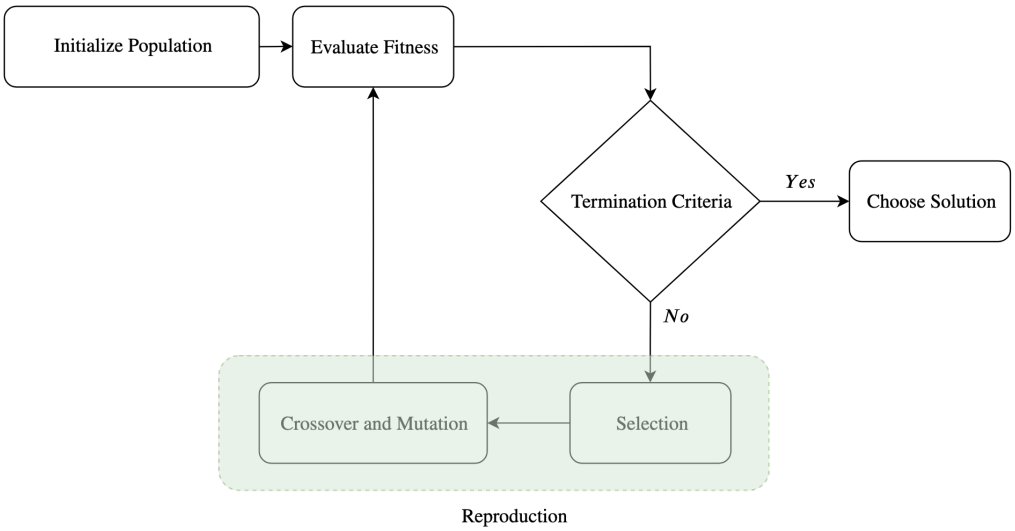


Figure 2.3: Flowchart of the evolution process of genetic algorithm. Each box is a step in the evolution process which we perform on the population. Notice the green box around 'Crossover -and Mutation' and selection. This is the step where we apply genetic operators on individuals and select the next possible solutions for the next population via reproduction. The arrows marks the cycle of loop and each run is the next generation of evolution process. Termination criteria is either when the population does not improve significantly over time or even when the amount of certain generations has passed.

2.5 Genetic Operators

2.5.1 Selection

Selection is how the individuals within population are competiting towards another to reproduce the next offspring for the next population. Most common approach is by selecting the fittest individuals as these are considered the best current solutions and let them pass the genes onto the next generation. This is known as elitism and will be discussed further below.

During selection process one might be attempted to only select the fittest candidates in the population, however choosing the fittest candidates might risk getting premature convergence in local optima [50] and is an undesirable condition. By having weaker solutions can in some cases prevent the extremely fit solutions from taking over the population.

2.5.2 Elitism

The elitism selection guarantees the fittest candidates are mixed into the pool of population for the next generation and the weakest one may risk being neglected away entirely, thus reducing the diversity. A potential downside is the risk of reaching local optima early on in the evolution process since the entire population is dominated entirely by only the fittest candidates. The benefit however

is its fast performance during simulation [35, p. 126] to reach convergence.

2.5.3 Roulette Wheel Selection

Roulette wheel selection is also known as fitness proportionate selection for selecting potential candidates for recombination. By applying fitness function on individuals we receive fitness for every candidates in the population. First we sum up the total fitness for all the candidates and divide every candidate on the summed fitness as shown in equation 2.10. After dividing we get probability for selection between $[0, 1]$ where the fittest candidate will have higher probability of being selected for recombination marked by P_i. Fitness for one candidate is noted by f_i. N is the population of the algorithm.

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.10}$$

Downside of using this selection is similar to elitism where one highly fit candidate can dominate over other weaker candidates if their fitness are much less, resulting in premature convergence. This is noticeable during the end of runs in GA.

Tournament Selection

Tournament selection is another genetic operator for selecting possible candidates from a population as offsprings. It usually involves running multiple tournaments for a given generation depending on how large the tournament size is. Selection works as follows, we first select tournament size which is called k. The size determines how many candidates are randomly selected from the population in order to compete towards one another in the tournament. There are two ways to determine the champion of the tournament. One is always selecting the fittest candidate. The second one is little more advanced.

First, we rank the fitness of the candidates in chronological order. Then we assign a random probability between $p \in [0, 1]$ and yields a scaled probability based on candidates rankings. Value from the first probability ensures that the fittest candidate always has higher probabilty of being elected than the lower ones. We choose another random probability number $r \in [0, 1]$.

$$\begin{aligned} P_{individual_1} &= p \\ P_{individual_2} &= p(1-p)^1 \\ P_{individual_3} &= p(1-p)^2 \\ &\vdots \\ P_{individual_k} &= p(1-p)^{k-1} \end{aligned} \tag{2.11}$$

2.5.4 Genetic Recombination

This is the recombination process between two individuals and and gives birth to offsprings which inherit traits from both of the individuals during the recombination. If we take human mating process, we inherit 50 % from one

parent's chromosome and the remaining 50 % from the other parent's chromosome. The recombination of these two chromosomes selects the best traits from chromosomes that increases the probability of survival given its surrounding environment [39]. However in the GA's one can decide how the individual is recombined depending on the crossover method we select. In this section it will walk through three of the most common crossovers in GA. There exists variety of other crossovers in GA, but the thesis will not cover them. Umbarkar and Sheth [60] did thorough research and review for many of the crossover operators and discussed their advantage and disadvantages. Operators are dependent on the encoding type and as such is the major criteria for selecting the GA operator.

2.5.5 Uniform Crossover

Given a string of genes, we assign an equal probability of selecting a gene from both parents onto the next one with ramdonly chosen crosspover point. The equal probability is fixed rating in which we can determine and select on beforehand on the GA. However the rating, P, has to be $\in$ [0,1]. Figure 2.4 shows the simple uniform crossover with $P = 50$ % as fixed. Notice here, we essentially inherit approximately half of the gene from parent 1 and the other half from the parent 2. Uniform crossover does not generally have to be $P = 50$ % but could for instance be $P_1 = 70$ % from parent 1 and $P_2 = 30$ % from parent 2. The most common is $P = 50$ % because we have more variety from both parents, and that results in more genetic diversity which allows for more exploration or exploitation or maybe even both [60].

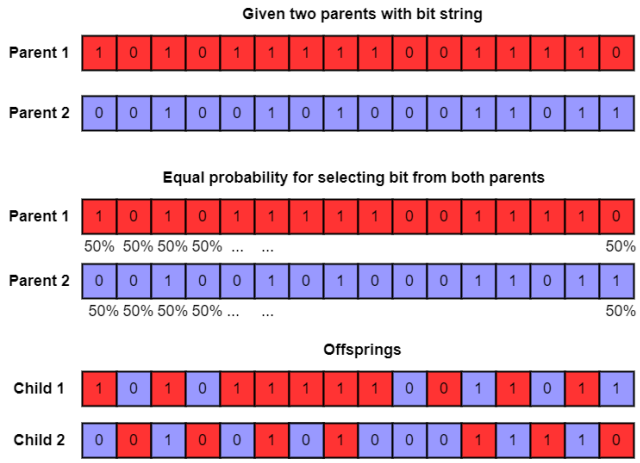


Figure 2.4: Genetic operator with Uniform-Crossover between two parents with P = 50%.

2.5.6 One-Point Crossover

One-point crossover is one of the most common and simplest genetic operator in GA. It uses single point in parents genetic string and then combine the parents at the same crossover-point to create offsprings. Figure 2.5 shows how the

genetic operator works. The green stapled line is the crossover-point for both of the parents. It does not necessarily have to start on the same index on the genetic string. It can be chosen at random where the crossover-point should be placed during recombination step. Red and purple shows the two respective parent's genes and how the children inherit their genes.

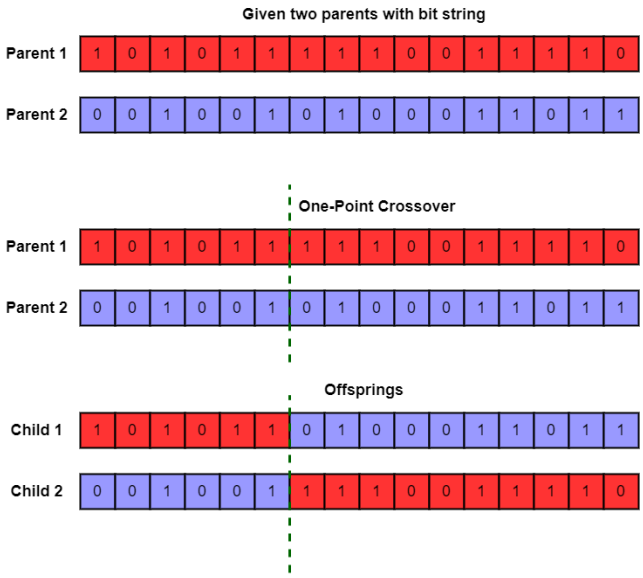


Figure 2.5: Genetic operator with One-Point Crossover between two parents.

2.5.7 Two-Point Crossover

Similar to One-Point Crossover in previous section 2.5.6. Difference is two crossover-points on both of the parents and then combine the parents to create offsprings with more variation in gene.

Given One-point Crossover and Two-point Crossover, it can also be extended it to K-Point Crossover. Select any arbitary K crossover-points within parent's string. This crossover provide great recombination of both parents to create the offsprings yielding more varied genes in the offsprings. A potential downside is risk of modifying the gene, losing the fittest candidate.

2.5.8 Genetic Mutation

Mutation process is the next genetic operator after recombination and is applied on every offsprings in the population. During this process, every offsprings have a probability of performing random mutation on their own gene. Similar to recombination, the mutation is dependent on the encoding type of the gene and as such it cannot use the same mutation on every gene. Getting an understanding of mutation, the example of using bit-flip will be used and explained in section 2.5.9. The actual mutation used, will be explained in section 2.5.10.

Role of mutation is to prevent the loss of diversity during GA evolution [35, p. 23]. This means we may reach towards the same solution or similar solutions

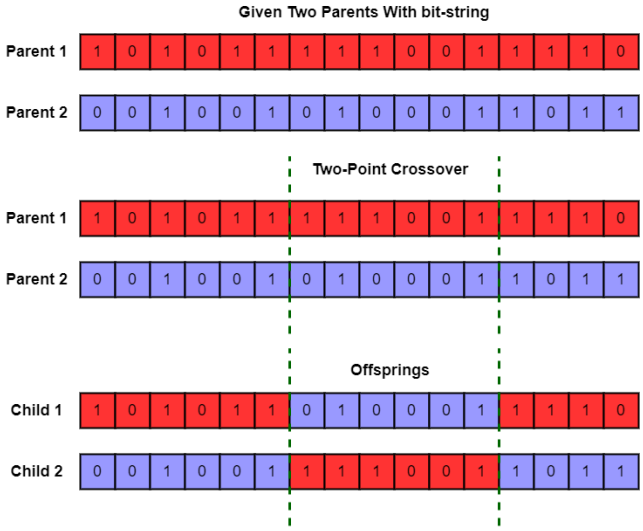


Figure 2.6: Genetic operator with Two-Point Crossover between two parents.

every run of GA, but however with mutation we add a probability of performing a random mutation on random gene after recombination. In this way, ensure the fixation of common gene is reduced. Usually, the mutation probability is low,$\leq$ 5, a common constant for most GA. Reason for low probability is to prevent losing fit candidates in the population. With high mutation probability it may risk the mutated gene to yield worse fitness than it was previous because the change might have changed the an important gene that contributed to most of the overall fitness [16]. Mutation allows for further exploration covering more search space while maintaining the current fit candidates. During the end where the GA algorithm might converge, will we notice that the mutation does not affect the overall population because as we the algorithm is converging.

2.5.9 Mutation: Bit-Flip

Bit-flip mutation is flipping one random bit in the gene to either from 0 to 1 or 1 to 0. Given an offspring with sequence of binary gene with length L, we assign probability of randomly flipping at one index in the sequence with $p = \frac{1}{L}$ where p is probability. Thus every bit-position has equal probability of getting randomly switched. Illustration of random bit-flip can be seen on figure 2.7. Following up on the figure, the length of the gene is 16 and shows us three random bit-flips.

2.5.10 Mutation: Gaussian

Mutation Gaussian is mutation operator designed for encoded genes operating with integer or floating number[60] rather than bits-string as previously explained in section 2.5.9.

For the mutation the only two parameters to select are the values for μ and σ and during the mutation it samples from the distribution. As such, given

Figure 2.7: Mutation with bit-flip

a vector of length L with random floating numbers, similar to bit-flip, add a probability of performing mutation at one index in the gene expressed by $p = \frac{1}{L}$ where p is probability. Assuming further that at least one mutation or more occurs we take that index position of that gene and we sample from $\mathcal{N}(\mu, \sigma^2)$. Our new value in the gene becomes

$$\hat{X} = X + \mathcal{N}(\mu, \sigma^2)$$

where X is the original value in the gene at the index position and $\hat{X}$ is the mutated value.

2.5.11 Evalutation of GA

From section 2.4.2 it is explained that the GA is stochastic. The evaluation requires multiple runs with the same parameters to get good estimation of performance [14, p. 126]. There are three methods used to evaluate the performance,

- Sucessrate (SR)
- Mean best fitness(MBF)
- Average number of evaluations to a solution (AES)

The used method for this thesis is the second one, mean best fitness. The reason is it is applicable for most of the GA and suits best.

2.6 Robot Simulation

A simulation is estimation of a model or system by solving analytic equations. [44]. It attempts to reproduce same behaviour of a mathematical model and to estimate the performance of systems/models of analytical solutions or methods [3]. A mathematical model is heavily influenced by large amount of data and using predictive simulation from that data one can approximate a real-world empirical system. Simulation runs and visualizes, animate and explore the temporal behaviour and the benefit of computer simulation is to gain insight of the proposed model and enhanced understanding of a problem.

2.6.1 Virtual Robotic Experimentation Platform

Virtual Robotic Experimentation Platform abbreviated V-rep, is developed by Coppelia Robotics [51] and is commercial robot simulator which offers great functionalities for multiple programming languages. V-rep provides a simple user-friendly GUI which allows for click and drag onto the environment. Common robots are also available in the local default library e.g UR5 manipulator [49] and NAO-robot[48]. The official language of V-REP is .lua but as mentioned earlier it allows remote-communication with multiple programming language such as Python, Matlab, C++/C and etc. This flexibility is useful for robotic projects where one can simultaneously use other software tools to transfer data from one platform to another with relative ease.

2.6.2 V-REP Hierarchy

V-REP is hierarchical based which in definition means the structure of an object in V-rep's scene is divided into multiple levels in a tree-like form. For example, given the figure 2.8.

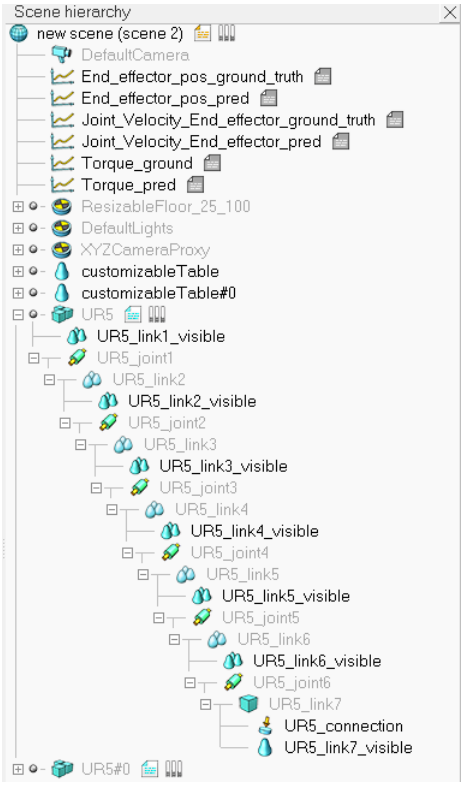


Figure 2.8: Figure displays the hiearchical structure of an V-REP scene. Observe the $UR5$-manipulator overall structure of UR5 arm with its joint and link's lengths. Principle behind such structure can be comparable with Denavit Hartenberg convention where we are transformning from one coordinate frame to another with respect to its previous coordinate frame.

2.6.3 V-REP versus Gazebo

Choosing a suitable simulator for a particular task is not trivial because each simulator simulates the environment different from each other [46]. They address the issue in terms of complexity, performance, programming language support and documentation. Lenka Pitonakova [46] did comparison and showed us V-rep simulator offers most features such as robot-libraries, scene editor, mesh-manipulation and supports remote connection for multiple programming langages with an active forum and documentation. In terms of performace it uses CPU most efficiently by spawning more threads whenever possible, utilizing whole of CPUs capability for simulation. Gazebo demands more than 50% CPU compared to V-REP and hence requires more powerful hardware for larger simulations [40]. Authors from [38] summarizes the key differences where largest difference lies in sensors and ROS-integration where Gazebo definitely has an advantage over V-REP.

Another downside with V-REP is performance and efficiency when simulating multiple robots especially with a large area. Lenka Pitonakova also investigated simulations of varying robots and scene-sizes and the results concluded that V-REP struggled with real-time simulation and used more memory and hence was not feasible while Gazebo's performance was affected only by small margin and that is despite V-REP's full utilization of CPU-cores.

2.6.4 Which physics engine is recommended in V-rep?

Before using V-REP simulation a physics engine must be determined because V-rep offers four physics engine to select from. These are Bullet v2.78, Bullet v2.83, Open Dynamics Engine(ODE) and Vortex. However Vortex is not free in the educational version of V-REP and will be excluded from comparison. So the question still remains which one do to choose from for our task and simulation? Table 2.1 illustrates that simulating for larger and more robots decreases the simulation time substantially.

From analyzation from table 2.1, the comparison between the two respective physics engine is neglible if one only wants to simulate in smaller environment. Later, in the experiments and results chapter the specified physics engine used will be given.

2.6.5 Remote API

V-REP offers remote client API with other programming languages, meaing other API can receive and send commands to V-reps environment using another language than the default language in V-REP as explained in section 2.6.1. Since remote client API requires one to transmit data from one platform to another, there is the issue of ensuring whether the simulation is to be run asynchronous or synchronous. V-rep provides the option to select which mode, but the entire environment must be setup beforehand.

There are two ways V-rep handles asynchronous environment. First method is where we are continously sending commands onto V-rep environment, but the instructions can instantaneously be performed independent of previous instructions. Problem could be when we send sequence-based instructions which varies over time and the instructions overlap. Here the sequence must be given

Comparison between Bullet and ODE physics engine		
	Bullet v2.78+Bullet v2.83	Open Dynamics Engine (ODE)
1 robot + Small scene	R $\geq$ 1 Usage of CPU: 180 %	R $\geq$ 1.0 Usage of CPU: 190%
5 robot + Small scene	R $\geq$ 0.52 Usage of CPU: 395 %	R $\geq$ 0.37 Usage of CPU: 395%
10 robot + Small scene	R $\geq$ 0.11 Usage of CPU: 400 %	R $\geq$ 0.099 Usage of CPU: 400%
50 robot + Small scene	Not Feasible	Not Feasible
1 robot + Large scene	R $\geq$ 0.96 Usage of CPU: 200 %	R $\geq$ 0.53 Usage of CPU: 190%
5 robot + Large scene	R $\geq$ 0.18 Usage of CPU: 400 %	R $\geq$ 0.1 Usage of CPU: 395%
10 robot + Large scene	R $\geq$ 0.052 Usage of CPU: 400 %	R $\geq$ 0.036 Usage of CPU: 400%
50 robot + Large scene	Not Feasible	Not Feasible

Table 2.1: Table displays the performance during simulation where if R $\geq$ 1, simulation could run faster than real-time. Observing the the amount of robots one can notice the significant performance reduction by only adding few additional robots. CPU-usage higher than 100 % includes the core of the CPU. 400 % means the CPU is using the 4-cores in CPU from [46].

at the correct timestep otherwise the simulation will not simulate the desired movement for example trajectory. Benefit of method one is if we are sending the same instruction repeatedly but is not time-dependent. Looking at figure 2.9 we observe the overall process of V-rep's handling of the following method.

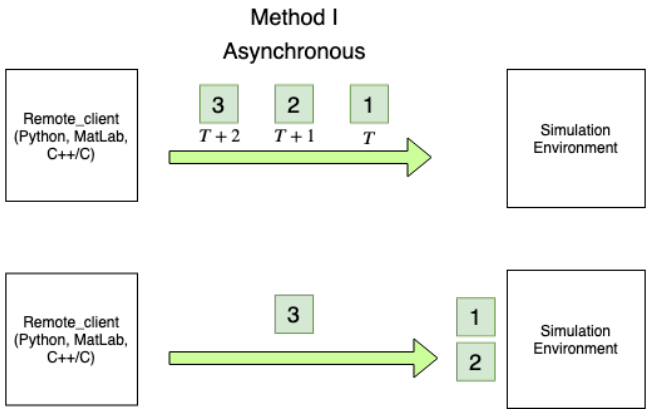


Figure 2.9: The figure shows simple illustration of the simplest asynchronous method. The green boxes are the instructions or command we are sending to the environment but as long as instructions are reaching the environment, V-rep will run the instruction and indicated by the instruction 1 and 2. These two instructions will overlap and as result V-rep will execute the first

The second method ensures the first instruction is applied and taken into action before executing the next instruction in the sequence. Even though the second method seem to have synchronous traits it does not. Underlying issue with the second method is that V-rep will perform the instructions but processing it and sending it to the environment takes time. The required completion time for the instruction is not something we can decide and we will call this unwanted time delay for τ. If we are looking at figure 2.10 one will see how V-rep sends back the additional completion-time of the execution and how it affects the next instruction. If we accumulate all of the τ for every timestep and we will notice considerable decrease in performance during simulation run.

2.6.6 Asynchronous Method I versus Asynchronous Method II

Which asynchronous method to use, depends on the task of the application. If we are dealing with time-dependent data it should be method two otherwise we should select method one as it is faster to process than method two. Since motion-capture is depenent of timesteps i.e. marker postions, marker velocities, marker acceleration we want to ensure the motion is performed exactly at the specified timestamp.

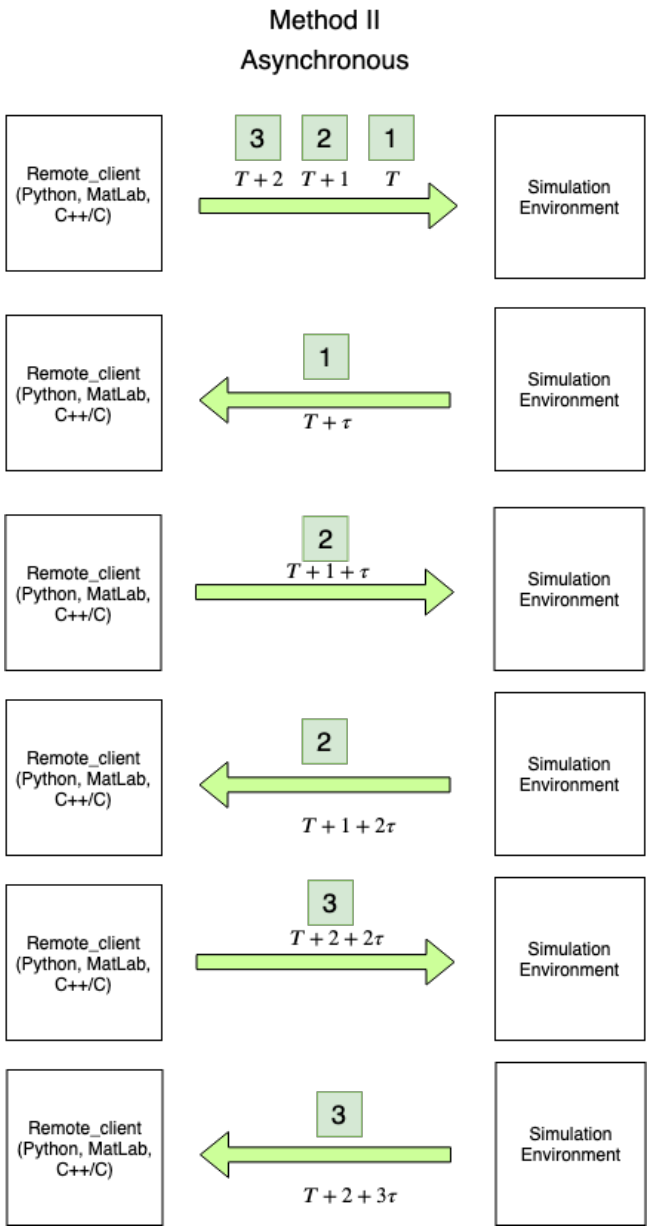


Figure 2.10: The figure shows how V-rep's method two handles the data to be sent over to the simulation environment from remote client. The green boxes marks the instructions in chronological order. First, the environment simulates the first instruction it receives and this process is finished before the next instruction is processed. The time upon completion is indicated by τ and is the value included on the arrow back from environment to client. Arrow back is V-rep's verification of completed instruction.

Chapter 3

Motion Capture Data

3.1 Motion Capture

Motion Capture (moCap) is the process of recording sequential movement of objects or person.The idea of motion capturing originates from the field of gait analysis where movement patterns from animals and humans. It is often used to record only the person's movement and not by the person's visual appearance. Utilizing multiple cameras from multiple angles one can recreate and calculate the 3D-positions of each marker's movement over many videoframes. The higher the sampling-rate between each consecutive frames, the faster it can capture and accurately estimate the original movement [36]. As such, by using motion-Capture system it can capture both complex and simple motion with low latency in real-time.

3.1.1 Optical Marker-based Vicon System

Vicon motion capture system is one of the most commercially succesful optical system for accurately recording 3D movement[61]. Vicon is marker-based which means it records the blue markers in the entire motion. Looking at figure 3.1, notice the markers and the grey triangles are the camera's field of view. Each marker is transmitted to computer and then processed to 3D-model.

3.2 Motion Capture Dataset - Hochschule Der Medien

The Hochschule Der Medien(HDM05) motion capture dataset provides more than three hours of systemtically and commented motion capture in **.C3D**, **.ASF** and **.AMC** format and includes up to 70 motion classes performed by various actors of various body sizes. The number 05 behind the name HDM, states the year it was recorded. It was also processed and shot at Stuttgart, Germany at Hochschule Der Medien [10] in 120 FPS. They used Marker-based Vicon system to capture the different actor's movement respectively.

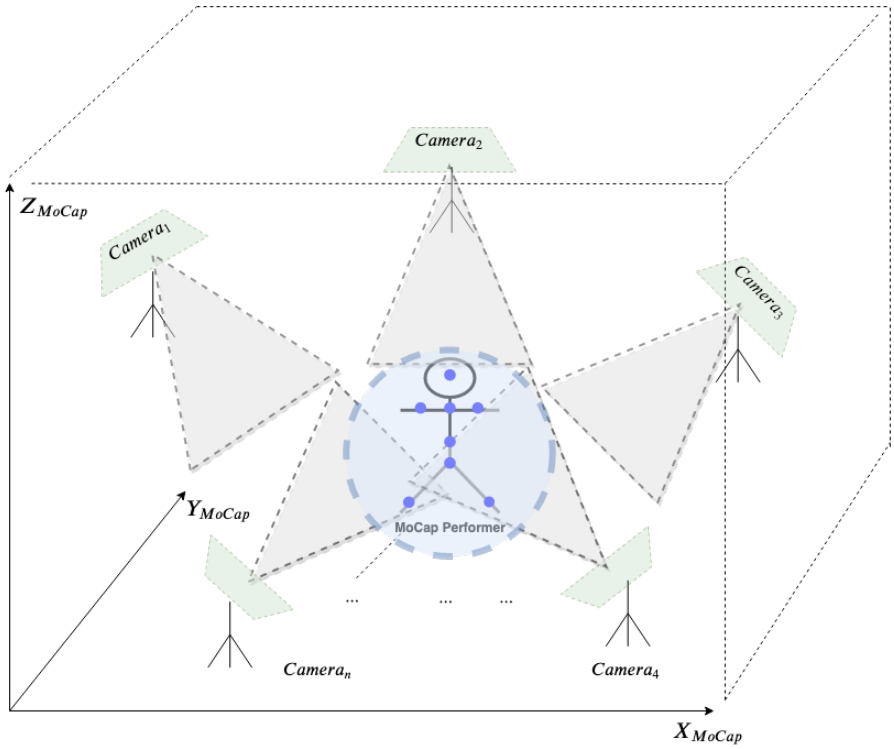


Figure 3.1: Simple illustration of Vicon Motion Capture system displays how the cameras are contributing of capturing the MoCap-actor and notice the axis X_{MoCap}, Y_{MoCap}, Z_{MoCap}. Every point, $p(X_{MoCap}, Y_{MoCap}, Z_{MoCap})$ is with reference to MoCap coordinate-frame. The blue area(circle) marks the captured volume space.

3.2.1 MoCap-Markers

The dataset provided was captured by non-professional actors of various sizes. Every marker represents particular part of the body and could be interpreted as joints on the human-body. The markers are recorded and estimated in 3D-space using triangulation[1]. The more markers and cameras a moCap optical system has, the more accurate can it recreate the original motion in computer processing software because it has more availability in spatio-temporal data [4]. However, by using more markers it increases the probability of getting missing markers due to multiple reasons [21] known as marker-occlusion. The fault could lie in the equipment such as hardware equipment or software applications not able to properly track the markers or the markers are overlapping each other in way the camera believes it is the same marker. Section 3.2.3 will discuss in further details how to deal with this problem and why it is relevant for HDM05-dataset.

[1]Triangulation is the process of determining the location of a point by forming triangles to the pont from known points.

3.2.2 Feature Extraction

The motion capture data contains many features but in order to extract features, a framework called MoCap Toolbox is used, developed by Toiviainen and Burger in Matlab[57]. The following toolbox contains many functions such as extracting the Cartesian Coordinates, marker names, joint-angles between the markers and many more. To view the entire list of features, have a look at at well-written documentation made by the authors[58]. However after 2015, the toolbox has remained outdated and has not received any new updates, bug-fixing or functions. Nymoen's extended version of the original author's MoCap toolbox improved it further by implementing new features such as animation, visualisation, plotting, statistical analysis and more [41].

3.2.3 Video Frames and Missing Markers

Each motion during the recording session is captured in video frames and the cameras from HDM05 are captured120 fps. That means for each second, 120 frames are stored and the duration of seconds can vary depending on how long the actor performs certain motion multiple times.

Some common known issue with MoCap data are missing markers during the recording session as briefly mentioned in 3.2.1. If one actor has 44 markers while another actor has 41 marker it creates a problem because the dataset-format should be equal for every actor in the dataset. There are many workarounds to this problem. One is by removing the actors with the different markers or two is by equalizing the markers to default size on from MoCap-actor or third perform linear interpolation to estimate where the missing marker could have been in the sequence at the given timestep. The last one is removing the frames at the given timestep at the time of marker occlusion occured.

3.2.4 Prepocessing Motion-Capture Dataset

Before using the data acquired from motion-capture dataset the MoCap data must be processed using the MoCap Toolbox. The advantage of MoCap Toolbox is its handling of multiple filetypes as explained in section 3.2. Every MoCap-filetype has their own structure and format and MoCap-Toolbox handles such filetypes effectively and in an organized manner.

3.2.5 Marker Position

The MoCap-markers are in an environment which has a local coordinate frame. Every marker's are Cartesian coordinates with reference to the local coordinates with the unit scale mm. As such the position of every markers can be written as,

$$p(x, y, z) = [p_1(X_1, Y_1, Z_1), p_2(X_2, Y_2, Z_2), \ldots, p_n(X_n, Y_n, Z_n)] \tag{3.1}$$

where $n = 1, 2, \ldots, n_{markers}$ and $p_1(X_1, Y_1, Z_1)$ is 3×1 vector and the length of $p(x, y, z)$ is $3 * n_{markers}$

3.2.6 Marker Velocity

To calculate the velocity of the markers one can estimate time-derivative of MoCap-data using the following method. By using the difference between two successive frames and butterworth smoothing filter [12] one can estimate the velocity of every markers. Velocity measures the displacement within a time-interval.

$$\dot{p}(x,y,z) = [\dot{p}_1(X_1,Y_1,Z_1), \dot{p}_2(X_2,Y_2,Z_2), \ldots, \dot{p}_n(X_n,Y_n,Z_n)] \tag{3.2}$$

where $n = 1, 2, \ldots, n_{markers}$ and $\dot{p}(x,y,z)$ is time-derivative of $p(x,y,z)$ and $\dot{p}_1(X_1,Y_1,Z_1)$ is 3×1 vector and the length of $\dot{p}(x,y,z)$ is $3 * n_{markers}$. The following function in Matlab Toolbox that performs this is

```
Velocity = mcarrayFunc(MoCap_data, 'mctimeder')
```

3.2.7 Marker Acceleration

Acceleration is the rate of change of the velocity with respect to time. To find the acceleration of a given marker, one can perform time-derivative of velocity of the marker using Savitzky-Golay Filter [55]. This filter is used to smooth data without distorting the signal by using convolution. During the convolution process it tries fitting sub-sets of adjacent datapoint with a low-degree polynomial using linear least squares(LLS)[30].

$$\ddot{p}(x,y,z) = [\ddot{p}_1(X_1,Y_1,Z_1), \ddot{p}_2(X_2,Y_2,Z_2), \ldots, \ddot{p}_n(X_n,Y_n,Z_n)] \tag{3.3}$$

where $n = 1, 2, \ldots, n_{markers}$ and $\ddot{p}(x,y,z)$ is time-derivative of $\dot{p}(x,y,z)$ and $\ddot{p}_1(X_1,Y_1,Z_1)$ is 3×1 vector and the length of $\ddot{p}(x,y,z)$ is $3 * n_{markers}$. As similar in section 3.2 the function is almost identical, the main difference is the added parameter $'2'$ which is the second time-derivative of velocity.

```
Acceleration = mcarrayFunc(MoCap_data, 'mctimeder',2)
```

3.2.8 Transformation of moCap-markers - Marker Reduction

The markers received from the HDM05 dataset are raw and unprocessed. This may include multiple moCap-markers not corresponding exactly to human anatomy as the moCap skeleton is different. Figure 3.3 displays all of the markers and the ideal goal is to reduce the markers to represent actual joints or skeleton of the human body.

In order to reduce the amount of markers, first grouptogether relevant markers into one common marker and let that marker represent the designated joint. Now that the markers are grouped together, a method is required to find the new coordinates to represent it. A common method is to take the Cartesian coordinates for every markers and sum them up together in X, Y and Z axis. At last step the average of the Cartesian coordinates in X,Y and Z-axis is calculated for the grouped markers. Figure 3.2 shows the illustration of the marker-reduction using four markers.

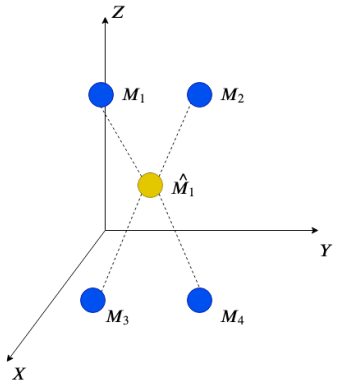


Figure 3.2: Figure displays four blue markers M_1, M_2, M_3 and M_4 are transformed into new yellow marker, $\hat{M}_1$, where it represents the four blue markers to be represented as joint.

Given marker positions as described in section 3.2.5 and suppose one are given set of grouped markers and calculating the Cartesian position for $\hat{M}_1$ one get the general formulation of calculating the new Cartesian point.

$$\begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \end{bmatrix} = \begin{bmatrix} \frac{\sum_{i=1}^{n} X_i}{n} \\ \frac{\sum_{i=1}^{n} Y_i}{n} \\ \frac{\sum_{i=1}^{n} Z_i}{n} \end{bmatrix} \tag{3.4}$$

where $n = 1, 2, ..., n_{markers}$ and below one can take example from figure 3.2 and find its new coordinates and as one may notice.

$$\begin{bmatrix} \hat{M}_{1x} \\ \hat{M}_{1y} \\ \hat{M}_{1z} \end{bmatrix} = \begin{bmatrix} \frac{M_{1x}+M_{2x}+M_{3x}+M_{4x}}{4} \\ \frac{M_{1y}+M_{2y}+M_{3y}+M_{4y}}{4} \\ \frac{M_{1z}+M_{2z}+M_{3z}+M_{4z}}{4} \end{bmatrix} \tag{3.5}$$

From equation 3.5 yields the new marker $\hat{M}_i$. Suppose there are many sets of grouped markers and one wants to find the new coordinates, it can simply be done using the same formula for all the grouped sets and as results from transformation we get reduced markers and the markers forms reduced version of MoCap-skeleton

3.2.9 44 MoCap-Markers

The 44-markers is the raw moCap-skeleton from HDM05-dataset and figure3.3 visualizes all of 44-markers. Notice how compact the markers are and how the skeleton is asymmetrical especially at the torso region.

3.2.10 7, 20, 28 MoCap-Markers

From the 44-markers moCap-markers, marker reduction is performed on it using the method from section 3.2.8 into three groups. They are 7, 20 and 28 markers and figures 3.4, 3.5 and 3.6 shows the newly generated moCap-skeleton that is symmetrical and body layout representing the human skeleton.

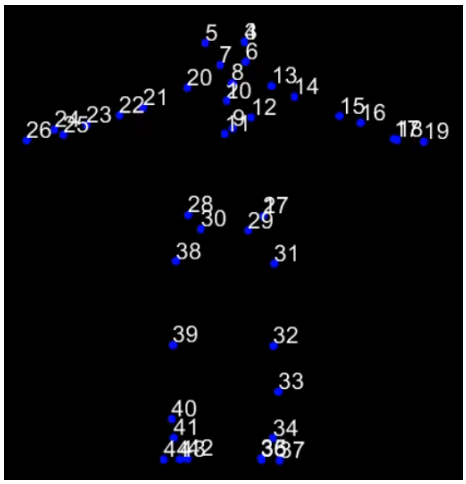


Figure 3.3: Motion Capture skeleton with 44 Markers

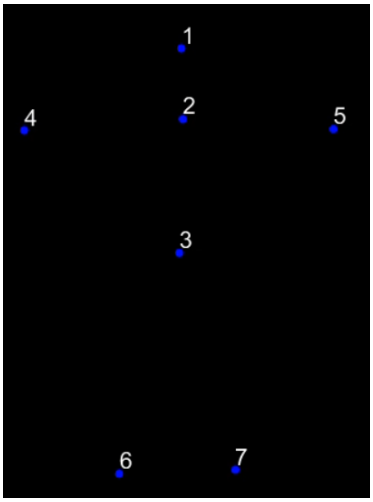


Figure 3.4: Motion Capture skeleton with 7 Markers

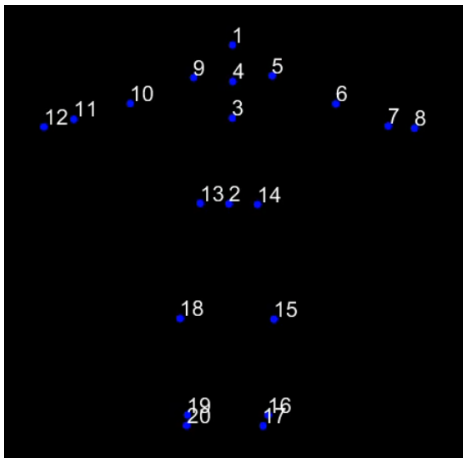


Figure 3.5: Motion Capture skeleton with 20 Markers

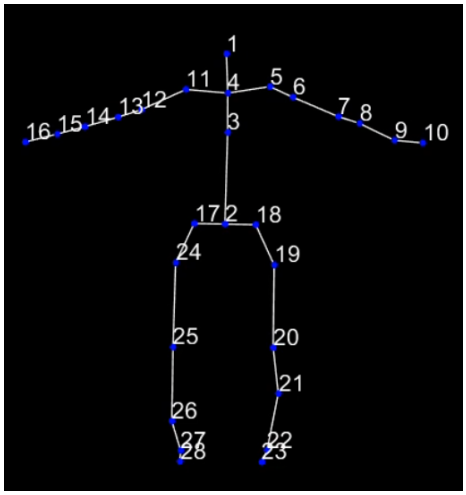


Figure 3.6: Motion Capture skeleton with 28 Markers

3.2.11 Quantity of Motion

Quantity of Motion can be summarized as summing up different MoCap-data features such as position, marker positions, marker velocity and etc. The idea is to analyze the overall movement of the motion and decide whether the information acquired could prove useful. For example, given a walking motion from MoCap and summing up the velocity at every frame t. A moment is taken to observe if there is a pattern in the motion in which may be interesting to extract from or could prove useful for getting an insight of different motions. For example, walking motion might have repetitive peaks at frame t and $t + t_{next}$ where t_{next} is future timestep.

Summing up different elements of MoCap-data is useful especially if one has multiple markers in which one want to observe the overall periodic response of a motion. Visualizing all markers into one plot might difficult to interpret and cause confusion, especially if there are more than 10 different graphs in one plot. In general the features from section 3.2.5, 3.2.6 and 3.2.7 are commonly known, so it is simple to sum up each of them.

$$Feature_{sum} = \sum_{m=1}^{n} P_m(x, y, z) \tag{3.6}$$

where $Feature_{sum}$ is 3×1 vector, $n = 1, 2, \ldots, n_{markers}$ and each component in $Feature_{sum}$ is the sum in X,Y,Z-axes

$$\begin{aligned} Absolute_{sum} &= \sum_{m=1}^{n} |P_m(x, y, z)| \\ &= \sum_{m=1}^{n} \sqrt{\sum_{i=1}^{3} (P_m(i)^2)} \end{aligned} \tag{3.7}$$

where $Absolute_{sum}$ is scalar, $n = 1, 2, \ldots, n_{markers}$, m is marker-index from the MoCap-data and $P_m(i)$ is value in the X,Y,Z-axes.

3.2.12 Kinetic Energy

Human body assert forces and energy when performing a movement and using the estimated velocity for every markers and the mass of the human MoCap one can determine the kinetic energy of selected body parts or the entire body. Using the basic equation of kinetic energy which is formulated, yields

$$K = \frac{1}{2} m v^2 + \frac{1}{2} I \omega^2 \tag{3.8}$$

where K is measured in Joule, m is mass, v is velocity, I is inertia and ω is angular velocity.

3.2.13 Mass and Velocity

Problem with HDM05 documentation is its lack specifying the actor's height and weight. As a consequence it becomes problematic when one wants to measure the kinetic energy using the equation 3.8 which measures the general kinetic energy. Since the marker's velocity is known, the only remaining parameter

needed is the the weight of the actor but it is not specified in the documentation. Using the average weight of a male in Germany, one can then calculate the kinetic energy for every markers. A german male weighs on average 88.8 Kilograms (kg) [5].

3.2.14 Weight of Body Parts

The general weight of the human body is not uniformly-distributed as the weight is distributed to differently arms, legs, head and more. Rudolfs et al. [9] and [62] did thorough research and managed to measure the weight percentage of a human body of different body parts. The thesis measures the kinetic energy with the weight-percentage and based on the percentage from these two references throughout the whole thesis.

3.2.15 Angles Between Two Markers

Now, calculating the angles between the markers requires two markers which we can perform as follows. Given two markers M_1 and M_2 ...

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \end{bmatrix} \tag{3.9}$$

Then we find the difference in length between each of the respective components,

$$\begin{bmatrix} X_{diff} \\ Y_{diff} \\ Z_{diff} \end{bmatrix} = \begin{bmatrix} X_2 - X_1 \\ Y_2 - Y_1 \\ Z_2 - Z_1 \end{bmatrix} \tag{3.10}$$

$$\begin{bmatrix} \theta_X \\ \theta_Y \\ \theta_Z \end{bmatrix} = \begin{bmatrix} \arctan(\frac{X_{diff}}{{Y_{diff}}^2 + {Z_{diff}}^2 + \epsilon}) \\ \arctan(\frac{Y_{diff}}{{X_{diff}}^2 + {Z_{diff}}^2 + \epsilon}) \\ \arctan(\frac{Z_{diff}}{{Y_{diff}}^2 + {X_{diff}}^2 + \epsilon}) \end{bmatrix} \tag{3.11}$$

where ϵ = 2.2204e-16 which is approximately zero and the added element ϵ prevents the function from returning an undefined value from dividing on zero on the arctan-function. The MoCap-Toolbox has the feature implemented and can be called using the following function below

```
joint_tmp = mcsegmangle(d, marker_1, marker_2);
```

Chapter 4

Method and Implementation

4.1 Proposed Method

A robot may have multiple degrees of freedom(DOF) and trying to transfer the motion-capture joint angles directly onto the robot may yield poor results because joint angles are directly affected by height, arm-length, and foot-length of the actor. When one human acts, many of the joints are contributing to perform that task. For example, when a human grabs an object, he can grab it in many different ways and that means joint angles can also be different. This idea could also be extracted onto the robot configuration. This can be solved by solving a set of equations as in (4.1) where the θ are the moCap-angles and the w is the weights. The length of n is variable and can be any arbitrary length dependent on the number of moCap-angles selected

$$\begin{aligned} \hat{\theta}_1 &= w_{11}\theta_1 + w_{12}\theta_2 + \cdots + w_{1n}\theta_n \\ \hat{\theta}_2 &= w_{21}\theta_1 + w_{22}\theta_2 + \cdots + w_{2n}\theta_n \\ &\vdots \\ \hat{\theta}_m &= w_{m1}\theta_1 + w_{m2}\theta_2 + \cdots + w_{mn}\theta_n \end{aligned} \tag{4.1}$$

where m represents robot's DOF and n the amount of joints for moCap-angles.

The purpose of the weights are uniquely defined for one actor, such that by only solving W in (4.2) we can then apply any arbitrary motion capture angles from one person and the robot manipulator will then attempt to recreate the same motion based on the angles from the original human actor. The newly solved θ-angles are the mapping from human to robotic configurations. Thus, by solving and optimizing the weight matrix we do not have to scale the joint angles for any other motion capture. the weight-matrix will implicitly describe the overall relation between human actor and robotic manipulator.

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \tag{4.2}$$

where W $\in \mathbb{R}^{m \times n}$ and m-column represents robot's DOF and n-row represents moCap-angles.

Simultanously, we can form the motion-capture angles, but in vector form such as

$$\theta_{mocap} = \begin{bmatrix} \theta_{mocap_1} & \theta_{mocap_2} & \dots & \theta_{mocap_n} \end{bmatrix} \tag{4.3}$$

and by performing matrix multiplication $\hat{\theta} = W\theta^T_{mocap}$ the results are equal to (4.1) which we are intending to solve and optimize.

$$\hat{\theta} = W\theta^T = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} \theta_{mocap1} & \theta_{mocap2} & \dots & \theta_{mocap_n} \end{bmatrix}^T \tag{4.4}$$

4.1.1 Weights Initialization

Initializing the weights for the weight-matrix in 4.2 can be done in multiple ways and choosing which method that suits best for our problem is not trivial. The most important step is to not assign all of the weight-components in W to equal value. The reason is during recombination operation in GA when the next offspring is being created, it will yield the same genes.

First initialize every element from 4.2 and then assign random number ranging from $[w_{lower}, w_{upper}]$ where w_{lower} is a negative floating number and w_{upper} is a positive floating number.

4.1.2 MoCap Coordinate frame to Robotic Coordinate frame

3D-points in the robotic coordinate frame is not equavalent to the moCap coordinate frame because every 3D-point in the robotic coordinate frame is with respect to its base frame. Hence, before working on the objective function, the markers must be transformed to the right coordinate frame to avoid scaling issues. Figure 4.1 displays the two respective coordinate frames.

In figure 4.2 transforms the moCap-points from motion-capture to the robotic coordinate frame. The green arrow displays the direction in which the transformation is performed from. There are multiple ways to transform from one coordinate frame to another, but here the root from the motion-capture is set to be the base of the robot coordinate frame. The root-marker from the moCap can be any marker points can be determined and selected at will.

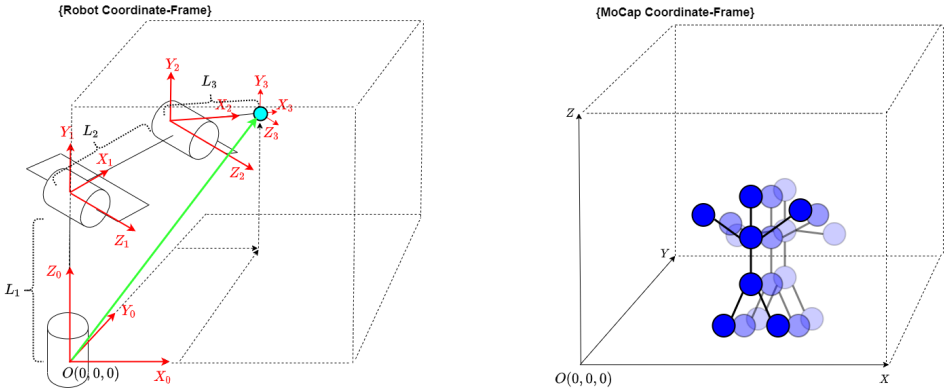


Figure 4.1: Two coordinates frames

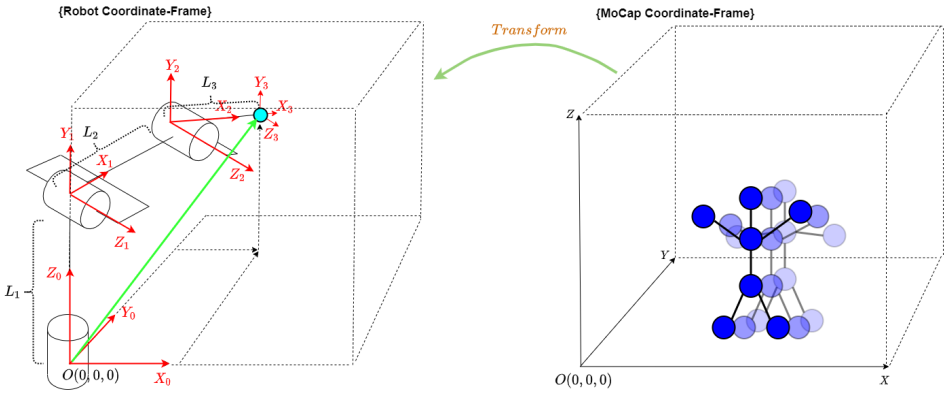


Figure 4.2: Transforming the points from human coordinate frame to robot coordinate frame

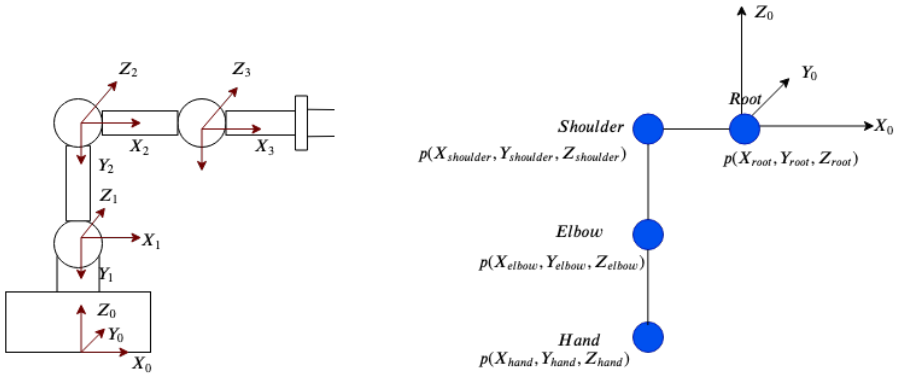


Figure 4.3: Robot manipulator(left) is in robotic space and human-markers(right) can be considered the moCap space.

4.1.3 Transforming from moCap-Coordinate frame to Robotic frame

Transforming from moCap-space to robotic space with respect to the base-frame of the robot manipulator can be done using homogenous transformation for every marker. The base-frame can be seen in fig 4.3 with frame $o_0x_0y_0z_0$, meaning the origin is $O(0,0,0)$. To transform from the moCap-frame to robotic coordinate frame requires at least two known 3D-point in the human coordinate. Reason for that is one of the two 3D-points become the root-marker and the other 3D-point position is with respect to root-marker. The root-marker is translated by its previous position in the 3D-space and becomes a point origin $O(0,0,0)$. Since the root-marker and base-frame have equal origin $O(0,0,0)$, this can be seen as the common-point for both.

As an example, extracting a small portion of human moCap-actor where an arm is chosen, this can be viewed on figure 4.3. From the figure the arm is represented by multiple moCap-markers and their position known from the moCap-dataset. Since there are more than two 3D-point, a root-marker is possible to select. Choosing the shoulder-marker as root-marker, then all of the new markers will be transformed with respect to the root-marker's origin. By subtracting the root-marker position for every marker on the human arm can be considered as translation with respect to root-marker.

$$T^B_{marker} = \begin{bmatrix} R & t_{root} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{marker} \\ 1 \end{bmatrix} \tag{4.5}$$

The homogenous transform can be seen in equation 4.5 where R $\in \mathbb{R}^{3\times3}$, is the rotation matrix, t $\in \mathbb{R}^{3\times1}$, is the old position of the root-marker and $P_{marker} \in \mathbb{R}^{3\times3}$, is the moCap-marker position in 3D. The equation 4.5 can be written as equation 4.6 with the shape of vectors and matrix.

$$\begin{aligned} T^B_{marker} = \begin{bmatrix} R & t_{root} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{marker} \\ 1 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & P_{x_{root}} \\ 0 & 1 & 0 & P_{y_{root}} \\ 0 & 0 & 1 & P_{z_{root}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{marker_x} \\ P_{marker_y} \\ P_{marker_z} \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} P_{x_{marker}} + P_{x_{root}} \\ P_{y_{marker}} + P_{y_{root}} \\ P_{z_{marker}} + P_{z_{root}} \\ 1 \end{bmatrix} \end{aligned} \tag{4.6}$$

Visualization of the process is displayed on figure 4.4, the blue points represents the original moCap-markers and the red points are the new transformed points. The green line is the distance or the total translation between the old and new markers. To differeniate them, the old markers are p_n and the new markers are $\hat{p_n}$.

4.2 Fitness Function

4.2.1 Euclidean Distance

Euclidean distance, also known as L2 norm, is the length between two points p and q connected by line segment. The distance is given by the Pythagorean

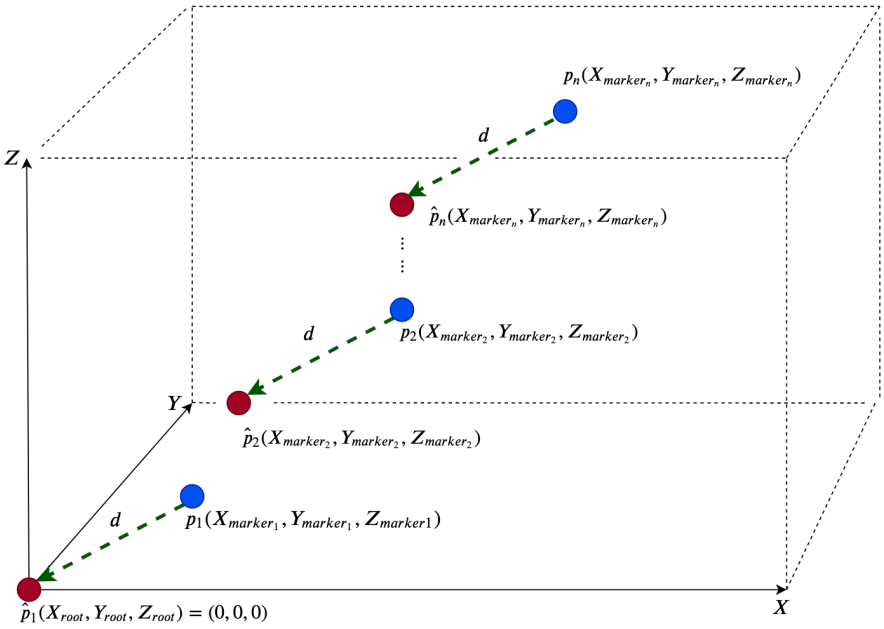


Figure 4.4: Every marker-positions is transformed into a new rescaled coordinate frame with origin $p(0, 0, 0)$

formula.

$$\begin{aligned} d(p,q) = d(q,p) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \end{aligned} \tag{4.7}$$

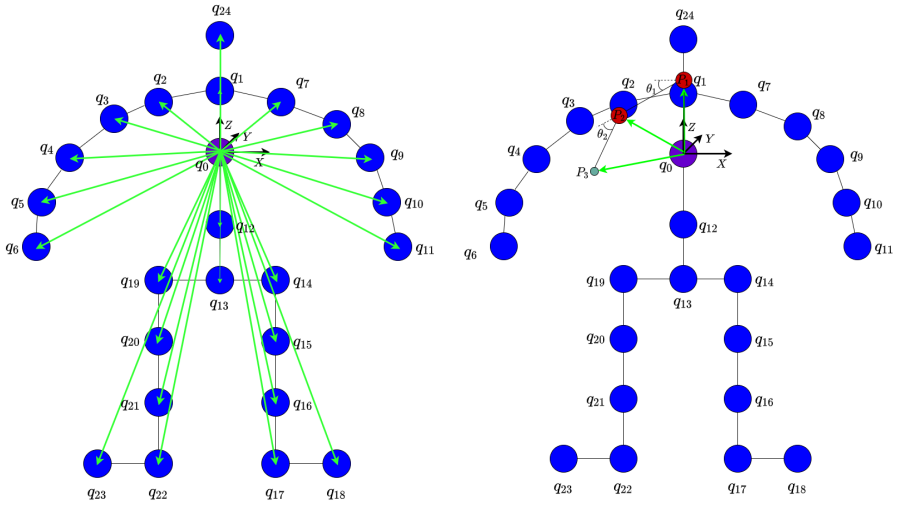
where $p \in \mathbb{R}^n$, $q \in \mathbb{R}^n$ and $n = 1, 2, \ldots$, dimensional space.

4.2.2 Every Joints' Distance Minimization (EJDM)

A distance matrix 4.8 is computed where each component inside the distance matrix represents the distance between the joint and the moCap-marker in 3D. This approach is inspired by Liarokapis et al. [29] where the distance is minimized between human arm and robotic manipulator, but the difference is they do not take into account multiple moCap-markers and only taking three points.

As mentioned, the distance matrix $Distance \in \mathbb{R}^{m \times n}$ is created with equivalant shape as the weight matrix in equation 4.2 and can seen in equation 4.8.

$$Distance = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix} \tag{4.8}$$



(a) (b)

Figure 4.5: figure 5.10a shows the translation for every marker with respect to the root-marker indicated by the purple marker q_0. Figure 4.5b on the right shows the translation of robotic manipulator's joint marked by red circles, p_i, where the base frame is placed on common marker indicated by the purple marker q_0. The green arrows on both figures are the translation-vector(s). The blue circles, q_i, are moCap-markers.

$$
\begin{array}{c}
\\
m \text{ Degrees-of-Freedom}
\end{array}
\begin{array}{c}
n \text{ moCap-marker} \\
\begin{bmatrix}
d_{11} & d_{12} & \cdots & d_{1n} \\
d_{21} & d_{22} & \cdots & d_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
d_{m1} & d_{m2} & \cdots & d_{mn}
\end{bmatrix}
\end{array}
$$

In order to get better understanding of distance matrix purpose, a few selected distance components will be selected and described further. For example, given d_{11}, d_{12} and d_{m2}, the first term d_{11} is the distance between joint 1 and moCap-marker 1 while the second term d_{12} is the distance from joint 1 to moCap-marker 2. The last component d_{m2} is the distance from the joint m in the robotic manipulator to moCap-marker 2. Every distance component, d_{mn} is calculated using the equation 4.9 and for each distance component in the distance matrix, the goal is to minimize the distance between every joint and moCap-marker the joint angles from equation 4.1. Figure 4.6 gives an illustration of the idea.

$$
\begin{aligned}
d_{m,n}(p,q) &= \sqrt{p_m^2 - q_n^2)} \\
&= \sqrt{(p_{m,x} - q_{n,x})^2 + (p_{m,y} - q_{n,y})^2 + (p_{m,z} - q_{n,z})^2}
\end{aligned}
\tag{4.9}
$$

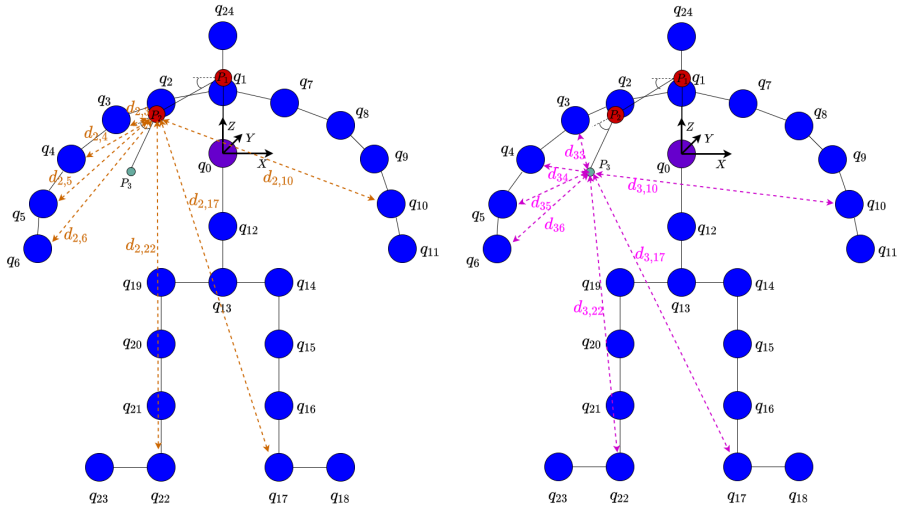
where $p_m \in \mathbb{R}^3$ is the manipulator joint position for joint m and $q_n \in \mathbb{R}^3$ is moCap-position for moCap-marker n. p_m is acquired from the Denavit-

Link	z_{i-1}	θ_{i-1}	x_i	α_i
1	z_0	θ_0	x_1	α_1
⋮	⋮	⋮	⋮	⋮
n	z_{n-1}	θ_{n-1}	x_n	α_n

Table 4.1: DH-table with the four parameters

Hartenberg transformation matrix, T_m^0 or more specific from forward-kinematic for the robotic manipulator. From the DH-table 4.1, the transformation matrices are found by just inserting the DH-parameters in 2.8 in background chapter for modeling a robot. q_n is extracted from the moCap-dataset.

Since the robot's position is dependent by the joint angles in 4.1, means the weights must be optimized to in order achieve lowest possible distance between moCap marker and robotic joint. The distance between p_m and q_n is calculated using Euclidean distance.



(a) (b)

Figure 4.6: 4.6a describes the distance from joint 2 (red) on the manipulator to moCap-markers (blue) indicated by the orange arrows. d_i, j is the distance component from eq. 4.9. To the right fig 4.6b shows the distance from the end-effector P_3 to moCap-markers(blue) indicated by the pink arrows. The purple marker is the root-marker for moCap and also the base-frame for the robotic manipulator and considered common point for both human and robot manipulator.

For the linear velocity, a velocity matrix. $Velocity \in \mathbb{R}^{m \times n}$ is created. It may be desirable to let the robotic manipulator move with similar motion speed as the moCap-actor. The same approach from distance matrix can be used here because when a human arm moves, it does not move one joint at the time, but rather multiple ones simultaneously each joint with their own velocity. For

example, consider a motion of throwing a ball with human arm as far as possible. The human arm is moving multiple joints at once, the shoulder, the elbow, the hand in order to exert enough force through speed. By moving one joint at the time will not yield the same result as the force exterted is not sufficient.

$$Velocity = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mn} \end{bmatrix} \tag{4.10}$$

The velocity for the joints can be achieved using the same weight-matrix from the moCap-angles and distance matrix. Thus, the velocity for every joint m in the robotic manipulator has a velocity $\dot{\hat{q}}_m$ calculated from equation 4.11. Remaining step is to minimize the velocity difference between the robotic manipulator joint, $\dot{\hat{q}}_m$ and moCap-markers, $\dot{q}_n$. For each velocity component, v_{mn}, in equation 4.10 the distance is measured using equation 4.12.

$$\dot{\hat{q}} = W\dot{q}^T = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} \dot{q}_{marker1} & \dot{q}_{marker2} & \cdots & \dot{q}_{marker_n} \end{bmatrix}^T \tag{4.11}$$

$$\begin{aligned} v_{m,n}(\dot{\hat{q}}, \dot{q}) &= \sqrt{(\dot{\hat{q}}_m^2 - \dot{q}_n^2)} \\ &= \sqrt{(\dot{\hat{q}}_{m,x} - \dot{q}_{n,x})^2 + (\dot{\hat{q}}_{m,y} - \dot{q}_{n,y})^2 + (\dot{\hat{q}}_{m,z} - \dot{q}_{n,z})^2} \end{aligned} \tag{4.12}$$

where $\dot{\hat{q}}_m \in \mathbb{R}^3$ is the manipulator joint's velocity for joint m and $\dot{q}_n \in \mathbb{R}^3$ is moCap-velocity for marker n. Now, in order to determine the fitness of an individual a real-valued scalar is needed. Given the distance-matrix and velocity-matrix, one can sum up every components for both of them. As such, the fitness function becomes,

$$f(q, \theta_{mocap}, \dot{q}_{mocap}) = Distance + Velocity = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij} \tag{4.13}$$

Given our fitness function we are minimizing the distance between the joints and the error between the velocity. In order to get scalar we just sum up the elements in the matrices and the lower the value it is, the better it is. Therefore it would not make sense to maximize our fitness function but minimize. Hence our GA will be minimizing the function as,

$$fitness = \max f(q, \theta_{mocap}, \dot{q}_{mocap}) \tag{4.14}$$

4.2.3 Every Joints' Distance Minimization With Joint Limits (EJDM-JL)

Here, additional limitations to the robotic manipulator is added, restricting its overall movement in workspace with the intention of letting the manipulator

move with same restrictions as humans. However, the fitness is otherwise similar to the previous. Using the same approach from [53], where for every joints in the robotic manipulator, a joint-angle is assigned joint-limitation with lower bound θ_{lower} and upper bound θ_{upper}. These two values θ_{lower}, θ_{upper} are variables and can be changed to any arbitrary angles. Following up, means there has to be $2 \times m$ additional values to be added and table 4.2 gives an easy overview consisting the joint limits. m is the total amount of joints on the manipulator.

Joint i	θ_{lower}	θ_{upper}
1	$\theta_{1,\alpha}$	$\theta_{1,\beta}$
2	$\theta_{2,\alpha}$	$\theta_{2,\beta}$
$\vdots$	$\vdots$	$\vdots$
n	$\theta_{n,\alpha}$	$\theta_{n,\beta}$

Table 4.2: Table displaying the joint limits for one particular joint link i. α is joint minimum angle and β is joint maximum angle is measured in either degrees or radians.

From equations 4.1 the joint angles for the robot manipulator are received, and depending on the value must reduce or increase the joint angle if it exceeds the limits according to table 4.2. One way of reducing is to multiply the joint angle with number less than one, i.e $\gamma < 1$. As such, the new joint angle θ_n is determined by equation 4.15. For every generation in GA, the joint angle $\hat{\theta}$ is decreased or increased by γ. If the joint angle is within the bounds, then return the joint angle as it is.

$$\hat{\theta}_n = \begin{cases} \hat{\theta}_n\gamma, & \text{if } \hat{\theta}_n \geq \theta_{n,\beta}. \\ \hat{\theta}_n\gamma, & \text{if } \hat{\theta}_n < \theta_{n,\alpha}. \\ \hat{\theta}_n, & \text{otherwise.} \end{cases} \tag{4.15}$$

With the new joint angles the same approach applies as in previous section 4.2.2 where two matrices called distance from equation 4.9, and velocity from equation 4.12 are created.

$$f(q, \theta_{mocap}, \dot{q}_{mocap}) = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij} \tag{4.16}$$

Similar to previous section 4.2.2 the fitness function is maximized and fitness becomes,

$$fitness = \max f(q, \theta_{mocap}, \dot{q}_{mocap}) \tag{4.17}$$

4.2.4 Body Segment Distance Minimization (BCDM)

Here, the mapping function is more similar to one-to-one meaning, looking at one particular bodysegment of moCap-actor which can be interpreted as taking one body-part of moCap-actor. In this function the intended goal is to let the robotic manipulator follow trajectory with different amount of moCap-markers. First step is taking all of the corresponding moCap-actor's body limb which

could either be foot, arm, leg and etc.. and take those moCap-markers. Then use the same approach as in section 4.2.2 with equations 4.1, 4.8 and 4.10.

$$f(q, \theta_{mocap}, \dot{q}_{mocap}) = Distance + Velocity = \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij} \quad (4.18)$$

where d_{ij} and v_{ij} is derived from equations 4.9 and 4.12. Here we will explore whether more or less moCap-markers contribute to the overall imitation.

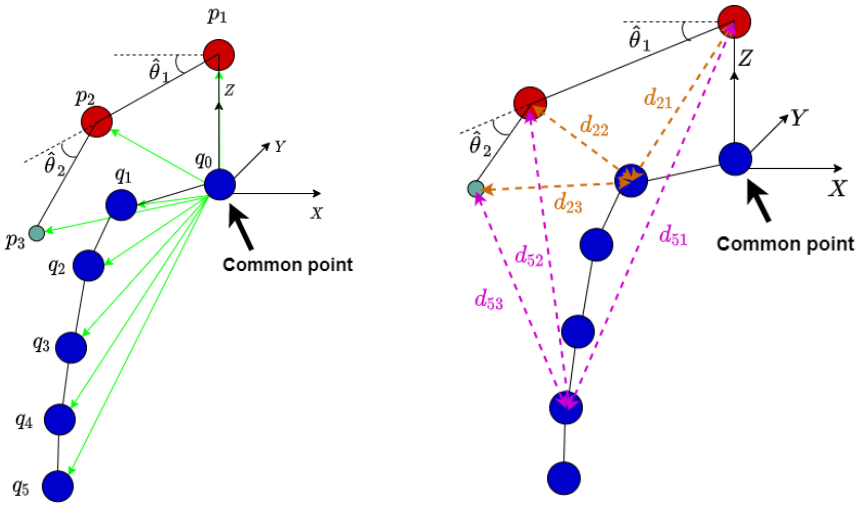


Figure 4.7: 4.7a shows the distance of the points with respect to the Common point and 4.7b shows Euclidean distance between the points with respect to the Common point.

4.2.5 Difference between BCDM and EJDM(-JL)

The main difference between BCDM and EJDM(-JL) is for the first one the mapping-method attempts to let the robotic manipulator perform an exact motion(Following trajectory) from moCap-motion with different amount of moCap-markers while the latter is attempting to perform creative movement.

What defines when the motion is "creative"[1] is vague and be interpreted. There are two ways to analyze the motion which are quantitavely and qualitatively. The first one is observing the data for instance the joint-angles and joint velocity to notice if there are rhythmic pattern that is mapped into the lower DOF robotic manipulator.

[1]Creative is defined as relating to or involving the use of the imagination or original ideas to create something

4.3 Implementation

With moCap-data, V-REP simulator and GA with mapping method, it is time to combine all into one complete module. Figure 4.8 displays the simple system overview of the setup and below will briefly explain each and every step in the figure.

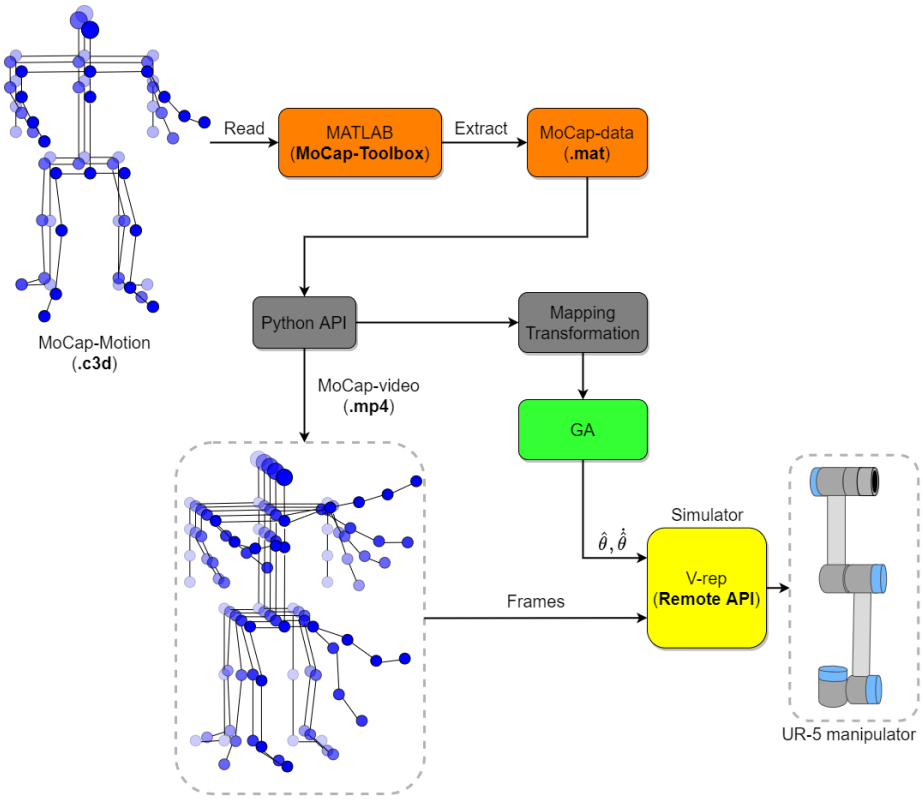


Figure 4.8: System Overview

4.3.1 Preprocessing moCap-data

This is the first step(orange) for loading the motion-capture file **.c3d** containing the movement, structure and position of every markers into matlab. It is also the step for processing the moCap-data and extracting it with the aid of moCap-toolbox from matlab[2] and create two files, one with the extracted data and the second one the video of the motion in **.mp4**.

[2]MatLab is multi-paradigm numerical computing programming language and mainly used for numerical computing.

4.3.2 Python API

The moCap-data in **.mat**-format arrives in Python API. Python has libraries and modules required to read the mentioned filetype. With the python API [3] there are perform two seperate tasks in this step but both of these tasks (4.3.3, 4.3.4) can be done independently and seperately offline.

4.3.3 Mapping Transformation

During this step and given the moCap-data the next action is to perform the mapping transformation required for both the moCap-markers and robotic manipulator to have common point which recalls to section 4.1.3. After transforming moCap-data appropriately, every data is in the right format and can be used on GA.

4.3.4 Video Preprocessing

The moCap-video **.mp4** is loaded and processed to be divided into the frames. When it is time to perform the simulation, the frames are shown with the motion with its corresponding moCap-data at the given timestep t. The movement from the moCap-video can be visualized simultaneously with the motion of robotic manipulator i.e. UR5 in V-REP as the settings for V-REP and parameters have been set for synchronous mode.

4.3.5 Genetic Algorithm - GA

Now arrives the process of executing the algorithm for optimization for the problem(green). The choice for GA is using DEAP [15] genetic algorithm library and generate a population of inidividuals whose genomes are string of weights with length $L = markers \times DOF$, and when during evaluation, the genome is reshaped the string into weight-matrix with shape as in equation 4.2. Given the weight-matrix, the applied fitness functions for the mapping-transformation is passed along with processed moCap-data from section 4.3.3. But, the important step is that the genome will always be a string for recombination, mutation and parent selection.

4.3.6 V-Rep

This is the last step in from the system overview(yellow) and here the applied the proposed solution from the GA. V-rep has built-in UR5-manipulator and can simply be extracted from the database and just ensure the robotic manipulator is communicating through Python Remote-API. During simulation a side-by-side frame from moCap-video and robotic movement and observe the mapped-transformation of the robot compared to the moCap-video. With the aid of V-rep it is also able to get feedback from the robotic manipulator for example joint-angles, joint-velocity, position of the end-effector and more. Figure 4.9 shows the simulation.

[3] Python is high-level, general-purpose programming language created by Guido van Rossum in 1991

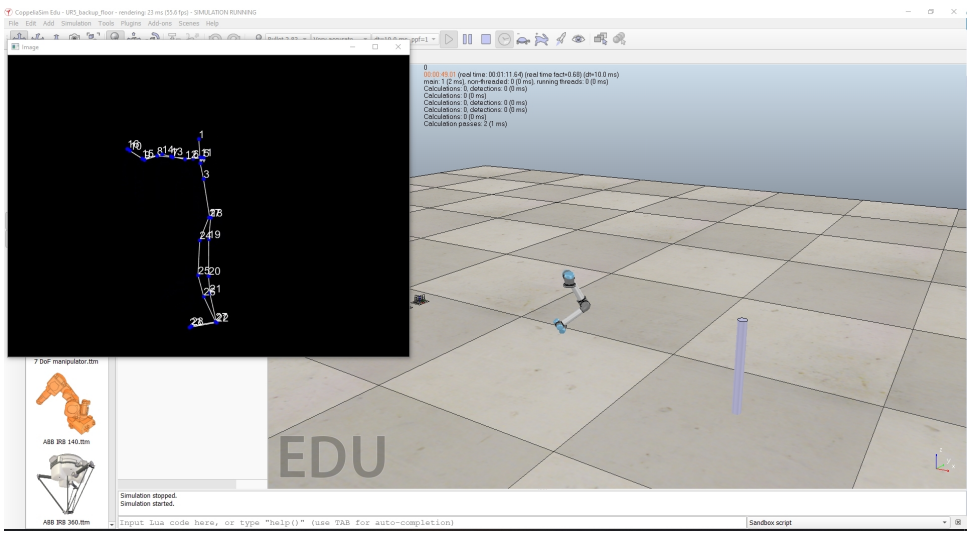


Figure 4.9: An UR5-manipulator is added to the environment and during simulation, packets are transmitted to the V-REP simulator containing the joint angles of the moCap-motion. A side-by-side view can be seen where the small left window is the moCap-motion and the V-REP is in the background. Due to asynchronous method, it manages to almost reach real-time simulation of R = 0.97, meaning the simulator is simulating close to real-time.

Chapter 5

Experiments and Results

5.1 Experiments

This chapter explains two main experiments, where in the first experiment we apply full-body moCap-motion onto the robot manipulator, more specifically UR5. The second experiment extracts small portion of the full-body moCap-motion and attempts to recreate the same trajectory for that body-part. Here, the suggested method of mapping us used and compared with euclidean distance and other related works in the field. All results and parameters will be presented in tables and graphs followed by analysis. At the end of this chapter will include brief summary of results and what was observed with our experiments.

5.1.1 UR5 DH-parameters

The proposed method is applied on UR5-manipulator and we can assume identical DH-table with the same DH-parameters as displayed on table 5.1 on every experiments.

Table 5.1: DH parameters and their values for UR5-manipulator

Link	d_i(m)	θ_i	a_i(m)	α_i
1	d_1	θ_1^*	0	0
2	0	θ_2^*	a_2	α_2
3	0	θ_3^*	a_3	0
4	d_4	θ_4^*	0	0
5	d_5	θ_5^*	0	α_5
6	d_6	θ_6^*	0	α_6

(a) DH-parameters overview

Link	d_i(m)	θ_i	a_i(m)	α_i
1	0.089	θ_1^*	0	0
2	0	θ_2^*	0.425	$\pi/2$
3	0	θ_3^*	0.39225	0
4	0.10915	θ_4^*	0	0
5	0.09456	θ_5^*	0	$\pi/2$
6	0.0823	θ_6^*	0	$-\pi/2$

(b) DH-parameters values

5.1.2 UR5-Joints Limits and Controller Values

The manufacturer of UR5-manipulator is also providing us the joint restrictions or limitation [27, p. 48] for its maximum velocity, maximum acceleration and

maximum torque of the joints. Here, we are interested in the physical properties of UR5 manipulator in order to simulate as accurately as possible.

V-rep has built-in PID controller inside most of the robotic joints and simultaneously we can select any values for different mechanical properties such as maximum allowed velocity for the joint, its torque and etc. Throughout every experiments we will use only PD components making it Proportional-Derivative controller.

Joint Properties	Value
q_{max}	2πrad
$\dot{q}_{max}$	3.2πrad
$\ddot{q}_{max}$	$15\frac{rad}{s^2}$
τ	150 Nm
$Proportional(P)$	250
$Integral\ (I)$	0
$Derivative\ (D)$	150

(a) Limitations for the physical joints of UR5

Joint i	θ_{lower}	θ_{upper}
1	-45	45
2	-75	75
3	-45	45
4	-75	75
5	-15	15
6	-35	35

(b) Joint limits for joint link i for EJDM-JL

Figure 5.1: Overview of UR5-Joint properties

5.1.3 GA-parameters and V-REP physics engine

For each and every experiment, a table will be displaying the parameters used and which fitness function we used. First, a brief explanation of every element from the table in order to avoid any potential misinterpretations. Starting from the top one has the generation count that shows long the GA is running the evolution. The next one is population size and the parent selection. The tournament size is written in parenthesis and specifiy the numbers of individuals competing towards each other. Following up one have crossover function followed by mutation operator. The percentage beside the Crossover probability determines the odds of performing the actual crossover between two individuals, while the percentage beside the mutation probability determine the odds of performing a random mutation after crossover. In the events of forgetfulness of how each and every one of them function, recall back to the chapter 2 in section 2.4.

In the same table include simulator settings such as its dynamics engine or in this instance, physics engine. First one is physics engine from V-REP. The next setting is how accurately the simulator should estimate. Physics time-step is how often it updates its values during simulation.

5.2 Experiment - Full Body

5.2.1 Testing Dancing-Motion with 7, 20 and 28 markers with EJDM and EJDM-JL

In this experiment the dancing-motion is applied for the robotic manipulator with 7, 20 and 28 moCap-markers and observe how the fitness EJDM and

EJDM-JL affect the overall general movement. Table 5.2 visualizes the parameters set for tests. For the joint limits, there is a table 5.1b displaying the desired joint angle to be within the bounded regions for the EJDM-JL function. The following dancing motion consist of 7324 frames and in the motion sequence it has periods of fast and slow moving movements.

GA Parameters	**Parameter value**
GA Total Runs	3
Generation	50
Population	300
Parent Selection	Tournament Selection (size=15)
Crossover	Two-Point Crossover, Crossover Probability (50%)
Mutation	Gaussian Mutation $\mathcal{N}(0,1)$, Mut. Probability (5%)
$W_{Init}(w_{min}, w_{max})$	-2, 2
Fitness Function	EDJM, EDJM-JL
$W_{7-markers}$	$W \in^{6\times7}$, $L = 42$
$W_{20-markers}$	$W \in^{6\times20}$, $L = 120$
$W_{28-markers}$	$W \in^{6\times28}$, $L = 168$
V-REP parameters	**Parameter**
Physics engine	Bullet 2.83
Physics settings	Very Accurate
Physics time-step	**dt**=10

Table 5.2: Genetic parameters for GA and V-REP parameters for 7, 20 and 28-markers

Motion: Dance	**Motion Description**
1	4 basic steps waltz
2	4 basic steps waltz with turning
3	wait
4	4 basic steps cha cha
5	4 promenades cha cha
6	4 cha cha turns

Table 5.3: Description of Dancing-motion

5.2.2 Results and analysis From Dancing-motion with 7,20 and 28-markers

On average the tests was performed on three runs and then averaging all three in order to get reliable average measurement from our test. The best total distance was 2856.38 with joint limits and 2860.14 without joint-limits. However, by comparing the average, there seems to be no difference between the two. The results can be seen in table 5.3 and figure 5.2 and all the runs was iterated over 50 generations and the total distance is reduced and show hints of going beyond it, especially if one observes closely at 5.2. Looking at 7-markers, it reached convergence rapidly after few generations, while for 20, and 28-markers

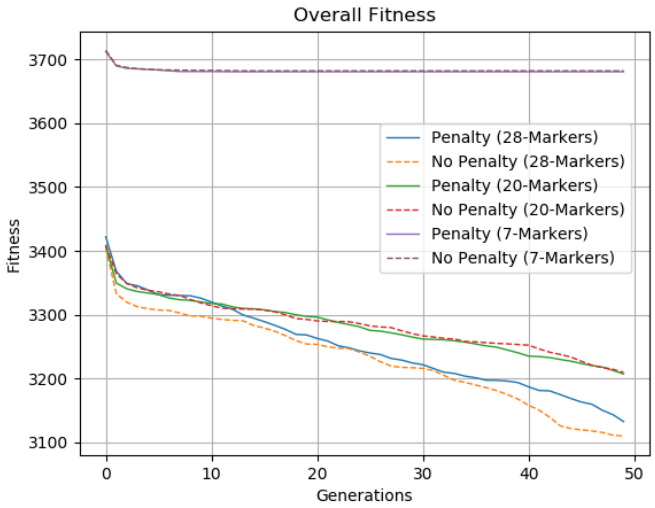


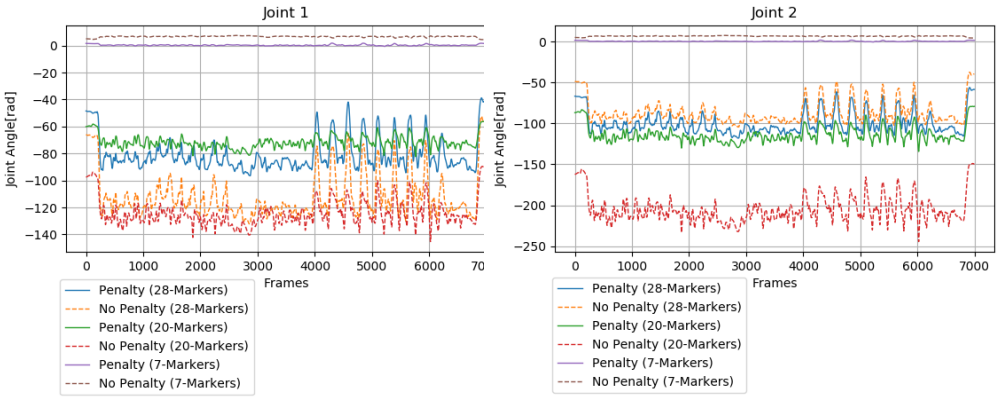
Figure 5.2: Average performance of three runs with GA using the 7,20, and 28 moCap markers using dancing-motion.

the total distance can be potentially be even lower. A boxplot 5.7 shows the comparison between the three different groups, with and without joint limit. The results show no clear difference between the groups.

During simulation the UR5 acts rapidly and moves in all directions, but through qualitative measurement finds there is evident of motion repeating itself in short intervals. Figures 5.4 ,5.5 and 5.6 displays the response of the joint angles from the UR5-manipulator.

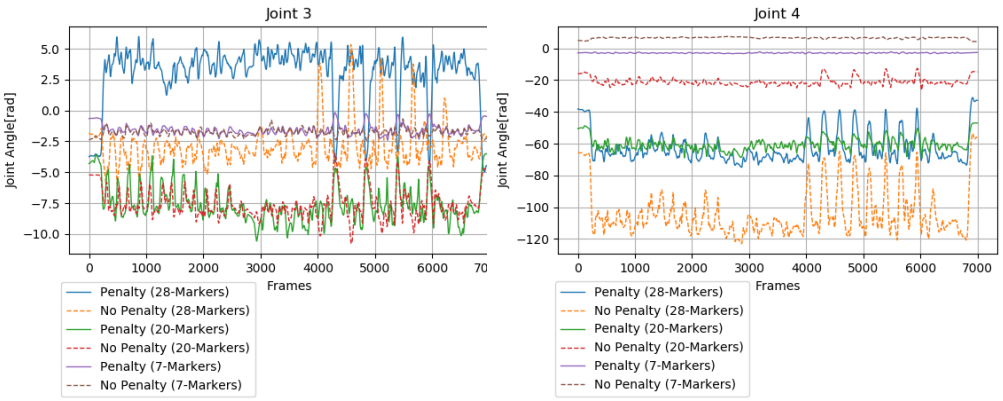
Type	Markers	Avg min	Avg max	Avg mean	Avg std
Dancing, EDJM-JL	28	2856.38	2970.20	2891.08	± 5.02
Dancing, EDJM	28	2860.14	2968.15	2891.52	± 5.04
Dancing, EDJM-JL	20	5256.48	5426.55	5275.73	± 3.37
Dancing, EDJM	20	5247.73	5405.49	5277.28	± 3.10
Dancing, EDJM-JL	7	3362.63	3412.04	3351.65	± 1.26
Dancing, EDJM	7	3362.63	3422.49	3365.55	± 1.16

Figure 5.3: Showing the average results of the three runs in dancing and the fitness value.



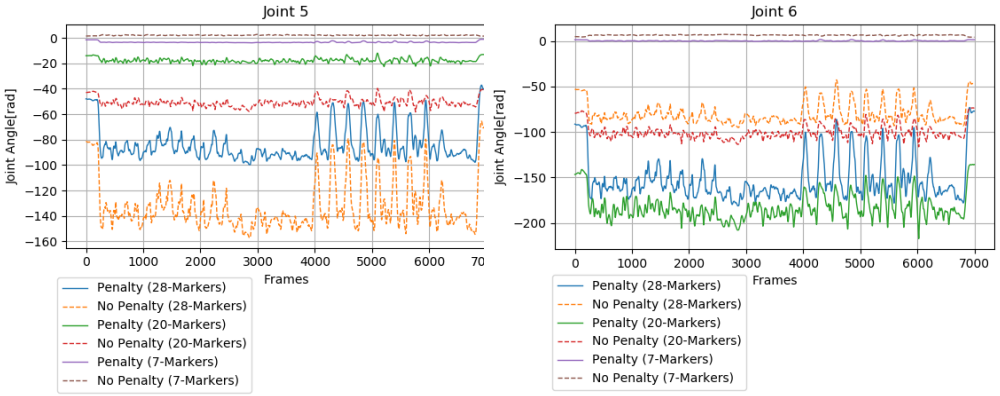
(a) (b)

Figure 5.4: Two plots displaying the joint angle values for Joint 1 and Joint 2 from UR5-robot with the Dancing-motion



(a) (b)

Figure 5.5: Two plots displaying the joint angle values for Joint 3 and Joint 4 from UR5-robot with the Dancing-motion



(a) (b)

Figure 5.6: Two plots displaying the joint angle values for Joint 5 and Joint 6 from UR5-robot with the Dancing-motion

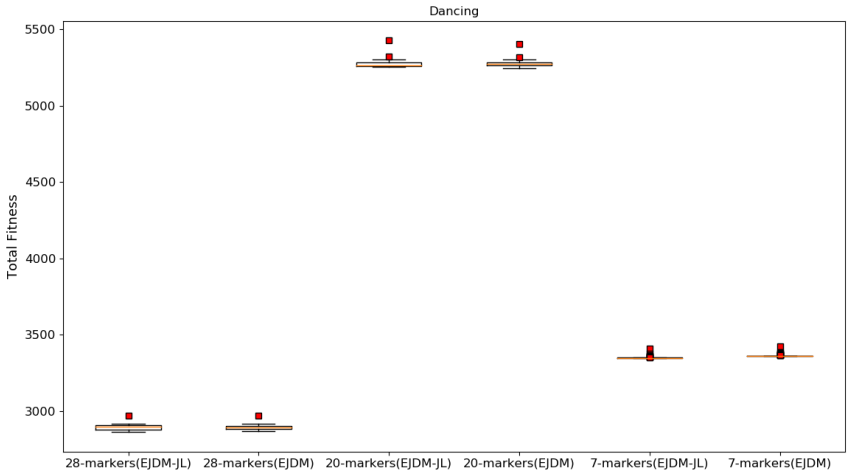


Figure 5.7: Boxplot showing the results from dancing with the various amount of markers. The median is the orange line and while the outliers are displayed with red squares.

5.2.3 Testing Walking-motion with 7, 20 and 28 markers using EJDM and EJDM-JL

Given the walking motion in 5.4 it consist of multiple steps and has in total 6995 frames. This motion is most simple where the movements are slow with some duration of it being fast, but not much faster than in workout or dance-motion. The same parameters for GA is the same from 5.2

Motion: Walk	Motion Description
1	walk 5 steps
2	turn around (right)
3	walk 5 steps (ducked)
4	walk 5 steps (backwards)
5	walk 5 steps (sideways, to the right, feet cross over)
6	3 double steps (sideways, to the left, no cross over)
7	3 double steps (sideways, to the right, cross over)
8	walk 5 steps (happily)
9	turn around (left)
10	walk 5 steps (sadly)
11	turn around (right)
12	walk 5 steps (creep)
13	turn around
14	walk 5 steps (shuffle)

Table 5.4: Overview of walking-motion

5.2.4 Results and analysis from Walking-motion with 7,20 and 28-markers

Same as in previous test, the GA was performed on three runs and then averaging of all the three in order to get an average measurement from this tests. The best total distance yielded was 3310.05 and without the joint limits it was 3132.72. The results from GA can be seen in table 5.9 and fig 5.8 and the total distance is reduced, but yields lower total distance than previous test. Looking again at 7-markers, the convergence reached local optima very quick compared to 20 and 28-markers. A boxplot 5.13 shows the comparison between the three different groups, with and without joint limit in more systematic way.

Running the optimal solution into simulation, the UR5 acts rapidly and moves in all directions but much slower, but there are no findings of motion repeating itself in short intervals as in previous test with dancing-motion. The joint response can seen on figures 5.10, 5.11 and 5.12.

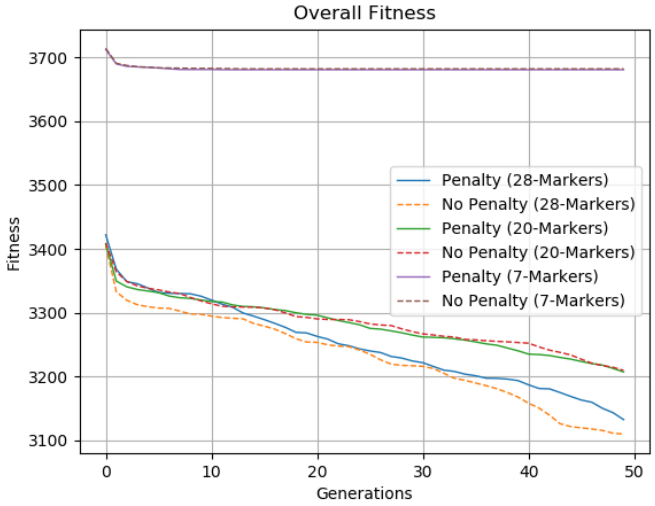
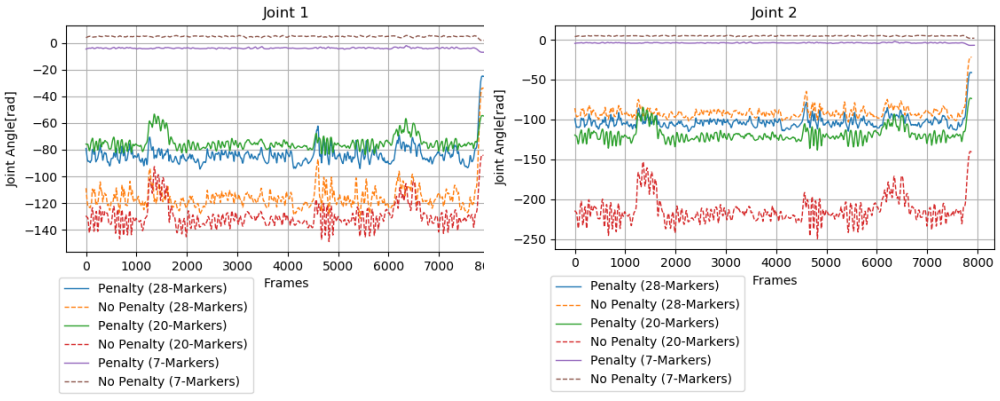


Figure 5.8: Average performance of three runs with GA using the 7,20, and 28 moCap markers using walking-motion.

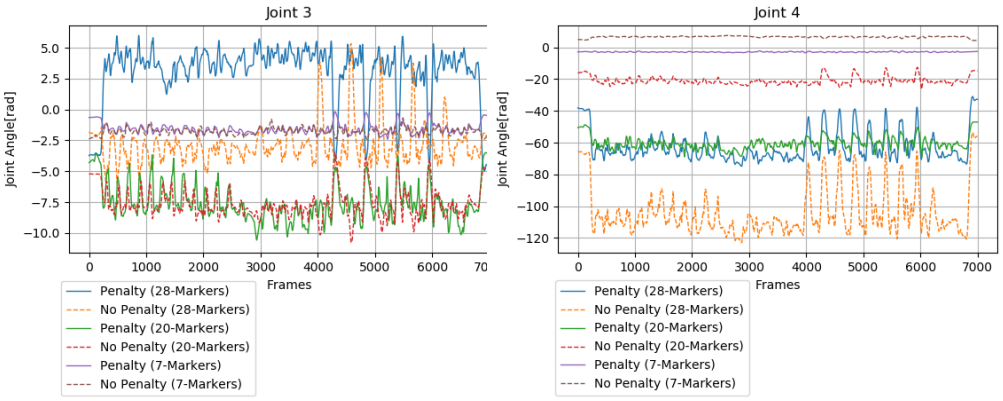
Type	Markers	Avg min	Avg max	Avg mean	Avg std
Walking, EDJM-JL	28	3132.72	3421.93	3256.42	± 3.55
Walking, EDJM	28	3110.05	3408.59	3230.92	± 10.73
Walking, EDJM-JL	20	3187.86	3425.84	3283.19	± 2.71
Walking, EDJM	20	3209.59	3409.31	3286.72	± 7.55
Walking, EDJM-JL	7	3678.01	3716.82	3683.07	± 1.87
Walking, EDJM	7	3678.01	3715.91	3681.50	± 2.26

Figure 5.9: Showing the average results of the three runs in dancing and the distance is measured in millimeters.



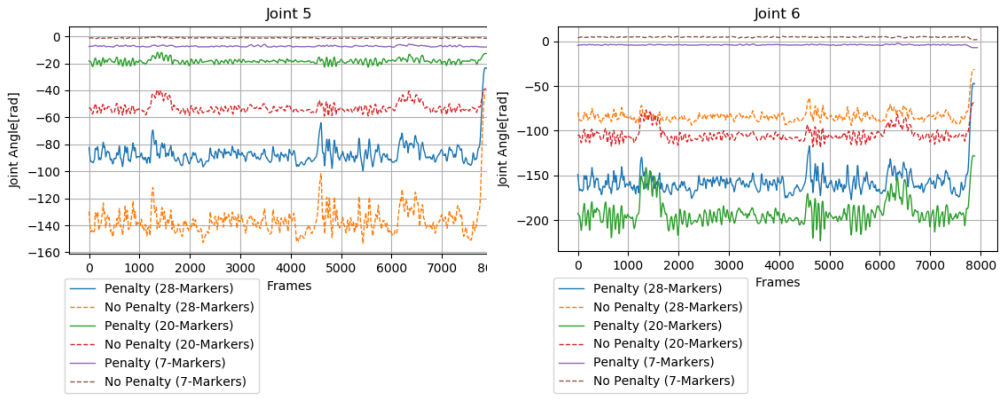
(a) (b)

Figure 5.10: Two plots displaying the joint angle values for the Joint 1 and Joint 2 from UR5-robot with the Walking-motion



(a) (b)

Figure 5.11: Two plots displaying the joint angle values for Joint 3 and Joint 4 from UR5-robot with the Walking-motion



(a) (b)

Figure 5.12: Two plots displaying the joint angle values for Joint 5 and Joint 6 from UR5-robot with the Walking-motion

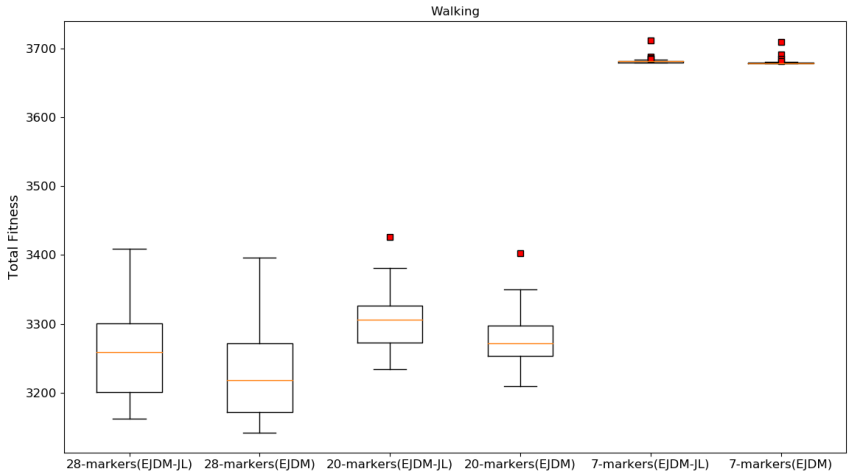


Figure 5.13: Boxplot showing the results from Walking with the various amount of markers. The median is the orange line and while the outliers are displayed with red squares.

5.2.5 Testing Workout-motion with 7, 20 and 28 markers

For the last motion it is performed with workout-motion and all of the used parameters are again given in5.2 and what is special about this motion is that the human moCap-performer performs rapid movements, especially in step 1 and step 2 from 5.5. It consists consists of 3316 frames meaning, there are few frames to learn from and more difficult.

Motion: Workout	Motion Description
1	4 jumping jacks
2	4 times skiing exercise
3	4 times elbow-to-knee exercise
4	4 squats

Table 5.5: Overview of Workout-motion

5.2.6 Results and analysis From Workout-motion with 7,20 and 28-markers

The GA is run three times and taking the averaging of all three runs yields the results from table 5.15 and fig 5.14. The best total distance yielded 2813.64 without the joint limits but with 2868.39. Further observations in the fig 5.15 shows the average distance for 20 markers is less than 28. However running the GA for longer than 50 generations could have reversed the effect given the graphs which seem to be reducing beyond the plot. A boxplot 5.19 shows the

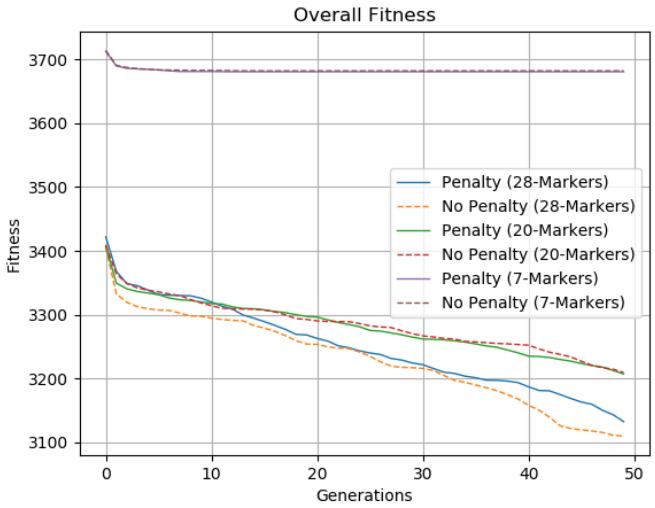


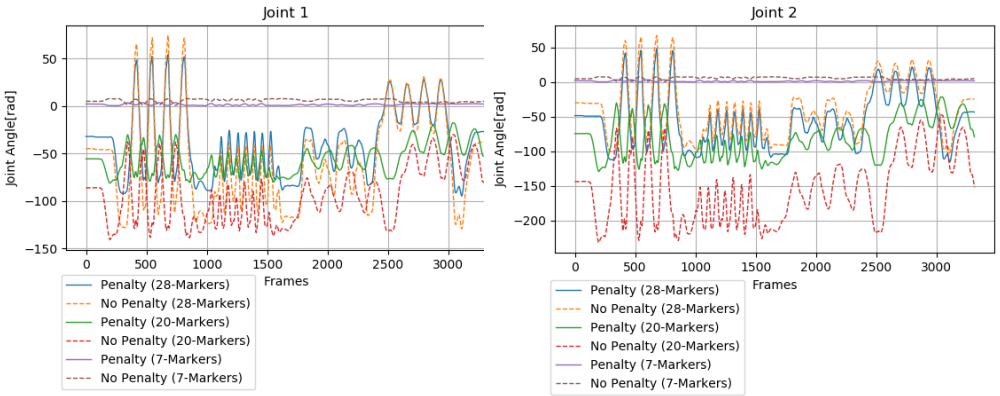
Figure 5.14: Average performance of three runs with GA using the 7,20, and 28 moCap markers using workout-motion.

comparison between the three different groups, with and without joint limit in more systematic way.

After GA has run, the optimal solution is applied onto the simulation, and the UR5 acts rapidly and moves even faster than both of the previous tests. The joint response can seen on figures 5.16, 5.17 and 5.17. Again, the joint angles are out of propotion, similar to previous tests.

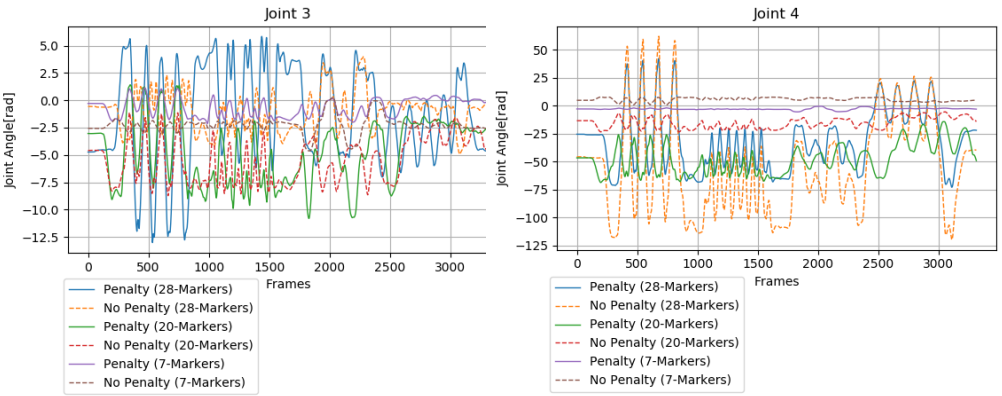
Type	Markers	Avg min	Avg max	Avg mean	Avg std
Workout, EDJM-JL	28	2868.39	3269.18	3069	± 4.24
Workout, EDJM	28	2813.64	2813.64	3050.01	± 4.65
Workout, EDJM-JL	20	2919.28	3148.05	3013.18	± 2.55
Workout, EDJM	20	2950.82	3140.89	3018.501	± 7.21
Workout, EDJM-JL	7	3341.42	3424.72	3353.39	± 3.56
Workout, EDJM	7	3341.42	3424.30	3353.35	± 1.18

Figure 5.15: Showing the average results of the three runs in dancing and the distance is measured in millimeters.



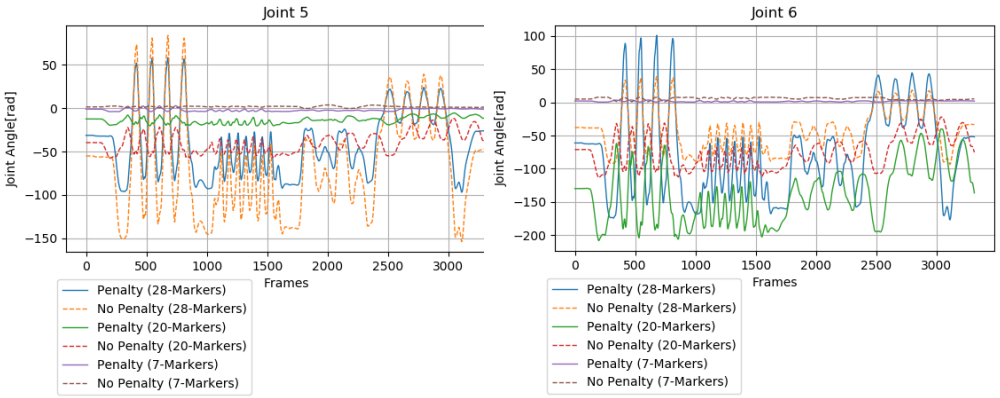
(a) (b)

Figure 5.16: Two plots displaying the joint angle values for Joint 5 and Joint 6 from UR5-robot with the Workout-motion



(a) (b)

Figure 5.17: Two plots displaying the joint angle values for Joint 5 and Joint 6 from UR5-robot with the Workout-motion



(a) (b)

Figure 5.18: Two plots displaying the joint angle values for Joint 5 and Joint 6 from UR5-robot with the Workout-motion

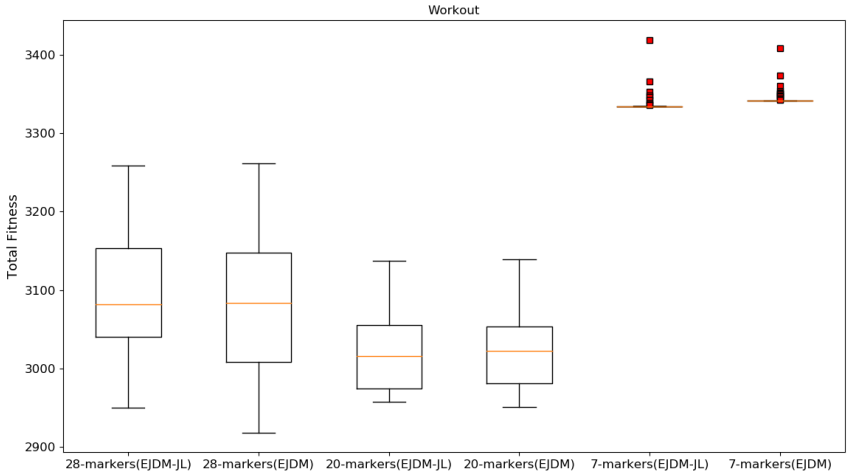
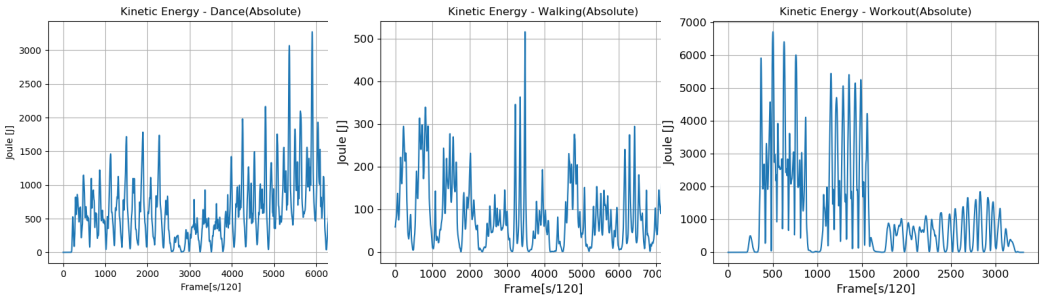


Figure 5.19: Boxplot showing the results from workout with the various amount of markers. The median is the orange line and while the outliers are displayed with red squares.



(a) (b) (c)

Figure 5.20: Figure(Left) 5.20a shows the total kinetic energy of Dancing-motion. Second figure(middle) 5.20b shows the kinetic energy of Walking-motion. Last figure(right) 5.20c shows the total kinetic energy of Workout-motion

5.2.7 Similarity between Joint angle and Kinetic Energy

The tests from the experiment share some similarity with the total kinetic energy of the motion from figure 5.20. The joint angles from all three motions seem to match the kinetic energy from the actual motion without having prior knowledge to it. The peaks from the joint angles seem to closely related to the actual motion. Example, fig 5.4 is similar to dancing-motion and fig 5.11 is similar to the total kinetic energy of walking-motion. Last, but not leastly fig 5.18 is similar to workout-motion. Further details and analysis will be discussed in the next chapter.

5.3 Experiment - Body Segments

In this experiment, By extracting the right-arm moCap-markers from the moCap-motion, the goal is to experiment whether various types of moving arm motion is able to be mapped onto the UR5-manipulator and optimizing the weights using GA for achieving the objective. The following experiment will primarily focus on 2,4 and 8 markers using EJDM and Euclidean distance.

It is worth mentioning the method EDJM has been tweaked for this test. Here the velocity term is excluded from the test and only the distance term remain. This is specified in the table 5.6. The same limitations for UR5-manipulator remains the same. Additional data with the distance or fitness table as in for example tables 5.9, 5.15, 5.3 is excluded as the comparison is done using another metric.

GA Parameters	**Parameter value**
GA Total Runs	3
Generation	50
Population	300
Parent Selection	Tournament Selection (size=15)
Crossover	Two-Point Crossover, Crossover Probability (50%)
Mutation	Gaussian Mutation $\mathcal{N}(0,1)$, Mut. Probability (5%)
$W_{Init}(w_{min}, w_{max})$	-2, 2
Fitness Function	EDJM (Velocity excluded), Euclidean
$W_{2-markers}$	$W \in^{6\times2}$, $L = 12$
$W_{4-markers}$	$W \in^{6\times4}$, $L = 24$
$W_{8-markers}$	$W \in^{6\times8}$, $L = 48$
V-REP parameters	**Parameter**
Physics engine	Bullet 2.83
Physics settings	Very Accurate
Physics time-step	**dt**=10

Table 5.6: Genetic parameters for GA and V-REP parameters for 2, 4 and 8-markers using BCDM

5.3.1 Testing I: Workout-motion with BCDM

In this experiment the right-arm motion is extracted using 2,4 and 8 markers to observe the proposed solution is able to follow the same trajectory as the workout-motion. Description of the motion can be seen in 5.5.

5.3.2 Results and analysis From Workout-motion with BCDM

Every test is run three times and the average data is measured to gain reliable data for the motion trajectory generated by the UR5. The results can be seen from table 5.7 where the mean-squared error (MSE) is used for the difference in position between moCap-markers and the joint position. The total distance from fig 5.21 show rapid convergence towards an optimal solution. All of the

joint's response from workout motion can be seen on figures 5.22, 5.23, 5.24, 5.25 and 5.26.

The best result is by using 4 markers and the second best is 6 markers and on third place it is 8-markers.

Method	**Markers**	**MSE**(*pos*)	**MSE** (*rad*)
EJDM	8 Markers	881.21	1.10
Euclidean	8 Markers	501.05	0.85
EJDM + Euclidean	8 Markers	569.36	0.79
EJDM	4 Markers	353	7.8
Euclidean	4 Markers	117	1.47
EJDM + Euclidean	4 Markers	183.07	1.38
EJDM	2 Markers	743	1.38
Euclidean	2 Markers	823.04	0.07
EJDM + Euclidean	2 Markers	826.39	0.09
EJDM	3 Markers	318.02	11.41
Euclidean	3 Markers	312.64	5.94
EJDM + Euclidean	3 Markers	352.57	5.02
(One-to-One mapping)	6 markers	520.107	0.0

Table 5.7: Overview of mean square error for Workout-motion

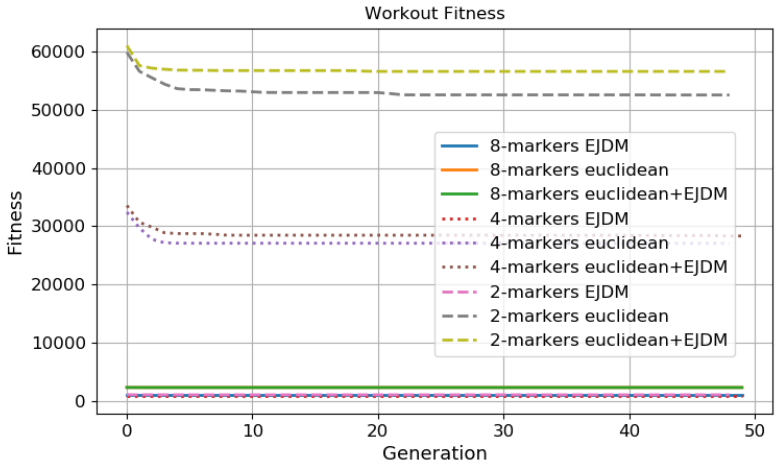
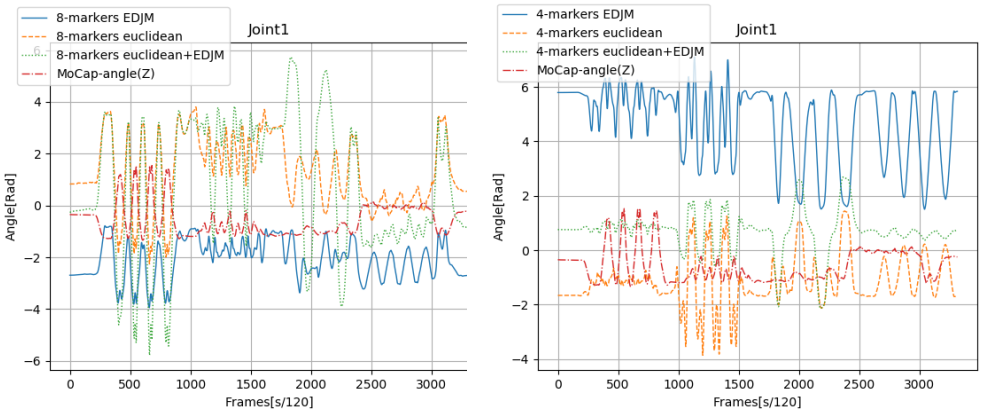
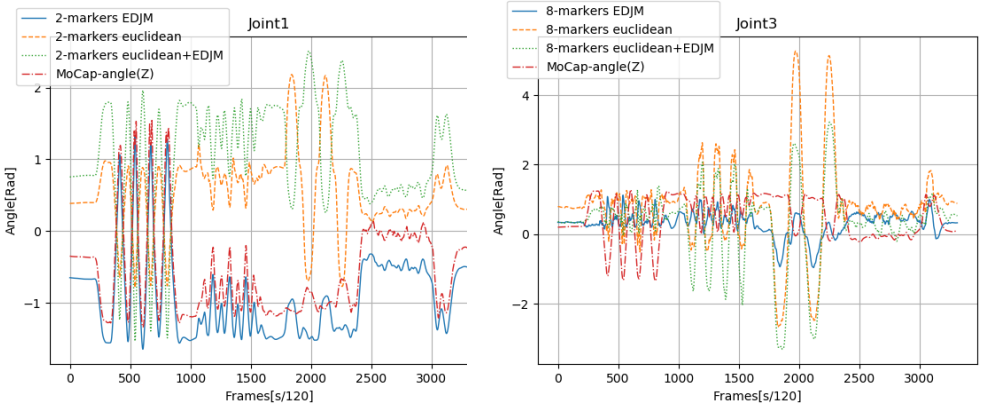


Figure 5.21: Figure showing how the different methods performs during GA using BCDM with workout.



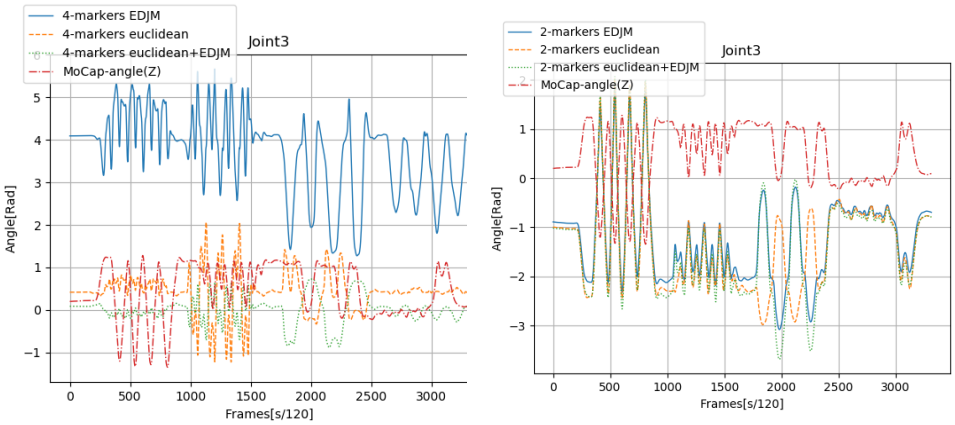
(a) (b)

Figure 5.22: Two plots displaying the joint angle values for dancing for the Joint 1 from UR5-robot using BCDM with the three different methods by using 4 and 8 markers.



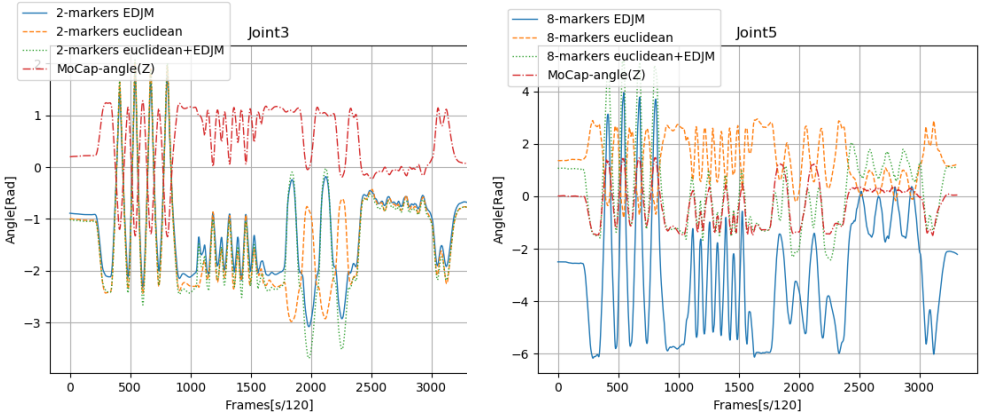
(a) (b)

Figure 5.23: Two plots displaying the joint angle values for dancing for the Joint 1 and Joint 3 from UR5-robot using BCDM with the three different methods by using 2 and 8 markers.



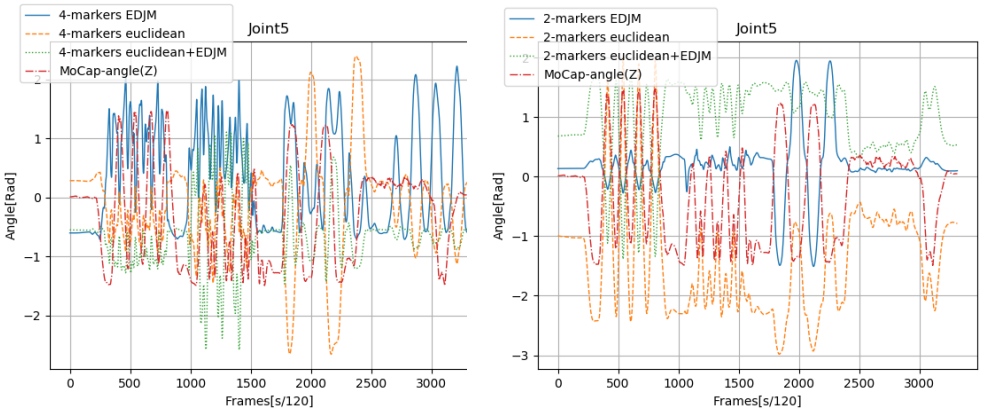
(a) (b)

Figure 5.24: Two plots displaying the joint angle values for dancing for the Joint 3 from UR5-robot using BCDM with the three different methods by using 2 and 4 markers.



(a) (b)

Figure 5.25: Two plots displaying the joint angle values for dancing for the Joint 3 and Joint 5 from UR5-robot using BCDM with the three different methods by using 2 and 8 markers.



(a) (b)

Figure 5.26: Two plots displaying the joint angle values for dancing for the Joint 5 from UR5-robot using BCDM with the three different methods by using 2 and 4 markers.

5.3.3 Testing II: Dance-motion with BCDM

Similar to previous test 5.3.1 In this test the right-arm motion is extracted using 2,4 and 8 markers with dancing-motion. The description of dancing-motion is described in table 5.5 but here the same right arm markers is extracted from dance-motion and table 5.8 shows the results from the different test and comparison with the same related works as in workout-motion.

5.3.4 Results and Analysis From Dance-motion using BCDM

Similar to previous test, every test with dancing-motion was run three times to get reliable data and the results from dancing motion with BCDM can be seen on tab 5.8 and fig 5.27 where the first table displays the mean-square-error between the moCap-markers and joint positions. Looking at fig 5.27, the fitness decreases and converges rapidly towards a solution. The pattern of rapid convergence can be compared to workout-motion where the GA is stuck in local-optima. All of the joint's response from workout motion can be seen on figures 5.28, 5.29, 5.30, 5.31 and 5.32.

Comparing by MSE, the best result is with 4-markers and followed up by 6-markers and then on third with 8-markers.

Method	**Markers**	**MSE**(*pos*)	**MSE** (*rad*)
EJDM	8 Markers	925.41	6.21
Euclidean	8 Markers	463.79	0.71
EJDM + Euclidean	8 Markers	458.10	1.60
EJDM	4 Markers	287.98	1.28
Euclidean	4 Markers	213.25	1.40
EJDM + Euclidean	4 Markers	227.69	0.97
EJDM	2 Markers	904.94	0.11
Euclidean	2 Markers	529.58	0.02
EJDM + Euclidean	2 Markers	724.01	0.18
EJDM	3 Markers	230.22	11.41
Euclidean	3 Markers	461.54	5.94
EJDM + Euclidean	3 Markers	432.82	5.02
(One-to-One mapping)	6 markers	276.05	0.0

Table 5.8: Overview of mean square error for Dance-motion

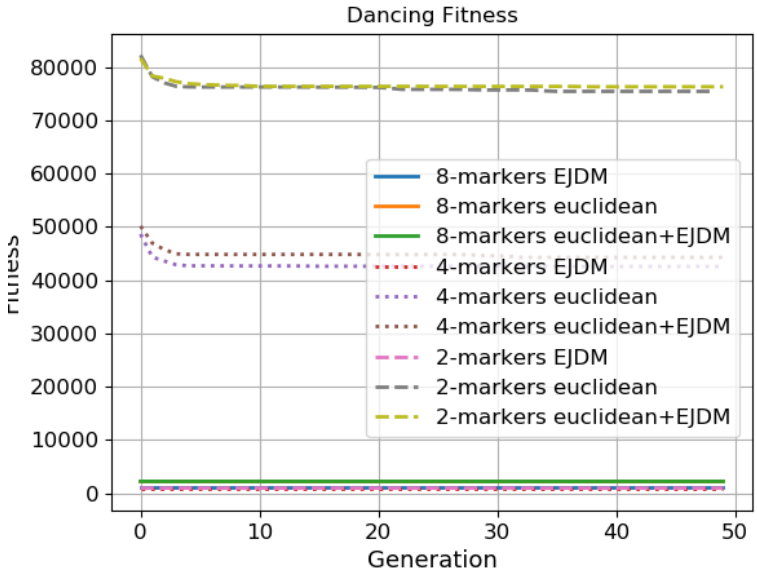
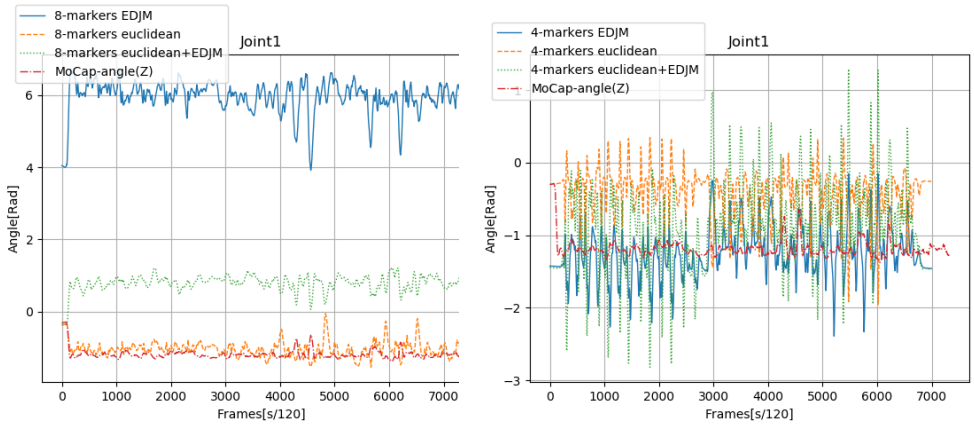
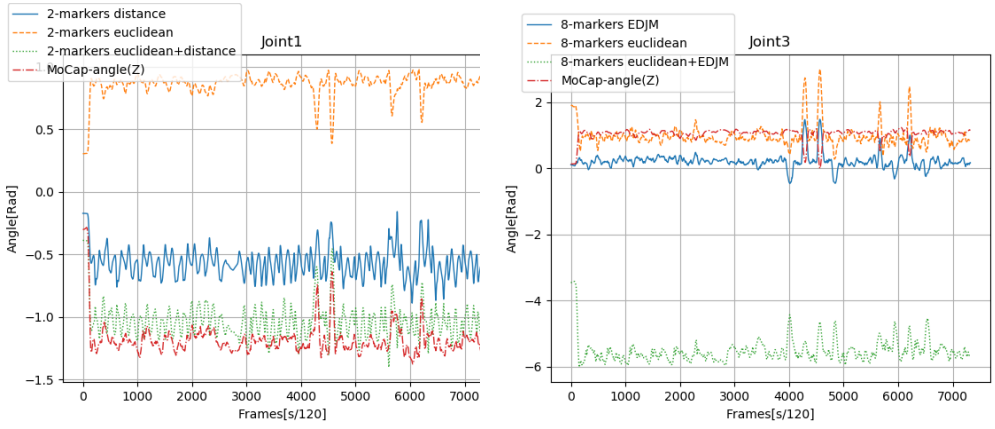


Figure 5.27: Figure showing how the different methods performs during GA using BCDM with Dancing.



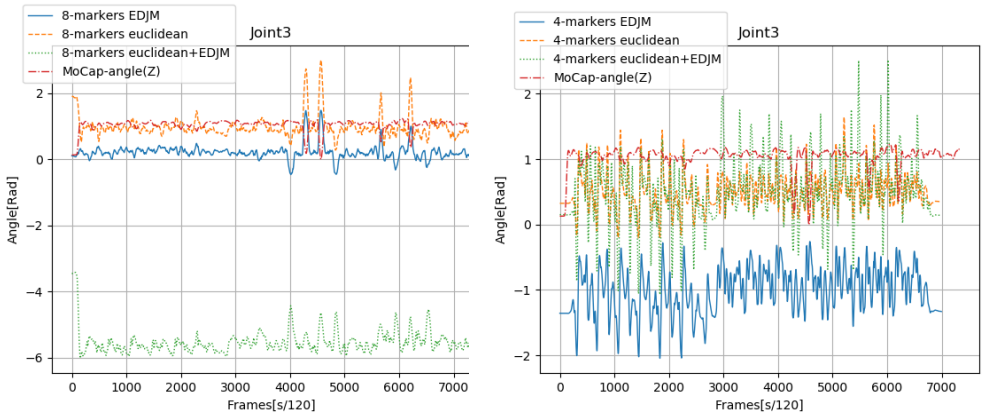
(a) (b)

Figure 5.28: Two plots displaying the joint angle values for dancing for the Joint 1 from UR5-robot using BCDM with the three different methods by using 4 and 8 markers.



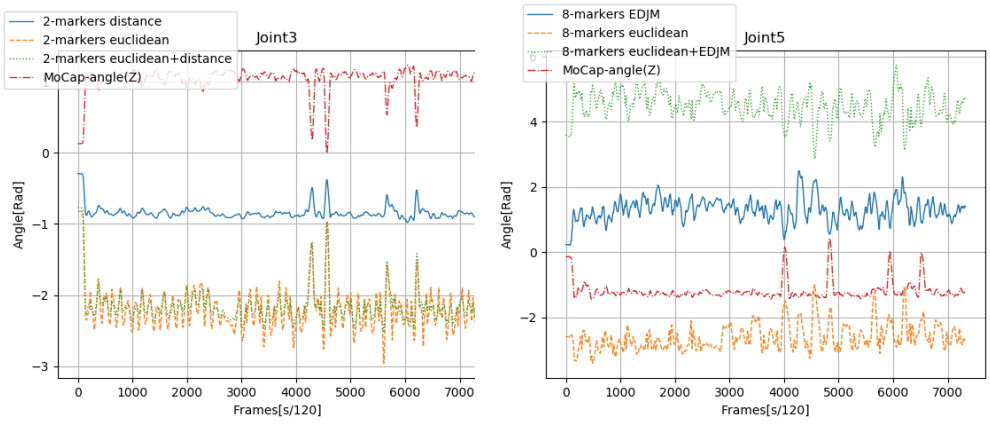
(a) (b)

Figure 5.29: Two plots displaying the joint angle values for dancing for the Joint 1 and Joint 3 from UR5-robot using BCDM with the three different methods by using 2 and 8 markers.



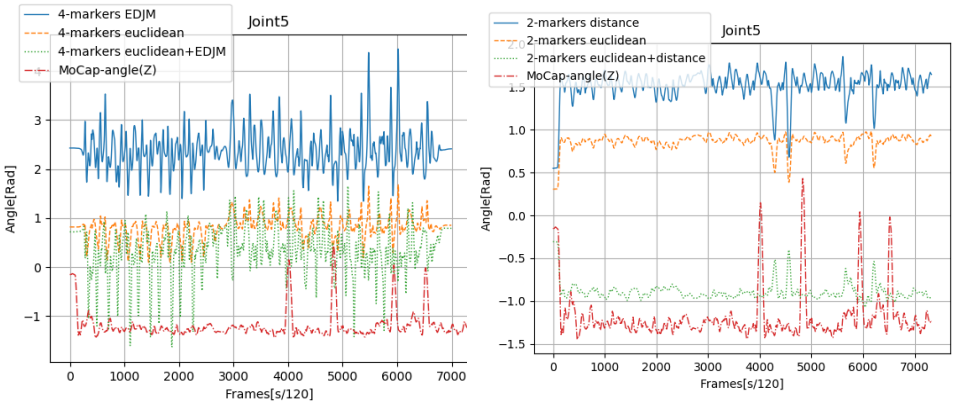
(a) (b)

Figure 5.30: Two plots displaying the joint angle values for dancing for the Joint 3 from UR5-robot using BCDM with the three different methods by using 4 and 8 markers



(a) (b)

Figure 5.31: Two plots displaying the joint angle values for dancing for the Joint 3 and Joint 5 from UR5-robot using BCDM with the three different methods by using 2, 8 markers



(a) (b)

Figure 5.32: Two plots displaying the joint angle values for dancing for the Joint 5 from UR5-robot using BCDM with the three different methods by using 2, 4 markers

5.3.5 Summary

The two experiments presented in this chapter were testing with multiple amount of moCap-markers and fitness function and weight matrix size. Through extensive experimentation, more markers do contribute to the mapping relations as the additional spatiotemporal data prove to increase the performance and the total distance minimized. In the first experiment, the larger the weight-matrix, the more of the total distance can be minimized between the robotic manipulator and moCap-actor with 28-markers. 7-Markers proved to be the worse. For the second experiment, the results varied depending on the amount of markers. After testing both of the experiments, the proposed method encountered new problems and challenges that was not considered. These new problems and challenges will be discussed in the next chapter.

Chapter 6

Discussion

6.1 General discussion

This chapter concludes the thesis with a general discussion of results, limitations, discoveries and issues, followed by a conclusion and suggestions for future work.

6.1.1 Full Body motion

The UR5-manipulator is used as the main robot for this experiment and during simulation the robot behaved unrealistically which comes mainly from the optimized weight matrix. The GA is stochastic in nature and require multiple runs in order to get reliable data. Looking at joint angles from the results one can notice the radians is out of proportion compared to experiment II-BCDM. The results demonstrate two things. First, this suggests that the velocity term from the fitness function is the only term contributing to the overall fitness that can be minimized the most. Second, the moCap-velocity is absolute and since the velocity term sums them, yields higher values. As such, the analysis from result during GA may suggest there are other methods for inhereting the velocity term. The total distance minimized with EJDML-JL always yielded larger total distance than without. A way to counter the velocity term from dominating is to implement velocity constraints.

However, looking apart from the scaled joint angles, there are findings from the joint angles that show similarity between the joint angles and the kinetic energy from the moCap-actors without the GA ever having prior knowledge of the kinetic energy. Comparing kinetic energy from figure 5.20c and joint 1, joint 3 and joint 5 from figures 5.16a 5.17a, and 5.18a it can be seen that the joint angle share the same peak at the different frames from the kinetic energy. One might argue that the weights just scale the joint angle from the moCap-data and hence yields the similarity of kinetic energy. However, this does not neglect that the total distance between the robotic manipulator and motion capture motion is minimized simultaneously. Alternatively, it could simply mean that the GA minimizes the velocity term the most or that the velocity term dominates as both distance and velocity is integrated together. This may imply that the total distance and velocity minimized is associated with characteristics of the moCap-motion.

From the GA's performance, the results from table 5.3, 5.15 and 5.9 demonstrate two things. First, the more moCap-markers that are available or used in the moCap structure, the more GA manages to minimize the distance from all of the markers. Second, the weights for the motion are used to minimize the total distance and create unique movements and motion to lower DOF robot. These results indicate that more markers do contribute to the overall motion as more weights are initialized and thus, more values to be optimized with the GA. Figures 5.2, 5.8 and 5.14 shows the total distance and velocity being minimized. For 7-markers, the convergence is reaching its solution between 5 - 7 generations. This is due to lack of few weights and illustrating that the one way or even perhaps the only way to reach out of the solution is through mutation. Solely relying on mutation with 5 % probability of getting out or finding solutions is unreliable. For 20-markers, the total distance and velocity seem to go down or show some traits of going further below after generation 50. But the GA is finished when the generation reaches 50. At last, 28-markers the total fitness is minimized even further and similar to 20-markers, the total fitnesscan go beyond 50 generations.

6.1.2 BCDM

From the experience from the first experiments, the velocity term from EDJM was excluded due to overshoot of the joint angles, leading to unrealistic movement from the robotic manipulator. The results in previous chapter at the table 5.6 in the fitness function clearly visualizes that.

The results from the second experiment proved to be far better as the joint angles generated were reasonably good as can be seen on figures 5.28, 5.29, 5.30, 5.31 and 5.32 for dancing motion. The joint angles did not overshoot, and since the manipulator is now being mapped according to the right arm movement, means the fitness function or the approached method can be used to compare towards other related research within the area. This will be explained later in section 6.1.4. The Mean-squared-error (MSE) was calculated and it was done by taking the distance between the joints of the UR5-robot and the moCap-markers and yielding the error in millimeters (mm). The MSE for joint angles was also performed and this one was calculated different. This was done by using the quantifying of motion, the joint angle is summed as explained in chapter 2 from section 3.2.11 where the sum from UR5 joint angles and moCap-angles was performed and then the MSE used to measure the difference between them. Again from the results, the fitness function with Euclidean distance prove to yield the least difference between joints and moCap-markers.

This was done with various amount of markers and the table 5.7 from results section with workout motion revealed that using different amount of markers does impact the performance GA. 4-markers proved to yield the best result with the fitness function using Euclidean distance. From the table 5.8 featuring dance-motion yields similar results. 6-markers or the one-to-one mapping prove to give quite good results but this is because the joint angles can be applied directly and the manipulator just copy them and hence reaches lower MSE. One issue with MSE is that the more markers, the higher the value as the amount of markers accumulate the MSE. Under certain assumptions can this be construed as good way of estimating the performance of BCDM. One such assumption is when comparing the values from two different marker groups occurs.

The plot from fig 5.27 for dancing and the plot from fig 5.21 show that fewer markers gives high total fitness which is the total distance and then it reaches towards early convergence after 3 - 4 generations. The total distance for the more markers seem to go towards zero or are closer to zero while fewer markers points towards in the opposite direction. Many of results seem to also overlap and these are very small and difficult to observe because the GA manages to minimize more distance of the markers and joint from the manipulator. Especially from the orange line that does not show on figures 5.27, 5.21. This is due to the orange line being behind the green line or equal to the orange line.

6.1.3 Limitations Of The Experiment(s)

There are several limitations to the proposed approach. The main concern are the weights and comparing to machine learning algorithms which consists of millions of parameters such as AlexNet [26] to solve more complex functions, the few weight parameters cannot search for optimal solutions because of the vast search space of the thesis' problem.

Another limitation of approach are the fitness functions with both EJDM(-JL) and BCDM, the fitness functions is the lack of time-term. The moCap-data may be time-dependent while the fitness function does not. The fitness function can be interpreted as minimizing the distance of posture between robotic manipulator and moCap, however the time duration between reaching it in the initial frame to the final frame is excluded. Certain trajectories require more time to reach and some trajectory requires slower, however this is not the case in the EJDM and BCDM methods.

A potential problem that may arise is if the robotic manipulator is much larger than the moCap-skeleton. Since EJDM(-JL) and BCDM are minimizing the distance between the joints and moCap markers it could cause the robotic manipulator to collide with itself. The fitness function does take collision into consideration.

One concern about the findings are the BCDM fitness function proving to yield inferior results than standard Euclidean distance between the pairs of markers and joints, despite varying the count of moCap-markers. Regarding this limitation, it could be argued the distance between the pairs of marker and joint is sufficient enough for minimizing the distance moCap-markers and joints and not with every joint as is the proposed approach.

6.1.4 Comparison of previous studies

A recent discovery from Chih Liu et al. [32] has a more complex objective function the potential issue with EJDM(-JL) and BCDM by implementing the collision and self-collision. This proved to be succesful for their experiment and robust. However, they matched the amount of skeleton coordinate i.e 6 reference markers with 6 DOf robot. This produces similar results using BCDM with 6-markers as it performed reasonably well. A similar pattern of results was obtained [34] where 20 markers were used on both the human and android upperbody. This indicate one-to-one mapping is a robust approach for mapping and heuristic methods or optimization methods are used to refine the kinematic-relation between the human and the robot manipulator.

6.2 Benefits of weight-matrix, EJDM(-JL) and BCDM

Using the EJDM(-JL) and BCDM reduces the amount of analytical equations required to make the mapping as one only needs the DH-parameters for a robotic manipulator. The forward-kinematic can be derived from DH-parameters and also the Jacobian matrix can also be found. These are procedural steps and is simple to follow given one has some knowledge about robot modeling. The fitness function is easily extendable by adding more features, and since the genes are random floating numbers, allows possibilities of exploitation. This means the weights can be represent anything. However, the weight-matrix in this thesis can be thought as mapping-matrix, where its values represent the relationship between moCap-actor and robotic manipulator.

The following approach is easily customizable for selecting markers and initializing the weight matrix. The proposed function solves minimizing distance between markers and joints even if the DOF from the robotic manipulator does not match the moCap-markers($DOF \neq markers$). Not only does it excels at following imitating moCap-motion, but it is also quick to optimize with GA.

6.3 Conclusion

The thesis researched if using different amount of motion capture markers affects the performance of mapping to robotic manipulator by optimizing weights using evolutionary algorithm. From the introduction section in chapter, an hypothesis was set for the thesis to prove and disprove by fulfilling the two required goals.

The first goal was **Observing how the motion-capture data is mapped into lower degree-of-freedom robots and if it is possible to capture the some unique characteristics or the *"essence"* of the motion**. The first experiment presented in the thesis proved that the evolution algorithm with EJDM managed to capture kinetic energy representation while reducing the distance between the joints and markers using various amount of markers. By using more markers, it is able to capture more of the nature of the motion whereas using fewer did not. During simulation, the joint angles generated were higher than expected and was not satisfactory. Given the accomplishments, the first goal of the thesis can lightheartedly be considered successfully.

The second goal was **Extract a portion of the human motion capture data for example human arm and determine if the robotic manipulator can imitate the same motion trajectory with different amount of markers.** The second experiment performed much better as a results of its predecessor i.e first experiment. By excluding the velocity term from BCDM, the joint angles were within reasonable range. As such, the results proved that BCDM manages to imitate human motion where using more markers does help its adaptation to the motion capture through minimizing distance. This goal proved to be successful.

The thesis addressed that mapping from motion capture to robotic manipulator is challenging and there are a lot of exploration or further investigation needed to gain desirable results. The question remains if the hypothesis is proved or disproved. The results from the first goal of the thesis was subpar,

however the second goal was fulfilled as the robot manipulator managed to follow the trajectory of the motion capture through optimization from GA. Thus, the hypothesis was achieved.

6.4 Future Work

GA is stochastic and needs to be run multiple times in order to receive reliable data to measure from. This is very time-consuming process and as a consequence, not every planned features or experiments were carried out. If the continuation of the thesis would resume again, there are several suggestions for improvement and ideas to create. The following section will describe the most important ones.

6.4.1 Jacobian-Matrix

There are plentiful of work that can be improved in our fitness function. First one is deriving the jacobian using the forward-kinematics from Denavit-Hartenberg and include in the fitness function or its values. What the jacobian provide is angular velocity and linear velocity with respect to base-frame. With the Jacobian Matrix it can prevent singularities from occuring. That is the robotic manipulators losing one degree-of-freedom. But the jacobian can also be used to solve inverse-kinematics [11]. This means the jacobian of the robotic manipulator contain a lot of analytical equations that can be used to solve additional tasks if it were to be implemented into the fitness function. An idea worth investigating is to see if the inverse-kinematics can solved with only motion-capture data.

6.4.2 Other motion capture limbs

An interesting aspect of the method is removing some of the limbs i.e markers from the motion-capture and observe if the response is different and observe how the robotic manipulator adapts to its new skeleton or layout. As it stands now, the thesis only experimented with different amount of markers at full-body and right arm motion. The idea is to remove some of the markers in the structure either from various places and explore the robotic manipulators adaptation with the fitness function.

6.4.3 Synchronized motion from Simulation

For the motion trajectory generated from the motion it is possible to implement Recurrent Neural Network [25] abbreviated RNN for synchronized movement by predicting the next joint angles for the time-step in the motion. During simulation, the frame and the video does not match perfectly. RNNs have yielded great results for sequential predictions based on the reviews from Zachary et al. [31].

6.4.4 Experimenting with Robotic Manipulator

Similar to motion-capture limbs, another future work could be to test fitness function on multiple robots with different amount of Degree-of-Freedoms in attempt to compare the performances between two robot manipulators. A few

mentioned robotic manipulators to be experimented on could perhaps be 6 DOF KUKA-arm [1] and 7-DOF Panda-arm [2]. This is to test different limb-length and possible offsets that may exists in the manipulator, and prismatic joints could also be used.

6.4.5 Third Fitness Function

During the development phase, there were planned to implement three methods in total, but the third one was excluded due to time-constraints. What if it was possible to generate unique movements from the knowledge of kinetic energy? The idea was to take the kinetic energy and only pass it as the only input for the GA and generate a motion that maximizes the movement with respect to kinetic energy based of a moCap-actor.

[1] Developed by KUKA Robotics
[2] Developed by FRANKA EMIKA

Bibliography

[1] Kazuhiko Akachi et al. “Development of Humanoid Robot HRP-3P”. In: 2005. DOI: 10.1109/ICHR.2005.1573544.

[2] Jacopo Aleotti, Alexander Skoglund and Tom Duckett. “Position Teaching of a Robot Arm by Demonstration with a Wearable Input Device”. In: 2019.

[3] Paul S. Andrews, Susan Stepney and Jon Timmis. “*Simulation as a Scientific Instrument*”. In: Simulation as a Scientific Instrument. 2012.

[4] Alexander M. Aurand, Jonathan S. Dufour and William S. Marras. “Accuracy map of an optical Motion Capture System with 42 or 21 Cameras in Large Measurement Volume”. In: *Journal of Bimechanics* (2017).

[5] *Average Height and Weight in the World*. https://www.worlddata.info/average-bodyheight.php. Accessed: 27.02.2020.

[6] Dominik Belter, Jan Wietrzykowski and Piotr Skyrzypczynski. “Employing Natural Terrain Semantics in Motion Planning for a multi-legged Robot”. In: 2018. DOI: https://doi.org/10.1007/s10846-018-0865-x.

[7] Taha Beyrouthy and Samer Al Kork. “EEG Mind Controlled Smart Prosthetic Arm - A Comprenhensive Study”. In: vol. 2. 2017, pp. 891–899. DOI: 10.25046/aj0203111.

[8] Aude Billard et al. “Robot Programming by Demonstration”. In: Springer Handbook of Robotics, 2008, pp. 1371–1394. DOI: DOI:10.1007/978-3-540-30301-5_60.

[9] “Body Segment Parameters”. In: (1966).

[10] ”Jochen Bomm et al. “Documentation: Mocap Database HDM05”. In: (2017).

[11] Samuel R. Buss. “Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods”. In: 2009.

[12] S. Butterworth. “”On the Theory of Filter Amplifiers””. In: *In Wireless Engineer* 7 (1930), 536–541.

[13] Wei hua Chieng. “The controllable Ball joint Mechanism”. In: 2006, pp. 1151–1158. DOI: 10.1299/jsmec.49.1151.

[14] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2015.

[15] Félix-Antoine Fortin et al. “DEAP: Evolutionary Algorithms Made Easy”. In: *Journal of Machine Learning Research* 13 (2012), pp. 2171–2175.

[16] Optimal Mutation Probability for Genetic Algorithm. “R. N. Greenwell and J. E. Angus and M. Finck”. In: 1995, pp. 1–11.

[17] David E. Goldberg. *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley Professional, 1989.

[18] Anand Gopalakrishnan et al. “A Neural Temporal Model for Human Motion Prediction”. In: 2018, pp. 12116–12125.

[19] Zichang Guo et al. “A Reinforcement Learning Approach for Inverse Kinematics of Arm Robot”. In: IEEE, 2019, pp. 95–99. DOI: https://doi.org/10.1145/3351180.3351199.

[20] Ahmed Hussein et al. “Imitation Learning: A survey of Learning Methods”. In: 2017.

[21] Tommi Jantunen et al. “Experiences from collecting motion capture data on continous signing”. In: 2012.

[22] Li-Hu Jhang, Carlo Santiago and Chian-Song Chiu. “Multi-Sensor Based glove Control of An Industrail Mobile Robot Arm”. In: IEEE, 2017. DOI: 10.1109/CACS.2017.8284267.

[23] Bahaa Ibraheem Kazem, Ali Ibrahim Mahdi and Ali Talib Qudah. “Motion Planning for a Robot Arm by Using Genetic Algorithm”. In: 2008.

[24] Kazuo Kiguchi, Takakazu Tanaka and Toshio Fukuda. “Neuro-Fuzzy Control of a Robotic Exoskeleton With EMG-Signals”. In: IEEE, 2004, pp. 481–490. DOI: 10.1109/TFUZZ.2004.832525.

[25] Philipp Kratzer, Marc Toussaint and Jim Mainprice. “Motion Prediction with Recurrent Neural Network Dynamical Models and Trajectory Optimization”. In: 2019.

[26] Alex Krizhevsky, Illya Sutskever and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: 2012. DOI: DOI: 10.1145/3065386.

[27] Katharina Kufieta. *Force estimation in Robotic Manipulators: Modeling, Simulation and Experiments*.

[28] Shuai Li, Yunong Zhang and Long Jin. “Kinematic Control of Redundant Manipulators Using Neural Networks”. In: vol. 28. 2017. DOI: 10.1109/TNNLS.2016.2574363.

[29] Minas Liarokapis, Panagiotis Artemiadis and Kostas Kyriakopoulos. “Functional Anthropomorphism for Human to Robot Motion Mapping”. In: (2012).

[30] *Linear Least Squares*. https://en.wikipedia.org/wiki/Linear_least_squares. Accessed: 27.02.2020.

[31] Zachary C. Lipton and John Berkowitz. “A Critical Review of Recurrent Neural Networks for Sequence Learning”. In: 2015.

[32] Chih-Yin Liu et al. “Motion Imitation and Augmentation System for a Six Degrees of Freedom Dual-Arm Robot”. In: IEEE, 2019. DOI: 10.1109/ACCESS.2019.2949019.

[33] Pedro Martin and Jose del R. Millan. “Robot arm reaching through neural inversion and reinforcement learning”. In: 1999. DOI: https://doi.org/10.1016/S0921-8890(99)00100-1.

[34] Daisuke Matsui et al. "Generating Natural Motion in an Android by Mapping Human Motion". In: IEEE, 2005. DOI: 10.1109/IROS.2005.1545125.

[35] Mitchell Melanie. *An Introduction to Genetic Algorthms*. Cambridge, Masschusetts, 1999.

[36] Pierre Merriaux et al. "A study of Vicon Positioning Performance". In: 2017.

[37] Mustafa Waad Abdullah Michael Weyrich. "Concept of a Three D.O.F Spherical-Joint Gripper for Industrial Robots". In: 2013. DOI: 10.1109/ETFA.2013.6648121.

[38] Stefano Michieletto, Davide Zanin and Emanuele Menegatti. "*NAO robot simulation for service robotics purposess*". In: NAO Robot Simulation for Service Robotics Purposes. November, 2013.

[39] *Misconceptions about evolution*. URL: "https://evolution.berkeley.edu/evolibrary/misconceptions_faq.php#a3".

[40] Lucas Nogueira. "*Comparative Analysis Between Gazebo and V-REP Robotic Simulators*". In: Comparative Analysis Between Gazebo and V-REP Robotic Simulators. December, 2014.

[41] Kristian Nymoen. *MoCap-Toolbox Extenstions*. Accessed: 2019. 2019. URL: https://github.com/krisny/mcarray.

[42] Ga-Ram Park et al. "Human-like Catching Motion of Humanoid Using Evolutionary Algorithm(EA)-based Imitation Learning". In: IEEE, 2009. DOI: 10.1109/ROMAN.2009.5326070.

[43] Ill-Woo Park et al. "Mechanical Design of humanoid Robot Platform KHR-3". In: IEEE, 2005.

[44] Steven L. Peck. "*Simulation as experiment: a philosophical reassessment for biological modeling*". In: Simulation as experiment: a philosophical reassessment for biological modeling. October, 2004.

[45] Jose Manuel Peula et al. "Pure Reactive Behavior learning using Case Based Reasoning for a vision based 4-legged robot". In: vol. 57. 2008, pp. 688–699. DOI: https://doi.org/10.1016/j.robot.2008.11.003.

[46] Lenka Pitonakova et al. "*Feature and performance comparison of the V-rep, Gazebo and ARGoS*". In: Feature and performance comparison of the V-rep, Gazebo and ARGoS. February, 2018.

[47] Jerry Pratt. "Intuitive Control of a Planar Bipedal Walking Robot". In: IEEE, 1998. DOI: 10.1109/ROBOT.1998.680611.

[48] Softbank Robotics. *NAO-Robot*. NAO-Robot. 2019.

[49] Universal Robots. *UR5-Manipulator*. UR5-Manipulator. 2019.

[50] Miguel Rocha and Jose Neves. "Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation". In: 1999. DOI: 10.1007/978-3-540-48765-4_16.

[51] Eric Rohmer, Surja P. N. Singh and Marc Freese. "*V-rep: A Versatile and Scalable Robot Simulation Framework*". In: V-rep: A Versatile and Scalable Robot Simulation Framework. November, 2013.

[52] Nizar Rokbani, Abdallah Zaidi and Adel.M Alimi. "Prototyping a Biped Robot Using an Educational Robotics Kit". In: 2012. DOI: 10.1109/ICEELI.2012.6360682.

[53] Takaaki Shiratori et al. "Temporal Scaling of Upper Body Motion for Sound Feedback System of a Dancing Humanoid Robot". In: 2007.

[54] B. Siciliano et al. *Robotics: Modeling, Planning and Control*. Springer, 2009.

[55] "Soothing and differentation of data by simplified least squares procedures". In: *Analytical Chemistry* 36 (1964), pp. 1627 –1639.

[56] Christopher Stanton, Stanton Bogdanovych and Edward Ratanasena. "Teleoperation of Humanoid Robot using Full-Body motion Capture, Example movements and Machine Learning". In: 2012.

[57] Petri Toiviainen and Birgitta Burger. *Motion-Capture Toolbox*. 7.10.2015.

[58] Petri Toivianen and Birgitta Burger. *Motion-Capture Toolbox Documentation*. Version 1.5. 2015. URL: https://www.jyu.fi/hytk/fi/laitokset/mutku/en/research/materials/mocaptoolbox/MCT_manual_v1.5.pdf.

[59] Ales Ude et al. "Automatic Generation of Kinematic Models for the Conversion of Human Motion Capture into Humanoid Robot Motion". In: 2000.

[60] Anantkumar J. Umbarkar and P.D. Sheth. "Crossover Operators in Genetic Algorithms: A review". In: (2015). DOI: 10.21917/ijsc.2015.0150.

[61] *Vicon Motion Capture Optical System*. Accessed: 02.03.2020. URL: https://www.vicon.com/.

[62] *Weight of Human Body Parts*. URL: http://robslink.com/SAS/democd79/body_part_weights.htm.

[63] Darrell Whitley and Jeffrey Horn. "Genetic Algorithm Difficulty and the Modality of Fitness Landscapes". In: (1995). DOI: https://doi.org/10.1016/B978-1-55860-356-1.50016-9.

[64] Jie Yang et al. "Walking Pattern Generation for Humanoid Robot Considering Upper Body Motion". In: IEEE, 2006. DOI: 10.1109/IROS.2006.282078.